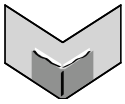
The Journal of Machine Learning Research
Volume 9
Print-Archive Edition

Pages 1–1436



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) 23-48 Submitted 5/07; Published 1/08

Linear-Time Computation of Similarity Measures for Sequential Data

Konrad Rieck KONRAD.RIECK@FIRST.FRAUNHOFER.DE
Pavel Laskov PAVEL.LASKOV@FIRST.FRAUNHOFER.DE
Fraunhofer FIRST.IDA
Kekuléstraße 7
12489 Berlin, Germany

Editor: Nello Cristianini

Abstract

Efficient and expressive comparison of sequences is an essential procedure for learning with sequential data. In this article we propose a generic framework for computation of similarity measures for sequences, covering various kernel, distance and non-metric similarity functions. The basis for comparison is embedding of sequences using a formal language, such as a set of natural words, *k*-grams or all contiguous subsequences. As realizations of the framework we provide linear-time algorithms of different complexity and capabilities using sorted arrays, tries and suffix trees as underlying data structures.

Experiments on data sets from bioinformatics, text processing and computer security illustrate the efficiency of the proposed algorithms—enabling peak performances of up to 10^6 pairwise comparisons per second. The utility of distances and non-metric similarity measures for sequences as alternatives to string kernels is demonstrated in applications of text categorization, network intrusion detection and transcription site recognition in DNA.

Keywords: string kernels, string distances, learning with sequential data

1. Introduction

Sequences of discrete symbols are one of the fundamental data representations in computer science. A great deal of applications—from search engines to document ranking, from gene finding to prediction of protein functions, from network surveillance tools to anti-virus programs—critically depend on analysis of sequential data. Providing an interface to sequential data is therefore an essential prerequisite for applications of machine learning in these domains.

Machine learning algorithms have been traditionally designed for vectorial data—probably due to the availability of well-defined calculus and mathematical analysis tools. A large body of such learning algorithms, however, can be formulated in terms of pairwise relationships between objects, which imposes a much looser constraint on the type of data that can be handled. Thus, a powerful abstraction between algorithms and data representations can be established.

The most prominent example of such abstraction is *kernel-based learning* (e.g., Müller et al., 2001; Schölkopf and Smola, 2002) in which pairwise relationships between objects are expressed by a Mercer kernel, an inner product in a reproducing kernel Hilbert space. Following the seminal work of Boser et al. (1992), various learning methods have been re-formulated in terms of kernels, such as PCA (Schölkopf et al., 1998b), ridge regression (Cherkassky et al., 1999), ICA (Harmeling et al., 2003) and many others. Although the initial motivation for the "kernel trick" was to allow efficient computation of an inner product in high-dimensional feature spaces, the importance of an

abstraction from data representation has been quickly realized (e.g., Vapnik, 1995). Consequently kernel-based methods have been proposed for non-vectorial domains, such as analysis of images (e.g., Schölkopf et al., 1998a; Chapelle et al., 1999), sequences (e.g., Jaakkola et al., 2000; Watkins, 2000; Zien et al., 2000) and structured data (e.g., Collins and Duffy, 2002; Gärtner et al., 2004).

Although kernel-based learning has gained significant attention in recent years, a Mercer kernel is only one of many possibilities for defining pairwise relationships between objects. Numerous applications exist for which relationships are defined as metric or non-metric distances (e.g., Anderberg, 1973; Jacobs et al., 2000; von Luxburg and Bousquet, 2004), similarity or dissimilarity measures (e.g., Graepel et al., 1999; Roth et al., 2003; Laub and Müller, 2004; Laub et al., 2006) or non-positive kernel functions (e.g., Ong et al., 2004; Haasdonk, 2005). It is therefore imperative to address pairwise comparison of objects in a most general setup.

The aim of this contribution is to develop a *general framework* for pairwise comparison of sequences. Its generality is manifested by the ability to handle a large number of kernel functions, distances and non-metric similarity measures. From considerations of efficiency, we focus on algorithms with linear-time asymptotic complexity in the sequence lengths—at the expense of narrowing the scope of similarity measures that can be handled. For example, we do not consider super-linear comparison algorithms such as the Levenshtein distance (Levenshtein, 1966) and the all-subsequences kernel (Lodhi et al., 2002).

The basis of our framework is embedding of sequences in a high-dimensional feature space using a *formal language*, a classical tool of computer science for modeling semantics of sequences. Some examples of such languages have been previously used for string kernels, such as the bag-of-words, k-gram or contiguous-subsequence kernel. Our formalization allows one to use a much larger set of possible languages in a unified fashion, for example subsequences defined by a finite set of delimiters or position-dependent languages. A further advantage of embedding using formal languages is separation of embedding models from algorithms, which allows one to investigate different data structures to obtain optimal efficiency in practice.

Several data structures have been previously considered for specific similarity measures, such as hash tables (Damashek, 1995), sorted arrays (Sonnenburg et al., 2007), tries (Leslie et al., 2002; Shawe-Taylor and Cristianini, 2004; Rieck et al., 2006), suffix trees using matching statistics (Vishwanathan and Smola, 2004), suffix trees using recursive matching (Rieck et al., 2007) and suffix arrays (Teo and Vishwanathan, 2006). All of these data structures allow one to develop linear-time algorithms for computation of certain similarity measures. Most of them are also suitable for the general framework developed in this paper; however, certain trade-offs exist between their asymptotic run-time complexity, implementation difficulty and restrictions on embedding languages they can handle. To provide an insight into these issues, we propose and analyze three data structures suitable for our framework: *sorted arrays*, *tries* and *suffix trees* with an extension to suffix arrays. The message of our analysis, supported by experimental evaluation, is that the choice of an optimal data structure depends on the embedding language to be used.

This article is organized as followed: In Section 2 we review related work on sequence comparison. In Section 3 we introduce a general framework for computation of similarity measures for sequences. Algorithms and data structures for linear-time computation are presented in Section 4. We evaluate the run-time performance and demonstrate the utility of distinct similarity measures in Section 5. The article is concluded in Section 6

2. Related Work

Assessing the similarity of two sequences is a classical problem of computer science. First approaches, the string distances of Hamming (1950) and Levenshtein (1966), originated in the domain of telecommunication for detection of erroneous data transmissions. The degree of dissimilarity between two sequences is determined by computing the shortest trace of operations—insertions, deletions and substitutions—that transform one sequence into the other (Sankoff and Kruskal, 1983). Applications in bioinformatics motivated extensions and adaptions of this concept, for example defining sequence similarity in terms of local and global alignments (Needleman and Wunsch, 1970; Smith and Waterman, 1981). However, similarity measures based on the Hamming distance are restricted to sequences of equal length and measures derived from the Levenshtein distance (e.g., Liao and Noble, 2003; Vert et al., 2004), come at the price of computational complexity: No linear-time algorithm for determining the shortest trace of operations is currently known. One of the fastest exact algorithms runs in $O(n^2/\log n)$ for sequences of length n (Masek and Patterson, 1980).

A different approach to sequence comparison originated in the field of information retrieval with the vector space or bag-of-words model (Salton et al., 1975; Salton, 1979). Textual documents are embedded into a vector space spanned by weighted frequencies of contained words. The similarity of two documents is assessed by an inner-product between the corresponding vectors. This concept was extended to k-grams—k consecutive characters or words—in the domain of natural language processing and computer linguistic (e.g., Suen, 1979; Cavnar and Trenkle, 1994; Damashek, 1995). The idea of determining similarity of sequences by an inner-product was revived in kernel-based learning in the form of bag-of-words kernels (e.g., Joachims, 1998; Drucker et al., 1999; Joachims, 2002) and various string kernels (e.g., Zien et al., 2000; Leslie et al., 2002; Vishwanathan and Smola, 2004). Moreover, research in bioinformatics and text processing advanced the capabilities of string kernels, for example, by considering gaps, mismatches and positions in sequences (e.g., Lodhi et al., 2002; Leslie et al., 2003; Leslie and Kuang, 2004; Rousu and Shawe-Taylor, 2005; Rätsch et al., 2007). The comparison framework proposed in this article shares the concept of embedding sequences with all of the above kernels, in fact most of the linear-time string kernels (e.g., Joachims, 1998; Leslie et al., 2002; Vishwanathan and Smola, 2004) are enclosed in the framework.

A further alternative for comparison of sequences are kernels derived from generative probability models, such as the Fisher kernel (Jaakkola et al., 2000) and the TOP kernel (Tsuda et al., 2002). Provided a generative model, for example a HMM trained over a corpus of sequences or modeled from prior knowledge, these kernel functions essentially correspond to inner-products of partial derivatives over model parameters. The approach enables the design of highly specific similarity measures which exploit the rich structure of generative models, for example, for prediction of DNA splice sites (Rätsch and Sonnenburg, 2004). The run-time complexity of the kernel computation, however, is determined by the number of model parameters, so that only simple models yield run-time linear in the sequence lengths. Moreover, obtaining a suitable parameter estimate for a probabilistic model can be difficult or even infeasible in practical applications.

3. Similarity Measures for Sequential Data

Before introducing the framework for computation of similarity measures, we need to establish some basic notation. A *sequence* $\mathbf{x}$ is a concatenation of symbols from an *alphabet* $\mathcal{A}$. The set of all possible concatenations of symbols from $\mathcal{A}$ is denoted by $\mathcal{A}^*$ and the set of all concatenations

of length k by $\mathcal{A}^k$. A *formal language* $L \subseteq \mathcal{A}^*$ is any set of finite-length sequences drawn from $\mathcal{A}$ (Hopcroft and Motwani, 2001). The length of a sequence $\mathbf{x}$ is denoted by $|\mathbf{x}|$ and the size of the alphabet by $|\mathcal{A}|$. A contiguous subsequence w of $\mathbf{x}$ is denoted by $w \sqsubseteq \mathbf{x}$, a prefix of $\mathbf{x}$ by $w \sqsubseteq_p \mathbf{x}$ and a suffix by $w \sqsubseteq_s \mathbf{x}$. Alternatively, a subsequence w of $\mathbf{x}$ ranging from position i to j is referred to as $\mathbf{x}[i..j]$.

3.1 Embedding Sequences using a Formal Language

The basis for embedding of a sequence $\mathbf{x}$ is a formal language L, whose elements are sequences spanning an $|L|$-dimensional feature space. We refer to L as the *embedding language* and to a sequence $w \in L$ as a *word* of L. There exist numerous ways to define L reflecting particular aspects of application domains, yet we focus on three definitions that have been widely used in previous research:

1. **Bag-of-words**. In this model, L corresponds to a set of words from a natural language. L can be either defined explicitly by providing a dictionary or implicitly by partitioning sequences according to a set of delimiter symbols $D \subset \mathcal{A}$ (e.g., Salton, 1979; Joachims, 2002).

$$L = \text{Dictionary (explicit)}, \quad L = (\mathcal{A} \setminus D)^* \text{ (implicit)}.$$

2. **K-grams** and **blended k-grams**. For the case of k-grams (in bioinformatics often referred to as k-mers), L is the set of all sequences of length k (e.g., Damashek, 1995; Leslie et al., 2002). The model of k-grams can further be "blended" by considering all sequences from length j up to k (e.g., Shawe-Taylor and Cristianini, 2004).

$$L = \mathcal{A}^k \text{ (k-grams)}, \quad L = \bigcup_{i=j}^{k} \mathcal{A}^i \text{ (blended k-grams)}.$$

3. **Contiguous sequences**. In the most general case, L corresponds to the set of all contiguous sequences or alternatively to blended k-grams with infinite k (e.g., Vishwanathan and Smola, 2004; Rieck et al., 2007).

$$L = \mathcal{A}^* \quad \text{or} \quad L = \bigcup_{i=1}^{\infty} \mathcal{A}^i.$$

Note that the alphabet $\mathcal{A}$ in the embedding languages may also be composed of higher semantic constructs, such as natural words or syntactic tokens. In these cases a k-gram corresponds to k consecutive words or tokens, and a bag-of-words models could represent textual clauses or phrases.

Given an embedding language L, a sequence $\mathbf{x}$ can be mapped into the $|L|$-dimensional feature space by calculating a function $\phi_w(\mathbf{x})$ for every $w \in L$ appearing in $\mathbf{x}$. The embedding function Φ for a sequence $\mathbf{x}$ is given by

$$\Phi : \mathbf{x} \mapsto (\phi_w(\mathbf{x}))_{w \in L} \text{ with } \phi_w(\mathbf{x}) := \text{occ}(w, \mathbf{x}) \cdot \mathcal{W}_w \tag{1}$$

where $\text{occ}(w, \mathbf{x})$ is the number of occurrences of w in the sequence $\mathbf{x}$ and $\mathcal{W}_w$ a weighting assigned to individual words. Alternatively $\text{occ}(w, \mathbf{x})$ may be defined as frequency, probability or binary flag for the occurrences of w in $\mathbf{x}$.

While the choice and design of an embedding language L offer a large degree of flexibility, it is often necessary to refine the amount of contribution for each word $w \in L$, for example it is a common practice in text processing to ignore stop words and terms that do not carry semantic content. In the embedding function (1) such refinement is realized by the weighting term $\mathcal{W}_w$. The following three weighting schemes for defining $\mathcal{W}_w$ have been proposed in previous research:

1. **Corpus dependent weighting.** The weight $\mathcal{W}_w$ is based on the occurrences of w in the corpus of sequences (see Salton et al., 1975). Most notable is the inverse document frequency (IDF) weighting, in which $\mathcal{W}_w$ is defined over the number of documents N and the frequency $d(w)$ of w in the corpus.
$$\mathcal{W}_w = \log_2 N - \log_2 d(w) + 1.$$
If $\mathrm{occ}(w, \mathbf{x})$ is the frequency of w in $\mathbf{x}$, the embedding function (1) corresponds to the well-known term frequency and inverse document frequency (TF-IDF) weighting scheme.

2. **Length dependent weighting.** The weight $\mathcal{W}_w$ is based on the length $|w|$ (see Shawe-Taylor and Cristianini, 2004; Vishwanathan and Smola, 2004), for example, so that longer words contribute more than shorter words to a similarity measure. A common approach is defining $\mathcal{W}_w$ using a decay factor $0 \leq \lambda \leq 1$.
$$\mathcal{W}_w = \lambda^{-|w|}.$$

3. **Position dependent weighting.** The weight $\mathcal{W}_w$ is based on the position of w in $\mathbf{x}$. Vishwanathan and Smola (2004) propose a direct weighting scheme, in which $\mathcal{W}_w$ is defined over positional weights $\mathcal{W}(k, \mathbf{x})$ for each position k in $\mathbf{x}$ as
$$\mathcal{W}_w = \mathcal{W}_{\mathbf{x}[i..j]} = \sum_{k=i}^{j} \mathcal{W}(k, \mathbf{x}).$$
An indirect approach to position dependent weighting can be implemented by extending the alphabet $\mathcal{A}$ with positional information to $\tilde{\mathcal{A}} = \mathcal{A} \times \mathbb{N}$, so that every element $(a, k) \in \tilde{A}$ of the extended alphabet is a pair of a symbol a and a position k.

The introduced weighting schemes can be coupled to further refine the embedding based on L, for example, in text processing the impact of a particular term might be influenced by the term frequency, inverse document frequency and its length.

3.2 Vectorial Similarity Measures for Sequences

With an embedding language L at hand, we can now express common vectorial similarity measures in the domain of sequences. Table 1 and 2 list well-known kernel and distance functions (see Vapnik, 1995; Schölkopf and Smola, 2002; Webb, 2002) in terms of L. The histogram intersection kernel in Table 1 derives from computer vision (see Swain and Ballard, 1991; Odone et al., 2005) and the Jensen-Shannon divergence in Table 2 is defined using $H(x, y) = x \log \frac{2x}{x+y} + y \log \frac{2y}{x+y}$.

A further and rather exotic class of vectorial similarity measures are *similarity coefficients* (see Sokal and Sneath, 1963; Anderberg, 1973). These coefficients have been designed for comparison of binary vectors and often express non-metric properties. They are constructed using three summation variables a, b and c, which reflect the number of matching components $(1/1)$, left mismatching

Kernel	$k(\mathbf{x},\mathbf{y})$
Linear	$\sum_{w\in L}\phi_w(\mathbf{x})\phi_w(\mathbf{y})$
Polynomial	$\left(\sum_{w\in L}\phi_w(\mathbf{x})\phi_w(\mathbf{y})+\theta\right)^p$
Sigmoidal	$\tanh\left(\sum_{w\in L}\phi_w(\mathbf{x})\phi_w(\mathbf{y})+\theta\right)$
Gaussian	$\exp\left(\frac{-d(\mathbf{x},\mathbf{y})^2}{2\sigma^2}\right)$
Histogram intersection	$\sum_{w\in L}\min(\phi_w(\mathbf{x}),\phi_w(\mathbf{y}))$

Table 1: Kernel functions for sequential data.

Distance	$d(\mathbf{x},\mathbf{y})$	**Distance**	$d(\mathbf{x},\mathbf{y})$
Manhattan	$\sum_{w\in L}\lvert\phi_w(\mathbf{x})-\phi_w(\mathbf{y})\rvert$	Chebyshev	$\max_{w\in L}\lvert\phi_w(\mathbf{x})-\phi_w(\mathbf{y})\rvert$
χ^2 distance	$\sum_{w\in L}\frac{(\phi_w(\mathbf{x})-\phi_w(\mathbf{y}))^2}{\phi_w(\mathbf{x})+\phi_w(\mathbf{y})}$	Geodesic	$\arccos\sum_{w\in L}\phi_w(\mathbf{x})\phi_w(\mathbf{y})$
Canberra	$\sum_{w\in L}\frac{\lvert\phi_w(\mathbf{x})-\phi_w(\mathbf{y})\rvert}{\phi_w(\mathbf{x})+\phi_w(\mathbf{y})}$	Hellinger^2	$\sum_{w\in_L}(\sqrt{\phi_w(\mathbf{x})}-\sqrt{\phi_w(\mathbf{y})})^2$
Minkowski^p	$\sum_{w\in L}\lvert\phi_w(\mathbf{x})-\phi_w(\mathbf{y})\rvert^p$	Jensen-Shannon	$\sum_{w\in L}H(\phi_w(\mathbf{x}),\phi_w(\mathbf{y}))$

Table 2: Distance functions for sequential data.

components $(0/1)$ and right mismatching components $(1/0)$ in two binary vectors. Common similarity coefficients are given in Table 3. For application to non-binary vectors the summation variables a,b,c can be extended in terms of an embedding language L (Rieck et al., 2006):

$$
\begin{aligned}
a &= \sum_{w\in L}\min(\phi_w(\mathbf{x}),\phi_w(\mathbf{y})),\\
b &= \sum_{w\in L}[\phi_w(\mathbf{x})-\min(\phi_w(\mathbf{x}),\phi_w(\mathbf{y}))],\\
c &= \sum_{w\in L}[\phi_w(\mathbf{y})-\min(\phi_w(\mathbf{x}),\phi_w(\mathbf{y}))].
\end{aligned}
$$

The above definition of a matches the histogram intersection kernel k provided in Table 1, so that alternatively all summation variables can be expressed by

$$
a = k(\mathbf{x},\mathbf{y}),\ b = k(\mathbf{x},\mathbf{x})-k(\mathbf{x},\mathbf{y}),\ c = k(\mathbf{y},\mathbf{y})-k(\mathbf{x},\mathbf{y}). \tag{2}
$$

Sim. Coeff.	$s(\mathbf{x},\mathbf{y})$	**Sim. Coeff.**	$s(\mathbf{x},\mathbf{y})$
Simpson	$a/\min(a+b,a+c)$	Kulczynski (1)	$a/(b+c)$
Jaccard	$a/(a+b+c)$	Kulczynski (2)	$\frac{1}{2}(a/(a+b)+a/(a+c))$
Braun-Blanquet	$a/\max(a+b,a+c)$	Otsuka, Ochiai	$a/\sqrt{(a+b)(a+c)}$
Czekanowski, Sørensen-Dice	$2a/(2a+b+c)$	Sokal-Sneath, Anderberg	$a/(a+2(b+c))$

Table 3: Similarity coefficients for sequential data

Hence, one can consider the similarity coefficients given in Table 3 as variations of the histogram intersection kernel, for example, the Jaccard coefficient can be formulated solely in terms of k:

$$s(\mathbf{x},\mathbf{y}) = \frac{a}{a+b+c} = \frac{k(\mathbf{x},\mathbf{y})}{k(\mathbf{x},\mathbf{x})+k(\mathbf{y},\mathbf{y})-k(\mathbf{x},\mathbf{y})}.$$

3.3 A Generic Framework for Similarity Measures

All of the similarity measures introduced in the previous section share a similar mathematical construction: an inner component-wise function is aggregated over each dimension using an outer operator, for example, the linear kernel is defined as the sum of component-wise products and the Chebyshev distance as the maximum of component-wise absolute differences.

One can exploit this shared structure to derive a unified formulation for similarity measures (Rieck et al., 2006, 2007), consisting of an inner function m and an outer operator $\oplus$ as follows

$$s(\mathbf{x},\mathbf{y}) = \bigoplus_{w\in L} m(\phi_w(\mathbf{x}),\phi_w(\mathbf{y})). \tag{3}$$

For convenience in later design of algorithms, we introduce a "multiplication" operator $\otimes$ which corresponds to executing the $\oplus$ operation k times. Thus, for any $n \in \mathbb{N}$ and $x \in \mathbb{R}$, we define $\otimes$ as

$$x \otimes n := \underbrace{x \oplus \ldots \oplus x}_{n}.$$

Given the unified form (3), kernel and distance functions presented in Table 1 and 2 can be re-formulated in terms of $\oplus$ and m. Adaptation of similarity coefficients to the unified form (3) involves a re-formulation of the summation variables a, b and c. The particular definitions of outer and inner functions for the presented similarity measures are given in Table 4. The polynomial and sigmoidal kernels as well as the Geodesic distance are not shown since they can be expressed using a linear kernel. For the Chebyshev distance the operator $\otimes$ represents the identity function, while for all other similarity measures it represents a multiplication.

Kernel	$\oplus$	$m(x,y)$
Linear	$+$	$x \cdot y$
Histogram inters.	$+$	$\min(x,y)$

Sim. Coef.	$\oplus$	$m(x,y)$
Variable a	$+$	$\min(x,y)$
Variable b	$+$	$x - \min(x,y)$
Variable c	$+$	$y - \min(x,y)$

Distance	$\oplus$	$m(x,y)$
Manhattan	$+$	$\lvert x-y\rvert$
χ^2 distance	$+$	$(x-y)^2/(x+y)$
Canberra	$+$	$\lvert x-y\rvert/(x+y)$
Minkowski^p	$+$	$\lvert x-y\rvert^p$
Chebyshev	max	$\lvert x-y\rvert$
Hellinger^2	$+$	$(\sqrt{x}-\sqrt{y})^2$
Jensen-Shannon	$+$	$H(x,y)$

Table 4: Unified formulation of similarity measures.

As a last step towards the development of comparison algorithms, we need to address the high dimensionality of the feature space induced by the embedding language L. The unified form (3) theoretically involves computation of m over all $w \in L$, which is practically infeasible for most L. Yet the feature space induced by L is sparse, since a sequence $\mathbf{x}$ comprises only a limited number of contiguous subsequences—at most $(|\mathbf{x}|^2+|\mathbf{x}|)/2$ subsequences. As a consequence of the sparseness

only very few terms $\phi_w(\mathbf{x})$ and $\phi_w(\mathbf{y})$ in the unified form (3) have non-zero values. We exploit this fact by defining $m(0,0) = \mathbf{e}$, where $\mathbf{e}$ is the neutral element for the operator $\oplus$, so that for any $x \in \mathbb{R}$ holds

$$x \oplus \mathbf{e} = x, \quad \mathbf{e} \oplus x = x.$$

By assigning $m(0,0)$ to $\mathbf{e}$, the computation of a similarity measure can be reduced to cases where either $\phi_w(\mathbf{x}) > 0$ or $\phi_w(\mathbf{y}) > 0$, as the term $m(0,0)$ does not affect the result of expression (3). We can now refine the unified form (3) by partitioning the similarity measures into *conjunctive* and *disjunctive* measures using an auxiliary function $\tilde{m}$:

$$s(\mathbf{x}, \mathbf{y}) = \bigoplus_{w \in L} \tilde{m}(\phi_w(\mathbf{x}), \phi_w(\mathbf{y})).$$

1. **Conjunctive similarity measures.** The inner function m only accounts pairwise non-zero components, so that for any $x \in \mathbb{R}$ holds $m(x,0) = \mathbf{e}$ and $m(0,x) = \mathbf{e}$.

   $$\tilde{m}(x,y) = \begin{cases} m(x,y) & \text{if } x > 0 \textbf{ and } y > 0 \\ \mathbf{e} & \text{otherwise.} \end{cases}$$

   Kernel functions fall into this class, except for the distance-based RBF kernel. By using a kernel to express similarity coefficients as shown in expression (2), similarity coefficients also exhibit the conjunctive property.

2. **Disjunctive similarity measures.** The inner function m requires at least one component to be non-zero, otherwise $m(0,0) = \mathbf{e}$ holds.

   $$\tilde{m}(x,y) = \begin{cases} m(x,y) & \text{if } x > 0 \textbf{ or } y > 0 \\ \mathbf{e} & \text{otherwise.} \end{cases}$$

   Except for the Geodesic distance, all of the presented distances fall into this class. Depending on the embedding language, this class is computational more expensive than conjunctive measures.

As a result, any similarity measure, including those in Table 1, 2 and 3, composed of an inner and outer function can be applied for efficient comparison of embedded sequences, if (a) a neutral element $\mathbf{e}$ for the outer function $\oplus$ exists and (b) the inner function m is either conjunctive or disjunctive, that is at least $m(0,0) = \mathbf{e}$ holds.

4. Algorithms and Data Structures

We now introduce data structures and algorithms for efficient computation of the proposed similarity measures. In particular, we present three approaches differing in capabilities and implementation complexity covering simple sorted arrays, tries and generalized suffix trees. For each approach, we briefly present the involved data structure, provide a discussion of the comparison algorithm and give run-time bounds for extraction and comparison of embedded sequences. Additionally, we provide implementation details that improve run-time performance in practice.

As an example running through this section we consider the two sequences $\mathbf{x} = abbaa$ and $\mathbf{y} = baaaab$ from the binary alphabet $\mathcal{A} = \{a, b\}$ and the embedding language of 3-grams, $L = \mathcal{A}^3$. For a data structure storing multiple words $w \in L$ of possibly different lengths, we denote by k the length of longest words.

4.1 Sorted Arrays

Data structure. A simple and intuitive representation for storage of embedded sequences are *sorted arrays* or alternatively sorted lists (Joachims, 2002; Rieck et al., 2006; Sonnenburg et al., 2007). Given an embedding language L and a sequence $\mathbf{x}$, all words $w \in L$ satisfying $w \sqsubseteq \mathbf{x}$ are maintained in an array X along with their embedding values $\phi_w(\mathbf{x})$. Each field x of X consists of two attributes: the stored word $word[x]$ and its embedding value $phi[x]$. The length of an array X is denoted by $|X|$. In order to support efficient comparison, the fields of X are sorted by contained words, for example, using the lexicographical order of the alphabet $\mathcal{A}$. Figure 1 illustrates the sorted arrays of 3-grams extracted from the two example sequences $\mathbf{x}$ and $\mathbf{y}$.

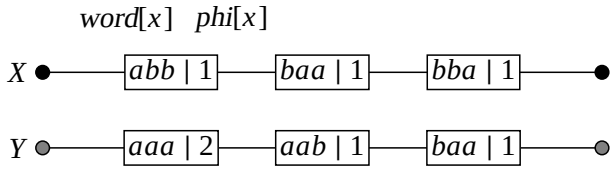


Figure 1: Sorted arrays of 3-grams for $\mathbf{x} = abbaa$ and $\mathbf{y} = baaaab$. The number in each field indicates the number of occurrences.

Algorithm. Comparison of two sorted arrays X and Y is carried out by looping over the fields of both arrays in the manner of merging sorted arrays (Knuth, 1973). During each iteration the inner function m is computed over contained words and aggregated using the operator $\oplus$. The corresponding comparison procedure in pseudo-code is given in Algorithm 1. Herein, we denote the case where a word w is present in $\mathbf{x}$ and $\mathbf{y}$ as *match* and the case of w being contained in either $\mathbf{x}$ or $\mathbf{y}$ as *mismatch*. For run-time improvement, these mismatches can be ignored in Algorithm 1 if a conjunctive similarity measure is computed (cf. Section 3.3).

Algorithm 1 Array-based sequence comparison

```
1:  function COMPARE(X, Y : Array) : ℝ
2:      s ← e, i ← 1, j ← 1
3:      while i ≤ |X| or j ≤ |Y| do
4:          x ← X[i], y ← Y[j]
5:          if y = NIL or word[x] < word[y] then                ▷ Mismatch at x
6:              s ← s ⊕ m(phi[x], 0)
7:              i ← i + 1
8:          else if x = NIL or word[x] > word[y] then           ▷ Mismatch at y
9:              s ← s ⊕ m(0, phi[y])
10:             j ← j + 1
11:         else                                                ▷ Match at x and y
12:             s ← s ⊕ m(phi[x], phi[y])
13:             i ← i + 1, j ← j + 1
14:     return s
```

Run-time. The comparison algorithm based on sorted arrays is simple to implement, yet it does not enable linear-time comparison for all embedding languages, for example, if $L = \mathcal{A}^*$. However, sorted arrays enable linear-time similarity measures, if there exist $O(|\mathbf{x}|)$ words with $w \sqsubseteq \mathbf{x}$, which implies all $w \in L$ have no or constant overlap in $\mathbf{x}$. Examples are the common bag-of-words and k-gram embedding languages.

Under these constraints a sorted array is extracted from a sequence $\mathbf{x}$ in $O(k|\mathbf{x}|)$ time using linear-time sorting, for example, radix sort (Knuth, 1973), where k is the maximum length of all words $w \in L$ in $\mathbf{x}$. The comparison algorithm requires at most $|\mathbf{x}| + |\mathbf{y}|$ iterations, so that the worst-case run-time is $O(k(|\mathbf{x}| + |\mathbf{y}|))$. For extraction and comparison the run-time complexity is linear in the sequence lengths due to the constraint on constant overlap of words, which implies $k|\mathbf{x}| \in O(|\mathbf{x}|)$ for any k and $\mathbf{x}$.

Implementation notes. The simple design of the sorted array approach enables a very efficient extension. If we consider a CPU with registers of b bits, we restrict the maximum word length k, so that every word fits into a CPU register. This restriction enables storage and comparison operations to be performed directly on the CPU, that is operations on words w with $|w| \leq k$ are executed in $O(1)$ time. Depending on the size of the alphabet $|\mathcal{A}|$ and the CPU bits b, the maximum word length is $\lfloor b / \log_2 |\mathcal{A}| \rfloor$. In many practical applications one can strongly benefit from this extensions, as k is either bounded anyway, for example, for k-grams, or longer words are particular rare and do not increase accuracy significantly. For example on current 64 bit CPU architectures one can restrict k to 32 for DNA sequences with $|\mathcal{A}| = 4$ and to $k = 10$ for textual documents with $|\mathcal{A}| \leq 64$. Alternatively, embedded words may also be represented using hash values of b bits, which enables considering words of arbitrary length, but introduces a probability for hash collisions (Knuth, 1973).

Another extension for computation of conjunctive measures using sorted arrays has been proposed by Sonnenburg et al. (2007). If two sequences $\mathbf{x}$ and $\mathbf{y}$ have unbalanced sizes $|\mathbf{x}| \ll |\mathbf{y}|$, one loops over the shorter sorted array X and performs a binary search procedure on Y, in favor of processing both sorted arrays in parallel. The worst-case run-time for this comparison is $O(k(|\mathbf{x}| \log_2 |\mathbf{y}|))$, so that one may automatically apply this extension if for two sequences $\mathbf{x}$ and $\mathbf{y}$ holds $|\mathbf{x}| \log_2 |\mathbf{y}| < |\mathbf{x}| + |\mathbf{y}|$.

4.2 Tries

Data structure. A *trie* is a tree structure for storage and retrieval of sequences. The edges of a trie are labeled with symbols of $\mathcal{A}$ (Fredkin, 1960; Knuth, 1973). A path from the root to a marked node x represents a stored sequence, hereafter denoted by $\bar{x}$. A trie node x contains a vector of size $|\mathcal{A}|$ linking to child nodes, a binary flag to indicate the end of a sequence $\mathit{mark}[x]$ and an embedding value $\mathit{phi}[x]$.[1] The i-th child node representing the i-th symbol in $\mathcal{A}$ is accessed via $\mathit{child}[x, i]$. If the i-th child is not present $\mathit{child}[x, i] = \text{NIL}$.

A sequences $\mathbf{x}$ is embedded using a trie X by storing all $w \in L$ with $w \sqsubseteq \mathbf{x}$ and corresponding $\phi_w(\mathbf{x})$ in X (Shawe-Taylor and Cristianini, 2004). Figure 2 shows tries of 3-grams for the two example sequences $\mathbf{x}$ and $\mathbf{y}$. Note, that for the embedding language of k-grams considered in Figure 2 all marked nodes are leaves, while for other embedding languages they may correspond to inner nodes, for example, for the case of blended k-grams, where every node in a trie marks the end of a sequence.

1. For convenience, we assume that child nodes are maintained in a vector, while in practice sorted lists, balanced trees or hash tables may be preferred.

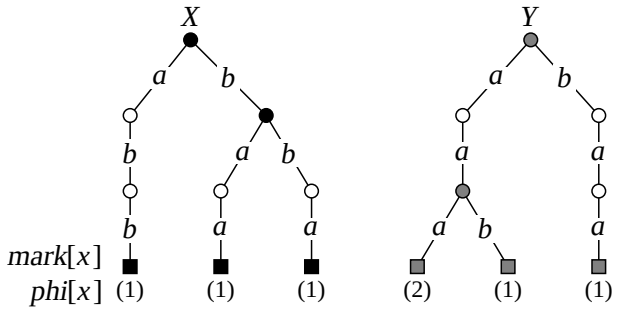


Figure 2: Tries of 3-grams for $\mathbf{x} = abbaa$ and $\mathbf{y} = baaaab$. The number in brackets at leaves indicate the number of occurrences. Marked nodes are squared. White nodes are implicit and not maintained in a compact trie representation.

Algorithm. Comparison of two tries is performed as in Algorithm 2: Starting at the root nodes, one recursively traverses both tries in parallel. If the traversal passes at least one marked node the inner function m is computed as either a matching or mismatching word occurred (Rieck et al., 2006). To simplify presentation of the algorithm, we assume that $mark[\text{NIL}]$ returns false and $child[\text{NIL}, i]$ returns NIL.

Algorithm 2 Trie-based sequence comparison

```
1:  function COMPARE(X, Y : Trie) : ℝ
2:      s ← e
3:      if X = NIL and Y = NIL then                          ▷ Base case
4:          return s
5:      for i ← 1, |A| do
6:          x ← child[X, i],  y ← child[Y, i]
7:          if mark[x] and not mark[y] then                  ▷ Mismatch at x
8:              s ← s ⊕ m(phi[x], 0)
9:          if not mark[x] and mark[y] then                  ▷ Mismatch at y
10:             s ← s ⊕ m(0, phi[y])
11:         if mark[x] and mark[y] then                      ▷ Match at x and y
12:             s ← s ⊕ m(phi[x], phi[y])
13:         s ← s ⊕ COMPARE(x, y)
14:     return s
```

Run-time. The trie-based approach enables linear-time similarity measures over a larger set of formal languages than sorted arrays. For tries we require all $w \in L$ with $w \sqsubseteq \mathbf{x}$ to have either constant overlap in $\mathbf{x}$ or to be prefix of another word, for example, as for the blended k-gram embedding languages.

To determine the run-time complexity on tries, we need to consider the following property: A trie storing n words of maximum length k has depth k and at most kn nodes. Thus, a sequence $\mathbf{x}$ containing $O(|\mathbf{x}|)$ words of maximum length k is embedded using a trie in $O(k|\mathbf{x}|)$ run-time. As an

invariant for the comparison procedure, the nodes x and y in Algorithm 2 stay at the same depth in each recursion. Hence, the comparison algorithm visits at most $k|\mathbf{x}| + k|\mathbf{y}|$ nodes, which results in a worst-case run-time of $O(k(|\mathbf{x}| + |\mathbf{y}|))$. The extraction and comparison run-time is linear in the sequence lengths, as we require words to either have constant overlap, which implies $k|\mathbf{x}| \in O(|\mathbf{x}|)$, or to be prefix of another word, which implies that both words share an identical path in the trie.

Implementation notes. The first extension for the trie data structure is aggregation of embedding values in nodes. If in Algorithm 2 a mismatch occurs at node x, the algorithm recursively descends to all child nodes of x. At this point, however, it is clear that all nodes below x will also be mismatches, as all words w with $\bar{x} \sqsubseteq_p w$ are not present in the compared trie. This fact can be exploited by maintaining an aggregated value φ_x at each node x given by

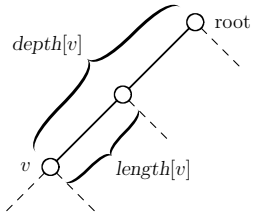
$$\varphi_x := \bigoplus_{w \in I} \phi_w(\mathbf{x}) \;\text{ with }\; I = \{w \in L \mid \bar{x} \sqsubseteq_p w\}.$$

Instead of recursively descending at a mismatching node x, one uses φ_x to retrieve the aggregation of all lower embedding values. The extension allows disjunctive similarity measures to be computed as efficient as conjunctive measures at a worst-case run-time of $O(k \min(|\mathbf{x}|, |\mathbf{y}|))$.

The second extension originates from the close relation of tries and suffix trees. The nodes of a trie can be classified as *implicit* if they link to only one child node and as *explicit* otherwise. By iteratively removing implicit nodes and appending their labels to edges of explicit parent nodes one obtains a *compact trie* (cf. Knuth, 1973; Gusfield, 1997). Edges are labeled by subsequences encoded using indices i and j pointing to $\mathbf{x}[i..j]$. The major benefit of compact tries is reduced space complexity, which decreases from $O(k|\mathbf{x}|)$ to $O(|\mathbf{x}|)$ independent of the maximum length k of stored words.

4.3 Generalized Suffix Trees

Data structure. A *generalized suffix tree* (GST) is a compact trie containing all suffixes of a set of sequences $\mathbf{x}_1, \ldots, \mathbf{x}_l$ (Gusfield, 1997). Every path in a GST from the root to a leaf corresponds to one suffix. A GST is obtained by extending each sequence $\mathbf{x}_i$ with a delimiter $\$_i \notin \mathcal{A}$ and constructing a suffix tree from the concatenation $\mathbf{z} = \mathbf{x}_1\$_1 \ldots \mathbf{x}_l\$_l$.



For each GST node v we denote by *children*$[v]$ the set of child nodes, by *length*$[v]$ the number of symbols on the incoming edge, by *depth*$[v]$ the total number of symbols on the path from the root node to v and by *phi*$[v, i]$ the number of suffixes of $\mathbf{x}_i$ passing through node v. As every subsequence of $\mathbf{x}_i$ is a prefix of some suffix, *phi*$[v, i]$ reflects the occurrences (alternatively frequency or binary flag) for all subsequences terminating on the edge to v. An example of a GST is given in Figure 3. In the remaining part we focus on the case of two sequences $\mathbf{x}$ and $\mathbf{y}$ delimited by $\$_1$ and $\$_2$, computation of similarity measures over a set of sequences being a straightforward extension.

Algorithm. Computation of similarity measures is carried out by traversing a GST in depth-first order (Rieck et al., 2007). An implementation in pseudo-code is given in Algorithm 3. At each node v the inner function m is computed using *phi*$[v, 1]$ and *phi*$[v, 2]$. To account for different words along an edge and to support various embedding languages the function FILTER is employed, which selects appropriate contributions similar to the weighting introduced by Vishwanathan and

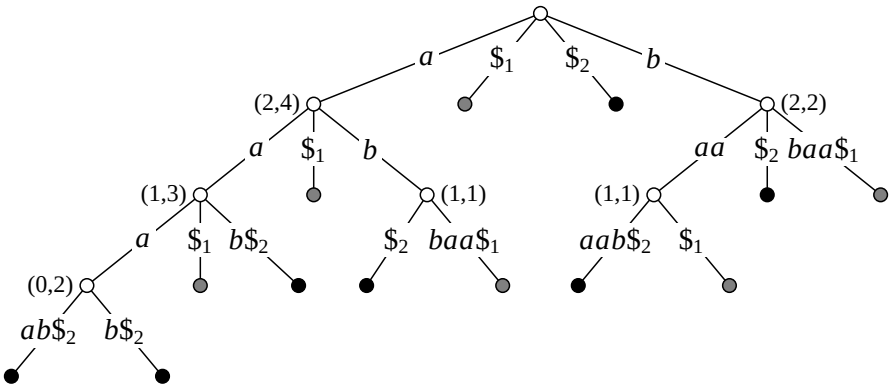


Figure 3: Generalized suffix tree for $\mathbf{x} = abbaa\$_1$ and $\mathbf{y} = baaaab\$_2$. The numbers in brackets at each inner node v correspond to $phi[v,1]$ and $phi[v,2]$. Edges are shown with associated subsequences instead of indices.

Smola (2004). At a node v the function takes $length[v]$ and $depth[v]$ of v as arguments to determine how much the node and its incoming edge contribute to the similarity measure, for example, for the embedding language of k-grams only nodes up to a path depth of k need to be considered.

Algorithm 3 GST-based sequence comparison

```
1:  function COMPARE(X, Y : A*) : R
2:      T ← CONCAT(X, Y)
3:      S ← SUFFIXTREE(T)
4:      return TRAVERSE(root[S])
5:  function TRAVERSE(v : Node) : R
6:      s ← e
7:      for c ← children[v] do
8:          s ← s ⊕ TRAVERSE(c)                ▷ Depth-first traversal
9:      n ← FILTER(length[v], depth[v])        ▷ Filter words on edge to v
10:     s ← s ⊕ m(phi[v,1], phi[v,2]) ⊗ n
11:     return s
```

Algorithm 4 shows a filter function for k-grams. The filter returns 0 for all edges that do not correspond to a k-gram, either because they are too shallow or too deep in the GST, and returns 1 if a k-gram terminates on the edge to a node v.

Algorithm 4 Filter function for k-grams, $L = \mathcal{A}^k$

```
function FILTER(v : Node) : N
    if depth[v] ≥ k and depth[v] − length[v] < k then
        return 1
    return 0
```

Another example of a filter is given in Algorithm 5. The filter implements the embedding language $L = \mathcal{A}^*$. The incoming edge to a node v contributes to a similarity measure by *length*[v], because exactly *length*[v] contiguous subsequences terminate on the edge to v.

Algorithm 5 Filter function for all contiguous subsequences, $L = \mathcal{A}^*$

```
function FILTER(v : Node) : ℕ
    return length[v]
```

The bag-of-words model can be implemented either by encoding each word as a symbol of $\mathcal{A}$ or by augmenting nodes to indicate the presence of delimiter symbols on edges. Further definitions of weighting schemes for string kernels, which are suitable for Algorithm 3, are given by Vishwanathan and Smola (2004).

Run-time. Suffix trees are well-known for their ability to enhance run-time performance of string algorithms (Gusfield, 1997). The advantage exploited herein is that a suffix tree comprises a quadratic amount of information, namely all suffixes, in a linear representation. Thus, a GST enables linear-time computation of similarity measures, even if a sequence $\mathbf{x}$ contains $O(|\mathbf{x}|^2)$ words and the embedding language corresponds to $L = \mathcal{A}^*$.

There are well-known algorithms for linear-time construction of suffix trees (e.g., Weiner, 1973; McCreight, 1976; Ukkonen, 1995), so that a GST for two sequences $\mathbf{x}$ and $\mathbf{y}$ can be constructed in $O(|\mathbf{x}| + |\mathbf{y}|)$ using the concatenation $\mathbf{z} = \mathbf{x}\$_1\mathbf{y}\$_2$. As a GST contains at most $2|\mathbf{z}|$ nodes, the worst-case run-time of any traversal is $O(|\mathbf{z}|) = O(|\mathbf{x}| + |\mathbf{y}|)$. Consequently, computation of similarity measures between sequences using a GST can be realized in time linear in the sequence lengths independent of the complexity of L.

Implementation notes. In practice the GST algorithm may suffer from high memory consumption, due to storage of child nodes and suffix links. To alleviate this problem an alternative data structure with similar properties—*suffix arrays*—was proposed by Manber and Myers (1993). A suffix array is an integer array enumerating the suffixes of a sequence $\mathbf{z}$ in lexicographical order. It can be constructed in linear run-time, however, algorithms with super-linear run-time are surprisingly faster on real-world data (see Manzini and Ferragina, 2004; Maniscalco and Puglisi, 2007).

Abouelhoda et al. (2004) propose a generic procedure for replacing suffix trees with enhanced suffix array, for example, as performed for the string kernel computation of Teo and Vishwanathan (2006). This procedure involves several auxiliary data structures for maintenance of child nodes and suffix links. In our implementation we follow a different approach based on the work of Kasai et al. (2001a) and Kasai et al. (2001b). Using a suffix array and an array of longest-common prefixes (LCP) for suffixes, we replace the traversal of the GST by looping over a generalized suffix array in linear time.

Application of suffix arrays reduces memory requirements by a factor of 4. About $11|\mathbf{z}|$ bytes are required for the modified GST algorithm: 8 bytes for a suffix and inverse suffix array, 2 bytes for sequence indices and on average 1 byte for an LCP array. In comparison, a suffix tree usually requires over $40|\mathbf{z}|$ bytes (Abouelhoda et al., 2004) and the enhanced suffix array of Teo and Vishwanathan (2006) about $19|\mathbf{z}|$ bytes.

5. Experiments and Applications

In order to evaluate the run-time performance of the proposed comparison algorithms in practice and to investigate the effect of different similarity measures on sequential data, we conducted experiments on real world sequences. We chose nine data sets from the domains of bioinformatics, text processing and computer security. Details about each data set, contained sequences and references are given in Table 5.

Name	**Sequence type**	**#**	$\lvert\mathcal{A}\rvert$	$\lvert\mathbf{x}\rvert_\mu$	**Reference**
Bioinformatics					
ARTS	DNA sequences	46794	4	2400	Sonnenburg et al. (2006)
C.Elegans	DNA sequences	10025	4	10000	Wormbase WS120
SPROT	Protein sequences	150807	23	467	O'Donovan et al. (2002)
Text processing					
Reuters	News articles	19042	92	839	Lewis (1997)
Heise	News articles	30146	96	1800	`www.heise.de`
RFC	Text documents	4589	106	49954	`www.rfc-editor.org`
Computer security					
HIDS	System call traces	25979	83	156	Lippmann et al. (2000)
NIDS	Connection payloads	21330	116	1274	Lippmann et al. (2000)
Spam	Emails bodies	33702	176	1539	Metsis et al. (2006)

Table 5: Data sets of sequences. The number of sequences in each set is denoted by #, the alphabet size by $|\mathcal{A}|$ and the average sequence length by $|\mathbf{x}|_\mu$.

5.1 Run-time Experiments

The linear-time algorithms presented in Section 4 build on data structures of increasing complexity and capability—sorted arrays are simple but limited in capabilities, tries are more involved, yet they do not cover all embedding languages and generalized suffix trees are relatively complex and support the full range of embedding languages. In practice, however, it is the absolute and not asymptotic run-time that matters. Since the absolute run-time is affected by hidden constant factors, depending on design and implementation of an algorithm, it can only be evaluated experimentally.

Therefore each algorithm was implemented using enhancements covered in implementation notes. In particular, for Algorithm 1 we incorporated 64-bit variables to realize a sorted 64-bit array, for Algorithm 2 we implemented a compact trie representation and for Algorithm 3 we used generalized suffix arrays in favor of suffix trees. For each of these algorithms we conducted experiments using different embedding languages to assess the run-time on the data sets given in Table 5.

We applied the following experimental procedure and averaged run-time over 10 individual runs: 500 sequences are randomly drawn from a data set and a 500×500 matrix is computed for the Manhattan distance using a chosen embedding language. The run-time of the matrix computation is measured and reported in pairwise comparisons per second. Note, that due to the symmetry of the Manhattan distance only $(n^2+n)/2$ comparisons need to be performed for an $n \times n$ matrix.

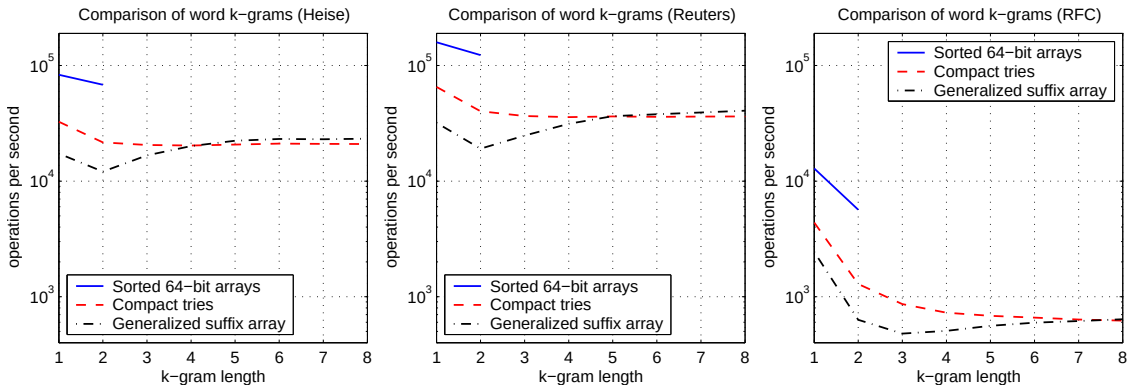


Figure 4: Run-time of sequences comparison over word k-grams for different algorithms. The x-axis gives the word k-gram lengths. The y-axis shows the number of comparison operations per second in log-scale.

Embedding language: bag-of-words. As a first embedding language, we consider the bag-of-words model. Since natural words can be defined only for textual data, we limit the focus of this experiment to text data sets in Table 5. In particular, we use the embedding language of *word k-grams*—covering the classic "bag of words" as word 1-grams—by using an alphabet of words instead of characters. Each symbol of the alphabet is stored in 32 bits, so that up to 2^{32} different words can be represented. Experiments are conducted for values of k ranging from 1 to 8.

Figure 4 shows the run-time performance of the implemented algorithms as a function of k on the Reuters, Heise and RFC data sets. The sorted array approach significantly outperforms the other algorithms on all data sets, yet it can only be applied for $k \leq 2$, as it is limited to 64 bits. For small values of k suffix arrays require more time for each comparison compared to compact tries, while for $k > 5$ their performance is similar to compact tries. This difference is explained by the number of unique k-grams $\nu_{\mathbf{x}}$ in each sequence $\mathbf{x}$. For small values of k often holds $\nu_{\mathbf{x}} < |\mathbf{x}|$, so that a trie comparison requires $O(k(\nu_{\mathbf{x}} + \nu_{\mathbf{y}}))$ time in contrast to $O(|\mathbf{x}| + |\mathbf{y}|)$ for a suffix array. The worse run-time performance on the RFC data set is due to longer sequences.

Embedding language: k-grams. For this experiment we focus on the embedding language of k-grams, which is not limited to a particular type of sequences, so that experiments were conducted for all data sets in Table 5. In contrast to the previous setup, k-grams are associated with the original alphabet of each data set: DNA bases and proteins for bioinformatics, characters for texts, and system calls and bytes for computer security. For each data set the value of k is varied from 1 to 19.

The run-time as a function of k for each data set and algorithm is given in Figure 5. The sorted array approach again yields the best performance on all data sets. Moreover, the limitation of sorted arrays to 64 bits does not effect all data sets, so that for DNA all k-gram lengths can be computed. The suffix array slightly outperforms the trie comparison for larger value of k, as its worst-case run-time is independent of the length of k-grams. Absolute performance in terms of the number of comparisons per second differs among data sets due to different average sequence lengths. For data sets with short sequences (e.g., HIDS, ARTS) performance rates up to 10^6 comparisons per second

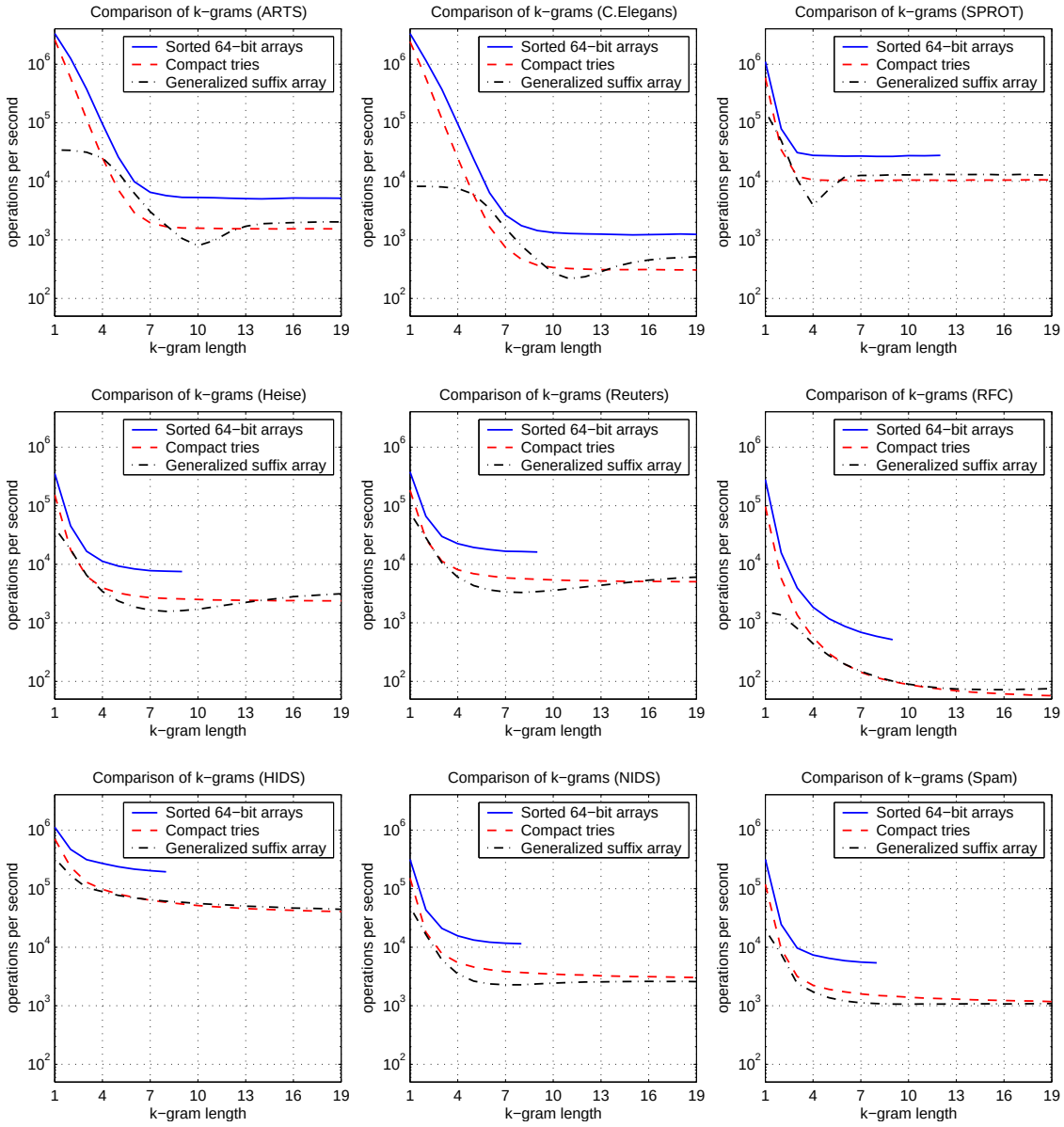


Figure 5: Run-time of sequences comparison over k-grams for different algorithms. The x-axis gives the k-gram lengths. The y-axis shows the number of comparison operations per second in log-scale.

are attainable, while for data sets with longer sequences (e.g., Spam, RFC) generally up to $10^3 - 10^4$ comparisons per second are achievable at best.

5.2 Applications

We now demonstrate that the ability of our approach to compute diverse similarity measures is beneficial in real applications, especially in unsupervised learning scenarios. Our experiments are performed for: (a) categorization of news articles, (b) intrusion detection in network traffic (c) transcription start site recognition in DNA sequences.

Unsupervised text categorization. For this experiment news articles from the topics "Coffee", "Interest", "Sugar" and "Trade" in the Reuters data set are selected. The learning task is to categorize these articles using clustering, without any prior knowledge of labels. As preprocessing we remove all stop words and words that occur in single articles only. We then compute dissimilarity matrices for the Euclidean, Manhattan and Jensen-Shannon distances using the bag-of-words embedding language as discussed in Section 3. The embedded articles are finally assigned to four clusters using complete linkage clustering (see Duda et al., 2001).

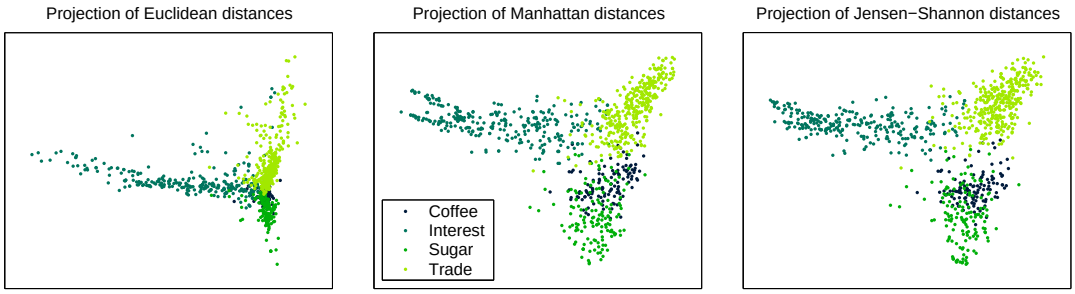
Figure 6(a) shows projections of the embedded articles obtained from the dissimilarity matrices using multidimensional scaling (see Duda et al., 2001). Although projections are limited in describing high-dimensional data, they visualize structure and, thus, help to interpret clustering results. For example, the projection of the Euclidean distances in Figure 6(a) noticeably differs in shape compared to the Manhattan and Jensen-Shannon distances.

The cluster assignments are presented in Figure 6(b) and the distribution of topic labels among clusters is given in Figure 6(c). For the Euclidean distance the clustering fails to unveil features discriminative for article topics, as the majority of articles is assigned to a single cluster. In comparison, the Manhattan and Jensen-Shannon distance allow categorization of the topics "Coffee" and "Sugar", due to observed high frequencies of respective words in articles. However, the Manhattan distance does no allow discrimination of the other two topics, as both are mixed among two clusters. The Jensen-Shannon distance enables better separation of all four topics. The topics "Coffee" and "Sugar" are almost perfectly assigned to clusters and the topics "Interest" and "Trade" each constitute the majority in a respective cluster.

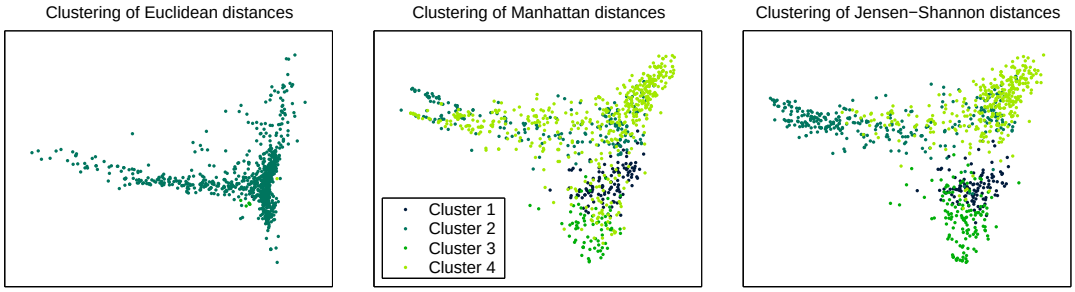
Network intrusion detection. Network intrusion detection aims to automatically identify hacker attacks in network traffic. As labels for such data are hard to obtain in practice, unsupervised learning has gained attention in intrusion detection research. The NIDS data set used for the runtime experiments (cf. Table 5) is known to contain major artifacts (see McHugh, 2000). In order to provide a fair investigation of the impact of various similarity measures on detection of attacks, we generated a smaller private data set. Normal traffic was recorded from the members of our laboratory by providing a virtual network. Additionally attacks were injected by a security expert.

For this experiment we pursue an unsupervised learning approach to network intrusion detection (Rieck and Laskov, 2007). The incoming byte sequences of network connections are embedded using the language of 5-grams, and Zeta, an unsupervised anomaly detection method based on k-nearest neighbors, is applied over the following similarity measures: the Euclidean, Manhattan and Jensen-Shannon distance and the second Kulczynski coefficient (see Section 3.2).

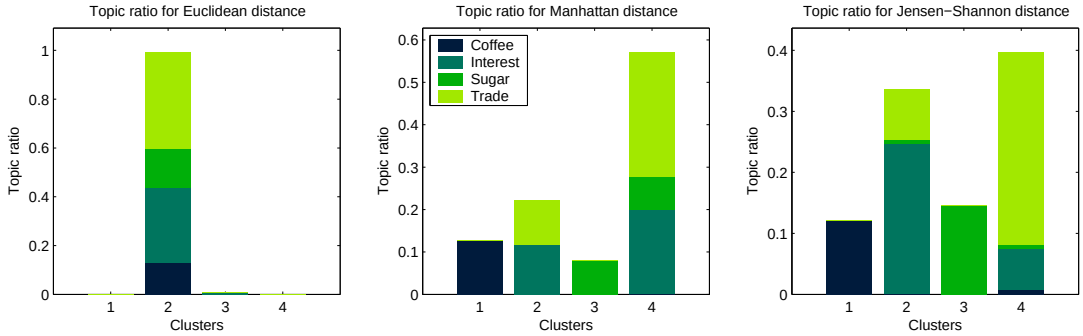
ROC curves for the detection of attacks in the network protocols HTTP, FTP and SMTP are shown in Figure 7. Application of the Jensen-Shannon distance and second Kulczynski coefficient



(a) Projection of embedded articles with true topic label assignments



(b) Projection of embedded articles with cluster assignments



(c) Ratio of topic labels in clusters of embedded articles

Figure 6: Clustering of Reuters articles using different similarity measures (bag-of-words).

yield the highest detection accuracy. Over 78% of all attacks are identified with no false-positives in an unsupervised setup. In comparison, the Euclidean and Manhattan distance give significantly lower detection rates on the FTP and SMTP protocols. The poor detection performance of the latter two similarity measures emphasizes that choosing a discriminative similarity measure is crucial for achieving high accuracy in a particular application.

Transcription start site recognition. The last application focuses on recognition of transcription start sites (TSS), which mark the beginning of genes in DNA. We consider the ARTS data set, which comprises human DNA sequences that either cover the TSS of protein coding genes or have been extracted randomly from the interior of genes. Following the approach of Sonnenburg et al. (2006) these sequences are embedded using the language of 6-grams and a support vector machine (SVM) and a bagged k-nearest neighbor classifier are trained and evaluated on the different partitions of

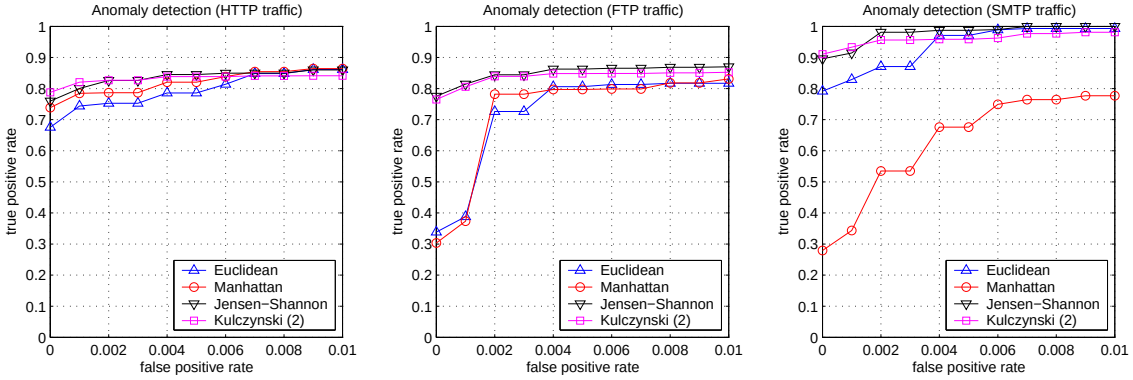


Figure 7: ROC curves for unsupervised anomaly detection on 5-grams of network connections and attacks recorded at our laboratory.

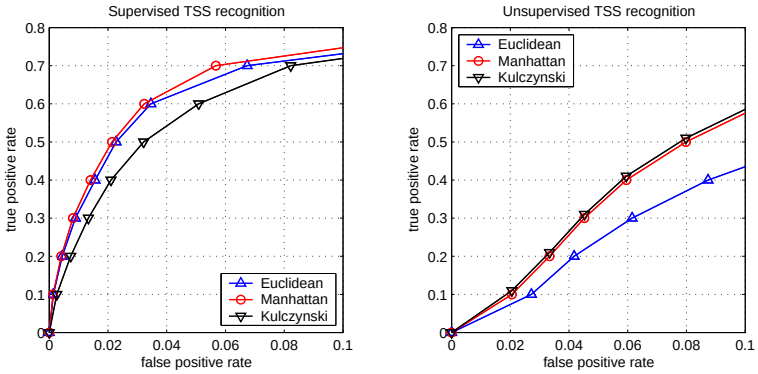


Figure 8: ROC curves for supervised and unsupervised recognition of transcription start sites (TSS) on 6-grams of DNA sequences (ARTS data set).

the data set. Each method is evaluated for the Euclidean distance, the Manhattan distance and the second Kulczynski coefficient. As only 10% of the sequences in the data set correspond to transcription start sites, we additionally apply the unsupervised outlier detection method Gamma (Harmeling et al., 2006), which is similar to the method employed in the previous experiment.

Figure 8 shows the performance achieved by the bagged k-nearest neighbor classifier and the unsupervised learning method.[2] The accuracy in both setups strongly depends on the chosen similarity measures. The metric distances yield better accuracy in the supervised setup. The second Kulczynski coefficient and also the Manhattan distance perform significantly better than the Euclidean distance in unsupervised application. In the absence of labels these measures express better discriminative properties for TSS recognition, that are difficult to access through Euclidean distances. For the supervised application, the classification performance is limited for all similarity

2. Results for the SVM are similar to the bagged k-nearest neighbor classifier and have been omitted.

measures, as only some discriminative features for TSS recognition are encapsulated in k-gram models (cf. Sonnenburg et al., 2006).

6. Conclusions

The framework for comparison of sequences proposed in this article provides means for efficient computation of a large variety of similarity measures, including kernels, distances and non-metric similarity coefficients. The framework is based on embedding of sequences in a high-dimensional feature space using formal languages, such as k-grams, contiguous subsequences, etc. Three implementations of the proposed framework using different data structures have been discussed and experimentally evaluated.

Although all three data structures that were considered—sorted arrays, tries and generalized suffix trees—have asymptotically linear run-time, significant differences in the absolute run-time have been observed in our experiments. The constant factors are affected by various design issues illustrated by our remarks on implementation of the proposed algorithms. In general, we have observed a consistent trade-off between run-time efficiency and complexity of embedding languages a particular data structure can handle. Sorted arrays are the most efficient data structure; however, their applicability is limited to k-grams and bag-of-words models. On the other end of the spectrum are generalized suffix trees (and their more space-efficient implementation using suffix arrays) that can handle unrestricted embedding languages—at a cost of more complicated algorithms and lower efficiency. The optimal data structure for computation of similarity measures thus depends on the embedding language to be used in a particular application.

The proposed framework offers machine learning researchers an opportunity to use a large variety of similarity measures for applications that involve sequential data. Although an optimal similarity measure—as it is well known and has been also observed in our experiments—depends on a particular application, the technical means for seamless incorporation of various similarity measures can be of great utility in practical applications of machine learning. Especially appealing is the possibility for efficient computation of non-Euclidean distances over embedded sequences, which extend applicable similarity measures for sequences beyond inner-products and kernel functions.

In general, the proposed framework demonstrates an important advantage of abstracting data representation—in the form of pairwise relationships—from learning algorithms, which will hopefully motivate further investigation of learning algorithms using a general form of such abstraction.

Acknowledgments

The authors gratefully acknowledge the funding from *Bundesministerium für Bildung und Forschung* under the project MIND (FKZ 01-SC40A) and REMIND (FKZ 01-IS07007A). The authors would like to thank Klaus-Robert Müller, Sören Sonnenburg, Mikio Braun and Vojtech Franc for fruitful discussions and support.

References

M. Abouelhoda, S. Kurtz, and E. Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86, 2004.

M. Anderberg. *Cluster Analysis for Applications*. Academic Press, Inc., New York, NY, USA, 1973.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of COLT*, pages 144–152. ACM Press, 1992.

W. B. Cavnar and J. M. Trenkle. N-gram-based text categorization. In *Proceedings of SDAIR*, pages 161–175, Las Vegas, NV, USA., Apr. 1994.

O. Chapelle, P. Haffner, and V. Vapnik. SVMs for histogram-based image classification. *IEEE Transaction on Neural Networks*, 10(5):1055–1064, 1999.

V. Cherkassky, S. Xuhui, F. Mulier, and V. Vapnik. Model complexity control for regression using vc generalization bounds. *IEEE Transactions on Neural Networks*, 10(5):1075–1089, 1999.

M. Collins and N. Duffy. Convolution kernel for natural language. In *Advances in Neural Information Proccessing Systems*, pages 625–632, 2002.

M. Damashek. Gauging similarity with *n*-grams: Language-independent categorization of text. *Science*, 267(5199):843–848, 1995.

H. Drucker, D. Wu, and V. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

R. Duda, P.E.Hart, and D.G.Stork. *Pattern Classification*. John Wiley & Sons, second edition, 2001.

E. Fredkin. Trie memory. *Communications of ACM*, 3(9):490–499, 1960.

T. Gärtner, J. Lloyd, and P. Flach. Kernels and distances for structured data. *Machine Learning*, 57 (3):205–232, 2004.

T. Graepel, R. Herbrich, P. Bollmann-Sdorra, and K. Obermayer. Classification on pairwise proximity data. In *Advances in Neural Information Processing Systems*, volume 11, 1999.

D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.

B. Haasdonk. Feature space interpretation of svms with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):482–492, 2005.

R. W. Hamming. Error-detecting and error-correcting codes. *Bell System Technical Journal*, 29(2): 147–160, 1950.

S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Müller. Kernel-based nonlinear blind source separation. *Neural Computation*, 15:1089–1124, 2003.

S. Harmeling, G. Dornhege, D. Tax, F. C. Meinecke, and K.-R. Müller. From outliers to prototypes: ordering data. *Neurocomputing*, 69(13–15):1608–1618, 2006.

J. Hopcroft and J. Motwani, R. Ullmann. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2 edition, 2001.

T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7:95–114, 2000.

D. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with nonmetric distances: Image retrieval and class representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (6):583–600, 2000.

T. Joachims. *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of ECML*, pages 137 – 142. Springer, 1998.

T. Kasai, H. Ariumar, and A. Setsuo. Efficient substring traversal with suffix arrays. Technical report, 185, Department of Informatics, Kyushu University, 2001a.

T. Kasai, G. Lee, H. Arimura, S. Arikawa, and K. Park. Linear-time longest-common-prefix computation in suffix arrays and its applications. In *Combinatorial Pattern Matching (CPM), 12th Annual Symposium*, pages 181–192, 2001b.

D. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, 1973.

J. Laub and K.-R. Müller. Feature discovery in non-metric pairwise data. *Journal of Machine Learning*, 5(Jul):801–818, July 2004.

J. Laub, V. Roth, J. Buhmann, and K.-R. Müller. On the information and representation of non-euclidean pairwise data. *Pattern Recognition*, 39(10):1815–1826, Oct. 2006.

C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 5:1435–1455, 2004.

C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proc. Pacific Symp. Biocomputing*, pages 564–575, 2002.

C. Leslie, E. Eskin, A. Cohen, J. Weston, and W. Noble. Mismatch string kernel for discriminative protein classification. *Bioinformatics*, 1(1):1–10, 2003.

V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1966.

D. Lewis. Reuters-21578 text categorization test collection. AT&T Labs Research, 1997.

L. Liao and W. Noble. Combining pairwise sequence similiarity and support vector machines for detecting remote protein evolutionary and structural relationships. *Journal of Computational Biology*, 10:857–868, 2003.

R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, 34(4):579–595, 2000.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

U. Manber and G. Myers. Suffix arrays: a new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948, 1993.

M. Maniscalco and S. Puglisi. An efficient, versatile approach to suffix sorting. *Journal of Experimental Algorithmics*, 12, Article No. 1.2, 2007.

G. Manzini and P. Ferragina. Engineering a lightweight suffix array construction algorithm. *Algorithmica*, 40:33–50, 2004.

W. Masek and M. Patterson. A faster algorithm for computing string edit distance. *Journal of Computer and System sciences*, 20(1):18–31, 1980.

E. M. McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23 (2):262–272, 1976.

J. McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information Systems Security*, 3(4):262–294, 2000.

V. Metsis, G. Androutsopoulos, and G. Paliouras. Spam filtering with naive bayes - which naive bayes? In *Proc. of the 3rd Conference on Email and Anti-Spam (CEAS)*, 2006.

K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Neural Networks*, 12(2):181–201, May 2001.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarties in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

F. Odone, A. Barla, and A. Verri. Building kernels from binary strings for image matching. *IEEE Transactions on Image Processing*, 14(2):169–180, 2005.

C. O'Donovan, M. Martin, A. Gattiker, E. Gasteiger, A. Bairoch, and R. Apweiler. High-quality protein knowledge resource: SWISS-PROT and TrEMBL. *Briefings in Bioinformatics*, 3(3): 275–284, 2002.

C. Ong, X. Mary, S. Canu, and A. Smola. Learning with non-positive kernels. In R. Greiner and D. Schuurmans, editors, *Proceedings of ICML*, pages 639–646. ACM Press, 2004.

G. Rätsch and S. Sonnenburg. Accurate splice site prediction for caenorhabditis elegans. In *Kernel Methods in Computational Biology*, pages 277–298. MIT Press, 2004.

G. Rätsch, S. Sonnenburg, J. Srinivasan, H. Witte, R. Sommer, K.-R. Müller, and B. Schölkopf. Improving the c. elegans genome annotation using machine learning. *PLoS Computational Biology*, 3:e20, 2007.

K. Rieck and P. Laskov. Language models for detection of unknown attacks in network traffic. *Journal in Computer Virology*, 2(4):243–256, 2007.

K. Rieck, P. Laskov, and K.-R. Müller. Efficient algorithms for similarity measures over sequential data: A look beyond kernels. In *Pattern Recognition, Proc. of 28th DAGM Symposium*, LNCS, pages 374–383, Sept. 2006.

K. Rieck, P. Laskov, and S. Sonnenburg. Computation of similarity measures for sequential data using generalized suffix trees. In *Advances in Neural Information Processing Systems 19*, pages 1177–1184, Cambridge, MA, 2007. MIT Press.

V. Roth, J. Laub, M. Kawanabe, and J. Buhmann. Optimal cluster preserving embedding of non-metric proximity data. *IEEE Trans. PAMI*, 25:1540–1551, Dec. 2003.

J. Rousu and J. Shawe-Taylor. Efficient computation of gapped substring kernels for large alphabets. *Journal of Machine Leaning Research*, 6:1323–1344, 2005.

G. Salton. Mathematics and information retrieval. *Journal of Documentation*, 35(1):1–29, 1979.

G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

D. Sankoff and J. Kruskal. *Time wraps, String edits, and Macromulecules: The Theory and Practice of Sequence Comparison*. Addision-Wesley Publishing Co., 1983.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

B. Schölkopf, P. Simard, A. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In *Advances in Neural Information Processing Systems*, volume 10, pages 640–646, Cambridge, MA, 1998a. MIT Press.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998b.

J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

R. Sokal and P. Sneath. *Principles of Numerical Taxonomy*. W.H. Freeman and Company, San Francisco, CA, USA, 1963.

S. Sonnenburg, A. Zien, and G. Rätsch. ARTS: Accurate Recognition of Transcription Starts in Human. *Bioinformatics*, 22(14):e472–e480, 2006.

S. Sonnenburg, G. Rätsch, and K. Rieck. Large scale learning with string kernels. In *Large Scale Kernel Machines*. MIT Press, 2007.

C. Y. Suen. N-gram statistics for natural language understanding and text processing. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1(2):164–172, Apr. 1979.

M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1), 1991.

C. Teo and S. Vishwanathan. Fast and space efficient string kernels using suffix arrays. In *Proceedings of ICML*, pages 939–936. ACM Press, 2006.

K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K. Müller. A new discriminative kernel from probabilistic models. *Neural Computation*, 14:2397–2414, 2002.

E. Ukkonen. Online construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

J.-P. Vert, H. Saigo, and T. Akutsu. Local alignment kernels for biological sequences. In *Kernel methods in Computational Biology*, pages 131–154. MIT Press, 2004.

S. Vishwanathan and A. Smola. Fast kernels for string and tree matching. In K. Tsuda, B. Schölkopf, and J. Vert, editors, *Kernels and Bioinformatics*, pages 113–130. MIT Press, 2004.

U. von Luxburg and O. Bousquet. Distance-based classification with Lipschitz functions. *Journal of Machine Learning Research*, 5:669–695, 2004.

C. Watkins. Dynamic alignment kernels. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39–50, Cambridge, MA, 2000. MIT Press.

A. Webb. *Statistical Pattern Recognition*. John Wiley and Sons Ltd., 2002.

P. Weiner. Linear pattern matching algorithms. In *Proc. 14th Annual Symposium on Switching and Automata Theory*, pages 1–11, 1973.

A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering Support Vector Machine Kernels That Recognize Translation Initiation Sites. *BioInformatics*, 16(9):799–807, Sept. 2000.

Journal of Machine Learning Research 9 (2008) 49-66 Submitted 7/07; Revised 11/07; Published 1/08

On the Suitable Domain for SVM Training in Image Coding

Gustavo Camps-Valls GUSTAVO.CAMPS@UV.ES
Dept. Enginyeria Electrònica
Universitat de València
46100 Burjassot, València, Spain

Juan Gutiérrez JUAN.GUTIERREZ@UV.ES
Dept. Informàtica
Universitat de València
46100 Burjassot, València, Spain

Gabriel Gómez-Pérez GABRIEL.GOMEZ@ANALOG.COM
Audio and Video Group
Analog Devices Inc.
Limerick, Ireland

Jesús Malo JESUS.MALO@UV.ES
Dept. d'Òptica
Universitat de València
46100 Burjassot, València, Spain

Editor: Donald Geman

Abstract

Conventional SVM-based image coding methods are founded on independently restricting the distortion in every image coefficient at some particular image representation. Geometrically, this implies allowing arbitrary signal distortions in an n-dimensional rectangle defined by the ε-insensitivity zone in each dimension of the selected image representation domain. Unfortunately, not every image representation domain is well-suited for such a simple, scalar-wise, approach because statistical and/or perceptual interactions between the coefficients may exist. These interactions imply that scalar approaches may induce distortions that do not follow the image statistics and/or are perceptually annoying. Taking into account these relations would imply using non-rectangular ε-insensitivity regions (allowing coupled distortions in different coefficients), which is beyond the conventional SVM formulation.

In this paper, we report a condition on the suitable domain for developing efficient SVM image coding schemes. We analytically demonstrate that no linear domain fulfills this condition because of the statistical and perceptual inter-coefficient relations that exist in these domains. This theoretical result is experimentally confirmed by comparing SVM learning in previously reported linear domains and in a recently proposed non-linear perceptual domain that simultaneously reduces the statistical and perceptual relations (so it is closer to fulfilling the proposed condition). These results highlight the relevance of an appropriate choice of the image representation before SVM learning.

Keywords: image coding, non-linear perception models, statistical independence, support vector machines, insensitivity zone

1. Problem Statement: The Diagonal Jacobian Condition

Image coding schemes based on support vector machines (SVM) have been successfully introduced in the literature. SVMs have been used in the spatial domain (Robinson and Kecman, 2000), in the block-DCT domain (Robinson and Kecman, 2003), and in the wavelet domain (Ahmed, 2005; Jiao et al., 2005). These coding methods take advantage of the ability of the support vector regression (SVR) algorithm for function approximation using a small number of parameters (signal samples, or support vectors) (Smola and Schölkopf, 2004). In all current SVM-based image coding techniques, a representation of the image is described by the entropy-coded weights associated to the support vectors necessary to approximate the signal with a given accuracy. Relaxing the accuracy bounds reduces the number of needed support vectors. In a given representation domain, reducing the number of support vectors increases the compression ratio at the expense of bigger distortion (lower image quality). By applying the standard SVR formulation, a certain amount of distortion in each sample of the image representation is allowed. In the original formulation, scalar restrictions on the errors are introduced using a constant ε-insensitivity value for every sample.

Recently, this procedure has been refined by Gómez-Pérez et al. (2005) using a profile-dependent SVR (Camps-Valls et al., 2001) that considers a different ε for each sample or frequency. This frequency-dependent insensitivity, ε_f, accounts for the fact that, according to simple (linear) perception models, not every sample in linear frequency domains (such as DCT or wavelets) contributes to the perceived distortion in the same way.

Despite different domains have been proposed for SVM training (spatial domain, block-DCT and wavelets) and different ε insensitivities per sample have been proposed, in conventional SVR formulation, the particular distortions introduced by regression in the different samples are not coupled. In all the reported SVM-based image coding schemes, the RBF kernel is used and the penalization parameter is fixed to an arbitrarily large value. In this setting, considering n-sample signals as n-dimensional vectors, the SVR guarantees that the approximated vectors are confined in n-dimensional rectangles around the original vectors. These rectangles are just n-dimensional cubes in the standard formulation or they have certain elongation if different ε_f are considered in each axis, f. Therefore, in all the reported SVM-based coding methods, these rectangles are always oriented along the axes of the (linear) image representation. According to this, a common feature of these (scalar-wise) approaches is that they give rise to decoupled distortions in each dimension. Pérez-Cruz et al. (2002) proposed a hyperspherical insensitivity zone to correct the penalization factor in each dimension of multi-output regression problems, but again, restrictions to each sample were still uncoupled.

This scalar-wise strategy is not the best option in domains where the different dimensions of the image representation are not independent. For instance, consider the situation where actually independent components, $\mathbf{r}_f$, are obtained from a given image representation, $\mathbf{y}$, applying some eventually non-linear transform, R:

$$\mathbf{y} \xrightarrow{R} \mathbf{r}.$$

In this case, SVM regression with scalar-wise error restriction makes sense in the $\mathbf{r}$ domain. However, the original $\mathbf{y}$ domain will not be suitable for the standard SVM regression unless the matrix ∇R is diagonal (up to any permutation of the dimensions, that is, only one non-zero element per row). Therefore, if transforms that achieve independence have non-diagonal Jacobian, scalar-wise restrictions in the original (coupled coefficients) domain $\mathbf{y}$ are not allowed.

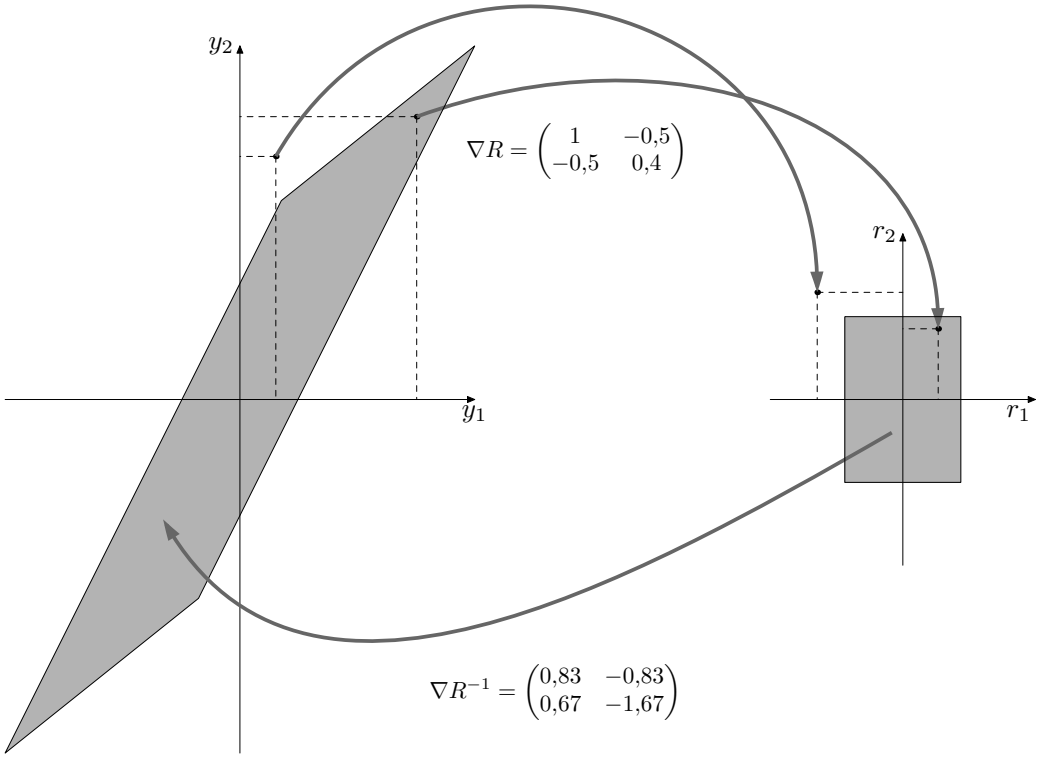


Figure 1: Insensitivity regions in different representation domains, **y** (left) and **r** (right), related by a non-diagonal transform ∇R and its inverse ∇R^{-1}.

Figure 1 illustrates this situation. The shaded region in the right plot (**r** domain) represents the *n*-dimensional box determined by the ε_f insensitivities in each dimension (f=1,2), in which a scalar-wise approach is appropriate due to independence among signal coefficients. Given that the particular ∇R transform is not diagonal, the corresponding shaded region in the left plot (the original **y** domain) is not aligned along the axes of the representation. This has negative implications: note that for the highlighted points, smaller distortions in both dimensions in the **y** domain (as implied by SVM with tighter *but scalar* ε_f insensitivities) do not necessarily imply lying inside the insensitivity region in the final truly independent (and meaningful) **r** domain. Therefore, the original **y** domain is not suitable for the direct application of conventional SVM, and consequently, non-trivial coupled insensitivity regions are required.

Summarizing, in the image coding context, the condition for an image representation **y** to be strictly suitable for conventional SVM learning is that *the transform that maps the original representation* **y** *to an independent coefficient representation* **r** *must be locally diagonal*.

As will be reviewed below, independence among coefficients (and the transforms to obtain them) may be defined in both statistical and perceptual terms (Hyvarinen et al., 2001; Malo et al., 2001; Epifanio et al., 2003; Malo et al., 2006). On the one hand, a locally diagonal relation to a statistically independent representation is desirable because independently induced distortions (as the conventional SVM approach does) will preserve the statistics of the distorted signal, that is, it will not introduce artificial-looking artifacts. On the other hand, a locally diagonal relation to a perceptually

independent representation is desirable because independently induced distortions do not give rise to increased subjective distortions due to non-trivial masking or facilitation interactions between the distortions in each dimension (Watson and Solomon, 1997).

In this work, we show that conventional linear domains do not fulfill the diagonal Jacobian condition in either the statistical case or in the perceptual case. This theoretical result is experimentally confirmed by comparing SVM learning in previously reported linear domains (Robinson and Kecman, 2003; Gómez-Pérez et al., 2005) and in a recently proposed non-linear perceptual domain that simultaneously reduces the statistical and the perceptual relations (Malo et al., 2006), thus, this non-linear perceptual domain is closer to fulfilling the proposed condition.

The rest of the paper is structured as follows. Section 2 reviews the fact that linear coefficients of the image representations commonly used for SVM training are neither statistically independent nor perceptually independent. Section 3 shows that transforms for obtaining statistical and/or perceptual independence from linear domains have non-diagonal Jacobian. This suggests that there is room to improve the performance of conventional SVM learning reported in linear domains. In Section 4, we propose the use of a perceptual representation for SVM training because it strictly fulfills the diagonal Jacobian condition in the perceptual sense and increases the statistical independence among coefficients, bringing it closer to fulfilling the condition in the statistical sense. The experimental image coding results confirm the superiority of this domain for SVM training in Section 5. Section 6 presents the conclusions and final remarks.

2. Statistical and Perceptual Relations Among Image Coefficients

Statistical independence among the coefficients of a signal representation refers to the fact that the joint PDF of the class of signals to be considered can be expressed as a product of the marginal PDFs in each dimension (Hyvarinen et al., 2001). Simple (second-order) descriptions of statistical dependence use the non-diagonal nature of the covariance matrix (Clarke, 1985; Gersho and Gray, 1992). More recent and accurate descriptions use higher-order moments, mutual information, or the non-Gaussian nature (sparsity) of marginal PDFs (Hyvarinen et al., 2001; Simoncelli, 1997).

Perceptual independence refers to the fact that the visibility of errors in coefficients of an image may depend on the energy of neighboring coefficients, a phenomenon known in the perceptual literature as *masking* or *facilitation* (Watson and Solomon, 1997). Perceptual dependence has been formalized just up to second order, and this may be described by the non-Euclidean nature of the perceptual metric matrix (Malo et al., 2001; Epifanio et al., 2003; Malo et al., 2006).

2.1 Statistical Relations

In recent years, a variety of approaches, known collectively as "independent component analysis" (ICA), have been developed to exploit higher-order statistics for the purpose of achieving a unique *linear* solution for coefficient independence (Hyvarinen et al., 2001). The basis functions obtained when these methods are applied to images are spatially localized and selective for both orientation and spatial frequency (Olshausen and Field, 1996; Bell and Sejnowski, 1997). Thus, they are similar to basis functions of multi-scale wavelet representations.

Despite its name, linear ICA does *not* actually produce statistically independent coefficients when applied to photographic images. Intuitively, independence would seem unlikely, since images are not formed from linear superpositions of independent patterns: the typical combination rule for the elements of an image is *occlusion*. Empirically, the coefficients of natural image decom-

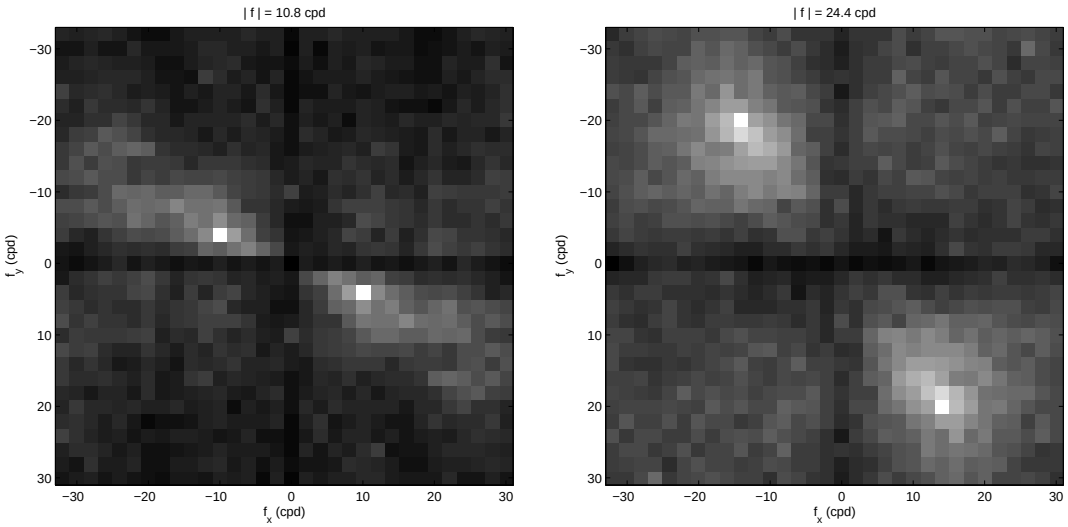


Figure 2: Statistical interaction of two particular coefficients of the local Fourier Transform with their neighbors in a natural image database. The absolute value of the frequency of these coefficients is $|f| = 10.8$ and $|f| = 24.4$ cycles/degree (cpd).

positions in spatially localized oscillating basis functions are found to be fairly well decorrelated (i.e., their covariance is almost zero). However, the amplitudes of coefficients at nearby spatial positions, orientations, and scales are highly correlated (even with orthonormal transforms) (Simoncelli, 1997; Buccigrossi and Simoncelli, 1999; Wainwright et al., 2001; Hyvarinen et al., 2003; Gutiérrez et al., 2006; Malo et al., 2006; Malo and Gutiérrez, 2006). This suggests that achieving statistical independence requires the introduction of non-linearities beyond linear ICA transforms.

Figure 2 reproduces one of many results that highlight the presence of statistical relations of natural image coefficients in block PCA or linear ICA-like domains: the energy of spatially localized oscillating filters is correlated with the energy of neighboring filters in scale and orientation (see Gutiérrez et al., 2006). A remarkable feature is that the interaction width increases with frequency, as has been reported in other domains, for example, wavelets (Buccigrossi and Simoncelli, 1999; Wainwright et al., 2001; Hyvarinen et al., 2003), and block-DCT (Malo et al., 2006).

In order to remove the remaining statistical relations in the linear domains $\mathbf{y}$, non-linear ICA methods are necessary (Hyvarinen et al., 2001; Lin, 1999; Karhunen et al., 2000; Jutten and Karhunen, 2003). Without lack of generality, non-linear ICA transforms can be schematically understood as a two-stage process (Malo and Gutiérrez, 2006):

$$\mathbf{x} \underset{\mathbf{T}^{-1}}{\overset{\mathbf{T}}{\rightleftarrows}} \mathbf{y} \underset{R^{-1}}{\overset{R}{\rightleftarrows}} \mathbf{r}\,, \tag{1}$$

where $\mathbf{x}$ is the image representation in the spatial domain, and $\mathbf{T}$ is a global unitary linear transform that removes second-order and eventually higher-order relations among coefficients in the spatial domain. Particular examples of $\mathbf{T}$ include block PCA, linear ICAs, DCT or wavelets. In the ICA literature notation, $\mathbf{T}$ is the *separating matrix* and $\mathbf{T}^{-1}$ is the *mixing matrix*. The second transform

R is an additional non-linearity that is introduced in order to remove the statistical relations that still remain in the **y** domain.

2.2 Perceptual Relations

Perceptual dependence among coefficients in different image representations can be understood by using the current model of V1 cortex. This model can also be summarized by the two-stage (linear and non-linear) process described in Equation (1). In this perceptual case, **T** is also a linear filter bank applied to the original input image in the spatial domain. This filter bank represents the linear behavior of V1 neurons whose receptive fields happen to be similar to wavelets or linear ICA basis functions (Olshausen and Field, 1996; Bell and Sejnowski, 1997). The second transform, *R*, is a non-linear function that accounts for the masking and facilitation phenomena that have been reported in the linear **y** domain (Foley, 1994; Watson and Solomon, 1997). Section 3.2 gives a parametric expression for the second non-linear stage, *R*: the divisive normalization model (Heeger, 1992; Foley, 1994; Watson and Solomon, 1997).

This class of models is based on psychophysical experiments assuming that the last domain, **r**, is perceptually Euclidean (i.e., perfect perceptual independence). An additional confirmation of this assumption is the success of (Euclidean) subjective image distortion measures defined in that domain (Teo and Heeger, 1994). Straightforward application of Riemannian geometry to obtain the perceptual metric matrix in other domains shows that the coefficients of linear domains **x** and **y**, or any other linear transform of them, are not perceptually independent (Epifanio et al., 2003).

Figure 3 illustrates the presence of perceptual relations between coefficients when using linear block frequency or wavelet-like domains, **y**: the *cross-masking* behavior. In this example, the visibility of the distortions added on top of the background image made of periodic patterns has to be assessed. This is a measure of the sensitivity of a particular perceptual mechanism to distortions in that dimension, Δy_f, when mechanisms tuned to other dimensions are simultaneously active, that is, $y_{f'} \neq 0$, with $f' \neq f$. As can be observed, low frequency noise is more visible in high frequency backgrounds than in low frequency backgrounds (e.g., left image). Similarly, high frequency noise is more visible in low frequency backgrounds than in high frequency ones (e.g., right image). That is to say, a signal of a specific frequency strongly masks the corresponding frequency analyzer, but it induces a smaller sensitivity reduction in the analyzers that are tuned to different frequencies. In other words, the reduction in sensitivity of a specific analyzer gets larger as the distance between the background frequency and the frequency of the analyzer gets smaller. The response of each frequency analyzer not only depends on the energy of the signal for that frequency band, but also on the energy of the signal in other frequency bands (cross-masking). This implies that a different amount of noise in each frequency band may be acceptable depending on the energy of that frequency band and on the energy of neighboring bands. This is what we have called *perceptual dependence* among different coefficients in the **y** domain.

At this point, it is important to stress the similarity between the set of computations to obtain statistically decoupled image coefficients and the known stages of biological vision. In fact, it has been hypothesized that biological visual systems have organized their sensors to exploit the particular statistics of the signals they have to process. See Barlow (2001), Simoncelli and Olshausen (2001), and Simoncelli (2003) for reviews on this hypothesis.

In particular, both the linear and the non-linear stages of the cortical processing have been successfully derived using redundancy reduction arguments: nowadays, the same class of linear

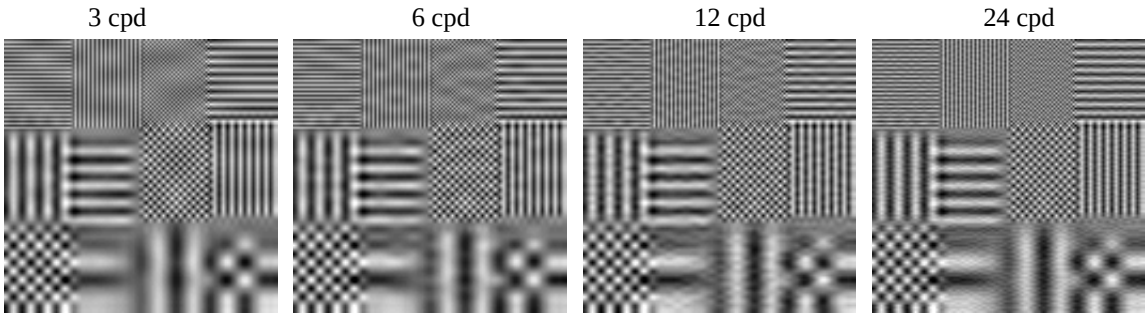


Figure 3: Illustrative example of perceptual dependence (cross-masking phenomenon). Equal energy noise of different frequency content, 3 cycl/deg (cpd), 6 cpd, 12 cpd and 24 cpd, shown on top of a background image. Sampling frequency assumes that these images subtend an angle of 3 deg.

stage **T** is used in transform coding algorithms and in vision models (Olshausen and Field, 1996; Bell and Sejnowski, 1997; Taubman and Marcellin, 2001), and new evidence supports the same idea for the second non-linear stage (Schwartz and Simoncelli, 2001; Malo and Gutiérrez, 2006). According to this, the statistical and perceptual transforms, R, that remove the above relations from the linear domains, **y**, would be very similar if not the same.

3. Statistical and Perceptual Independence Imply Non-diagonal Jacobian

In this section, we show that both statistical redundancy reduction transforms (e.g., non-linear ICA) and perceptual independence transforms (e.g., divisive normalization), have non-diagonal Jacobian for any linear image representation, so they are not strictly suitable for conventional SVM training.

3.1 Non-diagonal Jacobian in Non-linear ICA Transforms

One possible approach for dealing with global non-linear ICA is to act differentially by breaking the problem into local linear pieces that can then be integrated to obtain the global independent coefficient domain (Malo and Gutiérrez, 2006). Each differential sub-problem around a particular point (image) can be locally solved using the standard linear ICA methods restricted to the neighbors of that point (Lin, 1999).

Using the differential approach in the context of a two-stage process such as the one in Equation (1), it can be shown that (Malo and Gutiérrez, 2006):

$$\mathbf{r} = \mathbf{r_0} + \int_{\mathbf{x}_0}^{\mathbf{x}} \mathbf{T}_\ell(\mathbf{x}')\, d\mathbf{x}' = \mathbf{r_0} + \int_{\mathbf{x}_0}^{\mathbf{x}} \nabla R(\mathbf{T}\mathbf{x}')\, \mathbf{T}\, d\mathbf{x}', \tag{2}$$

where $\mathbf{T}_\ell(\mathbf{x}')$ is the local separating matrix for a neighborhood of the image $\mathbf{x}'$, and $\mathbf{T}$ is the global separating matrix for the whole PDF. Therefore, the Jacobian of the second non-linear stage is:

$$\nabla R(\mathbf{y}) = \nabla R(\mathbf{T}\mathbf{x}) = \mathbf{T}_\ell(\mathbf{x})\, \mathbf{T}^{-\mathbf{1}}. \tag{3}$$

As local linear independent features around a particular image, $\mathbf{x}$, differ in general from global linear independent features, that is, $\mathbf{T}_\ell(\mathbf{x}) \neq \mathbf{T}$, the above product is not the identity nor diagonal in general.

3.2 Non-diagonal Jacobian in Non-linear Perceptual Transforms

The current response model for the cortical frequency analyzers is non-linear (Heeger, 1992; Watson and Solomon, 1997). The outputs of the filters of the first linear stage, $\mathbf{y}$, undergo a non-linear sigmoid transform in which the energy of each linear coefficient is weighted by a linear *Contrast Sensitivity Function* (CSF) (Campbell and Robson, 1968; Malo et al., 1997) and is further normalized by a combination of the energies of neighbor coefficients in frequency,

$$r_f = R(\mathbf{y})_f = \frac{\mathrm{sgn}(y_f)\,|\alpha_f y_f|^\gamma}{\beta_f + \sum_{f'=1}^{n} h_{ff'}|\alpha_{f'} y_{f'}|^\gamma}, \tag{4}$$

where α_f (Figure 4[top left]) are CSF-like weights, β_f (Figure 4[top right]) control the sharpness of the response saturation for each coefficient, γ is the so called excitation exponent, and the matrix $h_{ff'}$ determines the interaction neighborhood in the non-linear normalization of the energy. This interaction matrix models the cross-masking behavior (cf. Section 2.2). The interaction in this matrix is assumed to be Gaussian (Watson and Solomon, 1997), and its width increases with the frequency. Figure 4[bottom] shows two examples of this Gaussian interaction for two particular coefficients in a local Fourier domain. Note that the width of the perceptual interaction neighborhood increases with the frequency in the same way as the width of the statistical interaction neighborhood shown in Figure 2. We used a value of $\gamma = 2$ in the experiments.

Taking derivatives in the general divisive normalization model, Equation (4), we obtain

$$\nabla R(\mathbf{y})_{ff'} = \mathrm{sgn}(y_f)\gamma\left(\frac{\alpha_f\,|\alpha_f y_f|^{\gamma-1}}{\beta_f + \sum_{f'=1}^{n} h_{ff'}|\alpha_{f'} y_{f'}|^\gamma}\delta_{ff'} - \frac{\alpha_{f'}|\alpha_f y_f|^\gamma|\alpha_{f'} y_{f'}|^{\gamma-1}}{(\beta_f + \sum_{f'=1}^{n} h_{ff'}|\alpha_{f'} y_{f'}|^\gamma)^2}h_{ff'}\right), \tag{5}$$

which is not diagonal because of the interaction matrix, h, which describes the cross-masking between each frequency f and the remaining $f' \neq f$.

Note that the intrinsic non-linear nature of both the statistical and perceptual transforms, Equations (3) and (5), makes the above results true for any linear domain under consideration. Specifically, if any other possible linear domain for image representation is considered, $\mathbf{y}' = \mathbf{T}'\mathbf{y}$, then the Jacobian of the corresponding independence transform, R', is

$$\nabla R'(\mathbf{y}') = \nabla R(\mathbf{y})\,\mathbf{T}'^{-\mathbf{1}},$$

which, in general, will also be non-diagonal because of the non-diagonal and point-dependent nature of $\nabla R(\mathbf{y})$.

To summarize, since no linear domain fulfills the diagonal Jacobian condition in either statistical or perceptual terms, the negative situation illustrated in Figure 1 may occur when using SVM in these domains. Therefore, improved results could be obtained if SVM learning were applied after some transform achieving independent coefficients, R.

4. SVM Learning in a Perceptually Independent Representation

In order to confirm the above theoretical results (i.e., the unsuitability of linear representation domains for SVM learning) and to assess the eventual gain that can be obtained from training SVR

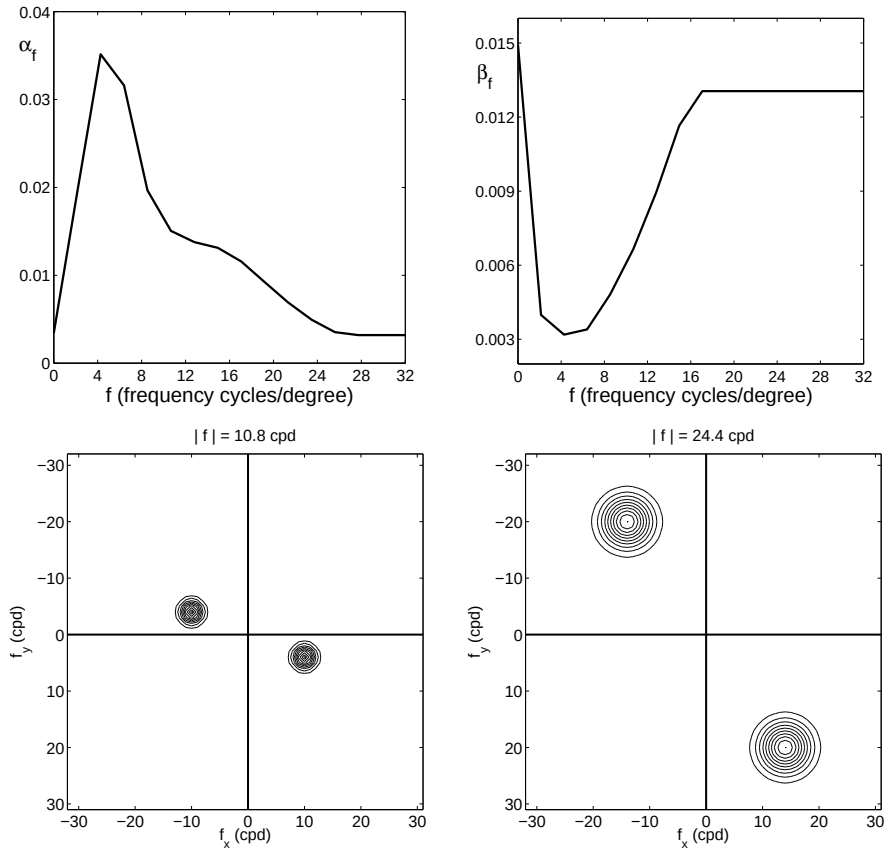


Figure 4: Parameters of the perceptual model: α_f (top left), β_f (top right). Bottom figures represent perceptual interaction neighborhoods $h_{ff'}$ of two particular coefficients of the local Fourier domain.

in a more appropriate domain, we should compare the performance of SVRs in previously reported linear domains (e.g., block-DCT or wavelets) and in one of the proposed non-linear domains (either the statistically independent domain or the perceptually independent domain).

Exploration of the statistical independence transform may have academic interest but, in its present formulation, it is not practical for coding purposes: direct application of non-linear ICA as in Equation (2) is very time-consuming for high dimensional vectors since lots of local ICA computations are needed to transform each block, and a very large image database is needed for a robust and significant computation of R. Besides, an equally expensive differential approach is also needed to compute the inverse R^{-1} for image decoding. In contrast, the perceptual non-linearity (and its inverse) are analytical. These analytical expressions are feasible for reasonable block sizes, and there are efficient iterative methods that can be used for larger vectors (Malo et al., 2006). In this paper, we explore the use of a psychophysically-based divisive normalized domain: first compute a block-DCT transform and then apply the divisive normalization model described above for each block. The results will be compared to the first competitive SVM coding results (Robinson

and Kecman, 2003) and the posterior improvements reported by Gómez-Pérez et al. (2005), both formulated in the linear block-DCT domain.

As stated in Section 2, by construction, the proposed domain is perceptually Euclidean with perceptually independent components. The Euclidean nature of this domain has an additional benefit: the ε-insensitivity design is very simple because a constant value is appropriate due to the constant perceptual relevance of all coefficients. Thus, direct application of the standard SVR method is theoretically appropriate in this domain.

Moreover, beyond its built-in perceptual benefits, this psychophysically-based divisive normalization has attractive statistical properties: it strongly reduces the mutual information between the final coefficients $\mathbf{r}$ (Malo et al., 2006). This is not surprising according to the hypothesis that try to explain the early stages of biological vision systems using information theory arguments (Barlow, 1961; Simoncelli and Olshausen, 2001). Specifically, dividing the energy of each linear coefficient by the energy of the neighbors, which are statistically related with it, cf. Figure 2, gives coefficients with reduced statistical dependence. Moreover, as the empirical non-linearities of perception have been reproduced using non-linear ICA in Equation (2) (Malo and Gutiérrez, 2006), the empirical divisive normalization can be seen as a convenient parametric way to obtain statistical independence.

5. Performance of SVM Learning in Different Domains

In this section, we analyze the performance of SVM-based coding algorithms in linear and non-linear domains through rate-distortion curves and explicit examples for visual comparison. In addition, we discuss how SVM selects support vectors in these domains to represent the image features.

5.1 Model Development and Experimental Setup

In the (linear) block-DCT domain, $\mathbf{y}$, we use the method introduced by Robinson and Kecman (2003) (RKi-1), in which the SVR is trained to learn a fixed (low-pass) number of DCT coefficients (those with frequency bigger than 20 cycl/deg are discarded); and the method proposed by Gómez-Pérez et al. (2005) (CSF-SVR), in which the relevance of all DCT coefficients is weighted according to the CSF criterion using an appropriately modulated ε_f. In the non-linear domain, $\mathbf{r}$, we use the SVR with constant insensitivity parameter ε (NL-SVR). In all cases, the block-size is 16×16, that is, $\mathbf{y}, \mathbf{r} \in \mathbb{R}^{256}$. The behavior of JPEG standard is also included in the experiments for comparison purposes.

As stated in Section 1, we used the RBF kernel and arbitrarily large penalization parameter in every SVR case. In all experiments, we trained the SVR models without the bias term, and modelled the absolute value of the DCT, $\mathbf{y}$, or response coefficients, $\mathbf{r}$. All the remaining free parameters (ε-insensitivity and Gaussian width of the RBF kernel σ) were optimized for all the considered models and different compression ratios. In the NL-SVM case, the parameters of the divisive normalization used in the experiments are shown in Figure 4. After training, the signal is described by the uniformly quantized Lagrange multipliers of the support vectors needed to keep the regression error below the thresholds ε_f. The last step is entropy coding of the quantized weights. The compression ratio is controlled by a factor applied to the thresholds, ε_f.

5.2 Model Comparison

In order to assess the quality of the coded images, three different measures were used: the standard (Euclidean) RMSE, the Maximum Perceptual Error (MPE) (Malo et al., 2000; Gómez-Pérez et al., 2005; Malo et al., 2006) and the also perceptually meaningful Structural SIMilarity (SSIM) index (Wang et al., 2004). Eight standard 256×256 monochrome 8 bits/pix images were used in the experiments. Average rate-distortion curves are plotted in Figure 5 in the range [0.05, 0.6] bits/pix (bpp). According to these entropy-per-sample data, original file size was 64 KBytes in every case, while the compressed image sizes were in the range [0.4, 4.8] KBytes. This implies that the compression ratios were in the range [160:1, 13:1].

In general, a clear gain over standard JPEG is obtained by all SVM-based methods. According to the standard Euclidean MSE point of view, the performance of RKi-1 and CSF-SVR algorithms is basically the same (note the overlapped curves in Figure 5(a)). However, it is widely known that the MSE results are not useful to represent the subjective quality of images, as extensively reported elsewhere (Girod, 1993; Teo and Heeger, 1994; Watson and Malo, 2002). When using more appropriate (perceptually meaningful) quality measures (Figures 5(b)-(c)), the CSF-SVR obtains a certain advantage over the RKi-1 algorithm for all compression rates, which was already reported by Gómez-Pérez et al. (2005). In all measures, and for the whole considered entropy range, the proposed NL-SVR clearly outperforms all previously reported methods, obtaining a noticeable gain at medium-to-high compression ratios (between 0.1 bpp (80:1) and 0.3 bpp (27:1)). Taking into account that the recommended bit rate for JPEG is about 0.5 bpp, from Figure 5 we can also conclude that the proposed technique achieves the similar quality levels at a lower bit rate in the range [0.15, 0.3] bpp.

Figure 6 shows representative visual results of the considered SVM strategies on standard images (Lena and Barbara) at the same bit rate (0.3 bpp, 27:1 compression ratio or 2.4 KBytes in 256×256 images). The visual inspection confirms that the *numerical* gain in MPE and SSIM shown in Figure 5 is also *perceptually significant*. Some conclusions can be extracted from this figure. First, as previously reported by Gómez-Pérez et al. (2005), RKi-1 leads to poorer (blocky) results because of the crude approximation of the CSF (as an ideal low-pass filter) and the equal relevance applied to the low-frequency DCT-coefficients. Second, despite the good performance yielded by the CSF-SVR approach to avoid blocking effects, it is worth noting that high frequency details are smoothed (e.g., see Barbara's scarf). These effects are highly alleviated by introducing SVR in the non-linear domain. See, for instance, Lena's eyes, her hat's feathers or the better reproduction of the high frequency pattern in Barbara's clothes.

Figure 7 shows the results obtained by all considered methods at a very high compression ratio for the Barbara image (0.05 bpp, 160:1 compression ratio or 0.4 KBytes in 256×256 images). This experiment is just intended to show the limits of methods performance since it is out of the recommended rate ranges. Even though this scenario is unrealistic, differences among methods are still noticeable: the proposed NL-SVR method reduces the blocky effects (note for instance that the face is better reproduced). This is due to a better distribution of support vectors in the perceptually independent domain.

5.3 Support Vector Distribution

The observed different perceptual image quality obtained with each approach is a direct consequence of support vector distribution in different domains. Figure 8 shows a representative example

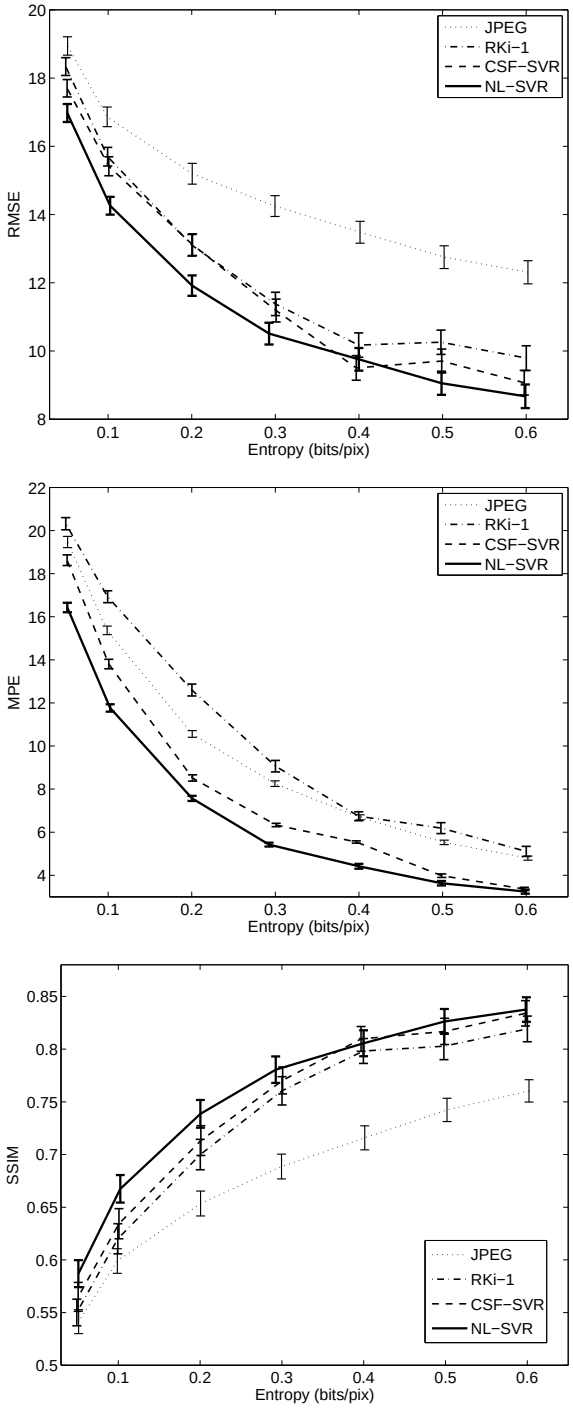


Figure 5: Average rate distortion curves over eight standard images (Lena, Barbara, Boats, Einstein, Peppers, Mandrill, Goldhill, Camera man) using objective and subjective measures for the considered JPEG (dotted) and the SVM approaches (RKi-1 dash-dotted, CSF-SVR dashed and NL-SVR solid). RMSE distortion (top), Maximum Perceptual Error, MPE (middle) (Malo et al., 2000; Gómez-Pérez et al., 2005; Malo et al., 2006), and Structural SIMilarity index, SSIM (bottom) (Wang et al., 2004).

Figure 6: Examples of decoded Lena (left) and Barbara (right) images at 0.3 bits/pix. From top to bottom: JPEG, RKi-1, CSF-SVR, and NL-SVR.

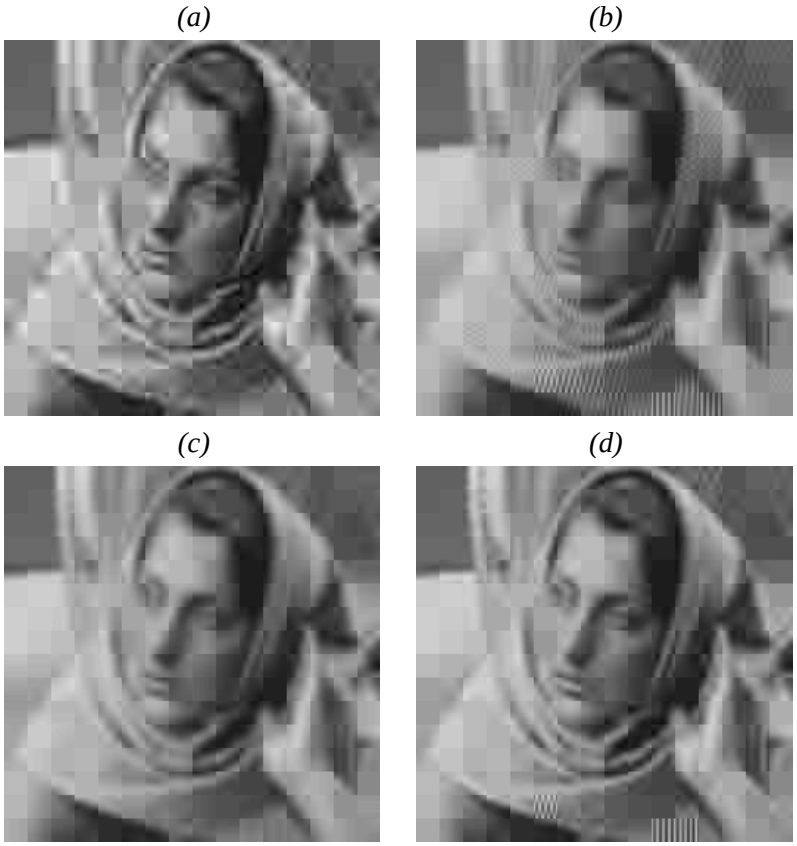


Figure 7: Examples of decoded Barbara images at a high compression ratio of 0.05 bits/pix (160:1) for (a) JPEG, (b) RKi-1, (c) CSF-SVR, and (d) NL-SVR.

of the distribution of the selected support vectors by the RKi-1 and the CSF-SVR models working in the linear DCT domain, and the NL-SVM working in the perceptually independent non-linear domain $\mathbf{r}$. Specifically, a block of Barbara's scarf at different compression ratios is used for illustration purposes.

The RKi-1 approach (Robinson and Kecman, 2003) uses a constant ε but, in order to consider the low subjective relevance of the high-frequency region, the corresponding coefficients are neglected. As a result, this approach only allocates support vectors in the low/medium frequency regions. The CSF-SVR approach uses a variable ε according to the CSF and gives rise to a more natural concentration of support vectors in the low/medium frequency region, which captures medium to high frequency details at lower compression rates (0.5 bits/pix). Note that the number of support vectors is bigger than in the RKi-1 approach, but it selects some necessary high-frequency coefficients to keep the error below the selected threshold. However, for bigger compression ratios (0.3 bits/pix), it misrepresents some high frequency, yet relevant, features (e.g., the peak from the stripes). The NL-SVM approach works in the non-linear transform domain, in which a more uniform coverage

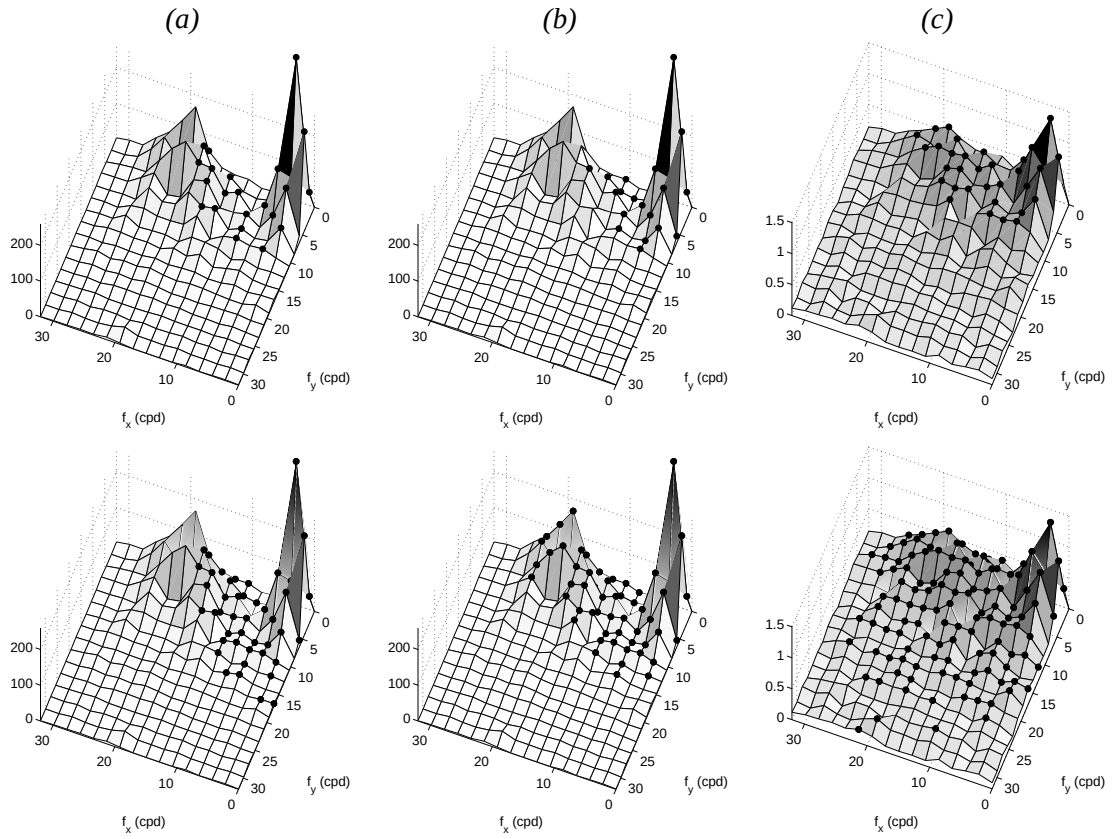


Figure 8: Signal in different domains and the selected support vectors by the SVM models in a block of the Barbara image at 0.3 bits/pix (top row) and 0.5 bits/pix (bottom row). Different domains are analyzed: (a) linear DCT using RKi-1, (b) linear DCT with CSF-SVM, and (c) non-linear perceptual domain with standard ε-SVM (NL-SVR).

of the domain is done, accounting for richer (and perceptually independent) coefficients to perform efficient sparse signal reconstruction.

It is important to remark that, for a given method (or domain), tightening ε_f implies (1) considering more support vectors, and (2) an increase in entropy (top and bottom rows in Figure 8, 0.3 bpp to 0.5 bpp). However, note that the relevant measure is the entropy and not the number of support vectors: even though the number of selected support vectors in the **r** domain is higher, their variance is lower, thus giving rise to the same entropy after entropy coding.

6. Conclusions

In this paper, we have reported a condition on the suitable domain for developing efficient SVM image coding schemes. The so-called *diagonal Jacobian condition* states that SVM regression with scalar-wise error restriction in a particular domain makes sense only if the transform that maps this domain to an independent coefficient representation is locally diagonal. We have demonstrated that,

in general, linear domains do not fulfill this condition because non-trivial statistical and perceptual inter-coefficient relations do exist in these domains.

This theoretical finding has been experimentally confirmed by observing that improved compression results are obtained when SVM is applied in a non-linear perceptual domain that starts from the same linear domain used by previously reported SVM-based image coding schemes. These results highlight the relevance of an appropriate image representation choice before SVM learning.

Further work is tied to the use of SVM-based coding schemes in statistically, rather than perceptually, independent non-linear ICA domains. In order to do so, local PCA instead of local ICA may be used in the local-to-global differential approach (Malo and Gutiérrez, 2006) to speed up the non-linear computation.

Acknowledgments

This work has been partly supported by the Spanish Ministry of Education and Science under grant CICYT TEC2006-13845/TCM and by the Generalitat Valenciana under grant GV-06/215.

References

R. Ahmed. Wavelet-based image compression using SVM learning and encoding techniques. *Proc. 8th IASTED International Conference Computer Graphics and Imaging*, 1:162–166, 2005.

H. B. Barlow. Redundancy reduction revisited. *Network: Comp. Neur. Sys.*, 12:241–253, 2001.

H. B. Barlow. Possible principles underlying the transformation of sensory messages. In WA Rosenblith, editor, *Sensory Communication*, pages 217–234. MIT Press, Cambridge, MA, 1961.

A. J. Bell and T. J. Sejnowski. The 'independent components' of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

R. W. Buccigrossi and E. P. Simoncelli. Image compression via joint statistical characterization in the wavelet domain. *IEEE Transactions on Image Processing*, 8(12):1688–1701, 1999.

F. W. Campbell and J. G. Robson. Application of Fourier analysis to the visibility of gratings. *Journal of Physiology*, 197(3):551–566, August 1968.

G. Camps-Valls, E. Soria-Olivas, J. Pérez-Ruixo, A. Artés-Rodríguez, F. Pérez-Cruz, and A. Figueiras-Vidal. A profile-dependent kernel-based regression for cyclosporine concentration prediction. In *Neural Information Processing Systems (NIPS) – Workshop on New Directions in Kernel-Based Learning Methods*, Vancouver, Canada, December 2001.

R. J. Clarke. *Transform Coding of Images*. Academic Press, New York, 1985.

I. Epifanio, J. Gutiérrez, and J. Malo. Linear transform for simultaneous diagonalization of covariance and perceptual metric matrix in image coding. *Pattern Recognition*, 36(8):1799–1811, August 2003.

J. M. Foley. Human luminance pattern mechanisms: Masking experiments require a new model. *Journal of the Optical Society of America A*, 11(6):1710–1719, 1994.

A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Press, Boston, 1992.

B. Girod. What's wrong with mean-squared error. In A. B. Watson, editor, *Digital Images and Human Vision*, pages 207–220. The MIT press, 1993.

G. Gómez-Pérez, G. Camps-Valls, J. Gutiérrez, and J. Malo. Perceptually adaptive insensitivity for support vector machine image coding. *IEEE Transactions on Neural Networks*, 16(6):1574–1581, 2005.

J. Gutiérrez, F. J. Ferri, and J. Malo. Regularization operators for natural images based on nonlinear perception models. *IEEE Transactions on Image Processing*, 15(1):189–2000, January 2006.

D. J. Heeger. Normalization of cell responses in cat striate cortex. *Vis. Neurosci.*, 9:181–198, 1992.

A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, New York, 2001.

A. Hyvarinen, J. Hurri, and J. Vayrynenm. Bubbles: a unifying framework for low-level statistical properties of natural image sequences. *Journal of the Optical Society of America A*, 20(7), 2003.

R. Jiao, Y. Li, Q. Wang, and B. Li. SVM regression and its application to image compression. In *International Conference on Intelligent Computing*, volume 3645, pages 747–756. Lecture Notes on Computer Science, 2005.

C. Jutten and J. Karhunen. Advances in nonlinear blind source separation. *Proc. of the 4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)*, pages 245–256, 2003.

J. Karhunen, S. Malaroiu, and M. Ilmoniemi. Local linear independent component analysis based on clustering. *Intl. J. Neur. Syst.*, 10(6), December 2000.

J. K. Lin. Factorizing probability density functions: Generalizing ICA. In *Proc. of the First Intl. Workshop on ICA and Signal Separation*, pages 313–318, Aussois, France, 1999.

J. Malo and J. Gutiérrez. V1 non-linear properties emerge from local-to-global non-linear ICA. *Network: Computation in Neural Systems*, 17:85–102, 2006.

J. Malo, A. M. Pons, A. Felipe, and J. M. Artigas. Characterization of human visual system threshold performance by a weighting function in the Gabor domain. *Journal of Modern Optics*, 44(1): 127–148, 1997.

J. Malo, F. Ferri, J. Albert, J. Soret, and J. M. Artigas. The role of perceptual contrast non-linearities in image transform coding. *Image & Vision Computing*, 18(3):233–246, February 2000.

J. Malo, J. Gutiérrez, I. Epifanio, F. Ferri, and J. M. Artigas. Perceptual feed-back in multigrid motion estimation using an improved DCT quantization. *IEEE Transactions on Image Processing*, 10(10):1411–1427, October 2001.

J. Malo, I. Epifanio, R. Navarro, and E. Simoncelli. Non-linear image representation for efficient perceptual coding. *IEEE Transactions on Image Processing*, 15(1):68–80, 2006.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

F. Pérez-Cruz, G. Camps-Valls, E. Soria-Olivas, J. J. Pérez-Ruixo, A. R. Figueiras-Vidal, and A. Artés-Rodríguez. Multi-dimensional function approximation and regression estimation. In *International Conference on Artificial Neural Networks, ICANN2002*, Madrid, Spain., Aug 2002. Lecture Notes in Computer Science. Springer–Verlag.

J. Robinson and V. Kecman. The use of Support Vector Machines in image compression. In *Proceedings of the International Conference on Engineering Intelligence Systems, EIS'2000*, volume 36, pages 93–96, University of Paisley, Scotland, U.K., June 2000.

J. Robinson and V. Kecman. Combining Support Vector Machine learning with the discrete cosine transform in image compression. *IEEE Transactions Neural Networks*, 14(4):950–958, July 2003.

O. Schwartz and E. P. Simoncelli. Natural signal statistics and sensory gain control. *Nature Neuroscience*, 4(8):819–825, 2001.

E. P. Simoncelli. Vision and the statistics of the visual environment. *Current Opinion in Neurobiology*, 13(2):144–149, 2003.

E. P. Simoncelli. Statistical models for images: Compression, restoration and synthesis. In *31st Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA*, 1997.

E. P. Simoncelli and B. O. Olshausen. Natural image statistics and neural representation. *Annual Review of Neuroscience*, 24:1193–1216, 2001.

A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

D. S. Taubman and M. W. Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers, Boston, 2001.

P. C. Teo and D. J. Heeger. Perceptual image distortion. *Proceedings of the First IEEE International Conference on Image Processing*, 2:982–986, 1994.

M. J. Wainwright, E. P. Simoncelli, and A. S. Willsky. Random cascades on wavelet trees and their use in analyzing and modeling natural images. *Applied and Computational Harmonic Analysis*, 11:89–123, 2001.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

A. B. Watson and J. Malo. Video quality measures based on the standard spatial observer. In *Proceedings of the IEEE International Conference on Image Proceedings*, volume 3, pages 41–44, 2002.

A. B. Watson and J. A. Solomon. A model of visual contrast gain control and pattern masking. *Journal of the Optical Society of America A*, 14(9):2379–2391, September 1997.

Journal of Machine Learning Research 9 (2008) 67-104 Submitted 3/07; Revised 9/07; Published 1/08

Discriminative Learning of Max-Sum Classifiers

Vojtěch Franc[*] XFRANCV@CMP.FELK.CVUT.CZ
Fraunhofer-FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany

Bogdan Savchynskyy BOGDAN@IMAGE.KIEV.UA
The International Research and Training Centre of Information Technologies and Systems
Prospect Akademika Glushkova, 40
Kiev, Ukraine, 03680

Editor: Marina Meilă

Abstract

The max-sum classifier predicts *n*-tuple of labels from *n*-tuple of observable variables by maximizing a sum of quality functions defined over neighbouring pairs of labels and observable variables. Predicting labels as MAP assignments of a Random Markov Field is a particular example of the max-sum classifier. Learning parameters of the max-sum classifier is a challenging problem because even computing the response of such classifier is NP-complete in general. Estimating parameters using the Maximum Likelihood approach is feasible only for a subclass of max-sum classifiers with an acyclic structure of neighbouring pairs. Recently, the discriminative methods represented by the perceptron and the Support Vector Machines, originally designed for binary linear classifiers, have been extended for learning some subclasses of the max-sum classifier. Besides the max-sum classifiers with the acyclic neighbouring structure, it has been shown that the discriminative learning is possible even with arbitrary neighbouring structure provided the quality functions fulfill some additional constraints. In this article, we extend the discriminative approach to other three classes of max-sum classifiers with an arbitrary neighbourhood structure. We derive learning algorithms for two subclasses of max-sum classifiers whose response can be computed in polynomial time: (i) the max-sum classifiers with supermodular quality functions and (ii) the max-sum classifiers whose response can be computed exactly by a linear programming relaxation. Moreover, we show that the learning problem can be approximately solved even for a general max-sum classifier.

Keywords: max-xum classifier, hidden Markov networks, support vector machines

1. Introduction

Let $(\mathcal{T},\mathcal{E})$ be an undirected graph, where $\mathcal{T}$ is a finite *set of objects* and $\mathcal{E} \subseteq \binom{\mathcal{T}}{2}$ is a set of object pairs defining *a neighborhood structure*. A pair $\{t,t'\}$ of objects belonging to $\mathcal{E}$ will be called *neighbors* or *neighboring objects*. Let each object $t \in \mathcal{T}$ be characterized by an observation x_t and a label y_t which take values from a finite set $\mathcal{X}$ and $\mathcal{Y}$, respectively. We denote an ordered $|\mathcal{T}|$-tuple of observations by $\mathbf{x} = (x_t \in \mathcal{X} \mid t \in \mathcal{T})$ and an ordered $|\mathcal{T}|$-tuple of labels by $\mathbf{y} = (y_t \in \mathcal{Y} \mid t \in \mathcal{T})$. Each object $t \in \mathcal{T}$ is assigned a function $q_t \colon \mathcal{Y} \times \mathcal{X} \to \mathbb{R}$ which determines a *quality* of a label y_t given an observation x_t. Each object pair $\{t,t'\} \in \mathcal{E}$ is assigned a function $g_{tt'} \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

∗. Secondary address for the first author is Center for Machine Perception, Czech Technical University, Faculty of Electrical Engineering, Technicka 2, 166 27 Prague 6, Czech Republic.

which determines the *quality* of labels y_t and $y_{t'}$. We adopt the convention $g_{tt'}(y,y') = g_{t't}(y',y)$. Let $\mathbf{q} \in \mathbb{R}^{|\mathcal{T}||\mathcal{X}||\mathcal{Y}|}$ and $\mathbf{g} \in \mathbb{R}^{|\mathcal{E}||\mathcal{Y}|^2}$ be ordered tuples which contain elements $q_t(y,x)$, $t \in \mathcal{T}$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$ and $g_{tt'}(y,y')$, $\{t,t'\} \in \mathcal{E}$, $y,y' \in \mathcal{Y}$, respectively. We consider a class of *structured classifiers* $f\colon \mathcal{X}^{\mathcal{T}} \to \mathcal{Y}^{\mathcal{T}}$ parametrized by $(\mathbf{q},\mathbf{g})$ that predict labeling $\mathbf{y}$ from observations $\mathbf{x}$ by selecting the labeling with the maximal quality, that is,

$$f(\mathbf{x};\mathbf{q},\mathbf{g}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} \left[\sum_{t\in\mathcal{T}} q_t(y_t,x_t) + \sum_{\{t,t'\}\in\mathcal{E}} g_{tt'}(y_t,y_{t'}) \right]. \tag{1}$$

We will call the classification rule of the form (1) a *max-sum classifier*. The problem of computing the output of the max-sum classifier, that is, the evaluation of the right-hand side of (1) is called the *max-sum labeling problem* or shortly the *max-sum problem* (it is also known as the weighted constraint satisfaction problem). The max-sum problem is known to be NP-complete in general. We will use the six-tuple $(\mathcal{T},\mathcal{E},\mathcal{Y},\mathbf{q},\mathbf{g},\mathbf{x})$ to denote a max-sum problem instance which must be solved to classify $\mathbf{x}$.

This article deals with a problem of learning the parameters $(\mathbf{q},\mathbf{g})$ of a max-sum classifier from a finite training set $\mathcal{L} = \left\{(\mathbf{x}^j,\mathbf{y}^j) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \mid j \in \{1,\dots,m\}\right\}$. Henceforth we will use the shortcut $\mathcal{J} = \{1,\dots,m\}$.

A typical example of a max-sum classifier is the maximum aposteriori (MAP) estimation in Markov models. In this case, the observations $\mathbf{x}$ and the labels $\mathbf{y}$ are assumed to be realizations of random variables $\mathbf{X} = (X_t \mid t \in \mathcal{T})$ and $\mathbf{Y} = (Y_t \mid t \in \mathcal{T})$. It is assumed that only the pairs of variables (X_t,Y_t), $t \in \mathcal{T}$ and $(Y_t,Y_{t'})$, $\{t,t'\} \in \mathcal{E}$ are directly statistically dependent. Then the joint probability of $\mathbf{X}$ and $\mathbf{Y}$ is given by the Gibbs distribution

$$P(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g}) = \frac{1}{Z} \exp F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g}), \tag{2}$$

where Z is the *partition function* which normalizes the distribution. The optimal Bayesian classifier which minimizes the probability of misclassification $f(\mathbf{x}) \neq \mathbf{y}$ assigns labels according to $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} P(\mathbf{y}\,|\,\mathbf{x})$. It is easy to see that the classifier (1) coincides with the optimal one which minimizes the misclassifications.

Applications of the classifier (1) are image de-noising (Besag, 1986), image labeling (Chou and Brown, 1990) stereo matching (Boykov et al., 2001), natural language processing (Collins, 2002), 3D image segmentation (Anguelov et al., 2005), etc.

1.1 Existing Approaches to Learning Max-Sum Classifiers

A *generative approach* to learning a max-sum classifier is based on an estimation of parameters $(\mathbf{q},\mathbf{g})$ of the Gibbs distribution (2). Having the distribution estimated, a classifier minimizing an expected risk (Bayesian risk) for a given loss function can be inferred using the Bayesian decision making framework. Maximum-Likelihood (ML) estimation methods are well known for Markov models with an acyclic graph $(\mathcal{T},\mathcal{E})$ (see, for example, Schlesinger and Hlaváč, 2002). In the general case, however, the ML estimation is not tractable because no polynomial time algorithm is known for computing a partition function exactly. Approximate methods to compute the partition function are based on a Gibbs sampler (Hinton and Sejnowski, 1986; Jerrum and Sinclair, 1993). Another disadvantage of the generative approach is the fact that little is known about an expected risk of the classifiers inferred from an imprecisely estimated statistical model.

A *discriminative approach* is an alternative method which does not require explicit modeling of the underlying probability distribution. It is based on a direct optimization of classifier parameters $(\mathbf{q},\mathbf{g})$ in order to minimize an error estimate of a classifier performance computed on a finite training set. It is easy to see that the score function $F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g})$ of the max-sum classifier (1) is linear in its parameters $(\mathbf{q},\mathbf{g})$ which allows to exploit methods for learning linear classifiers. In the case of a consistent training set, that is, if there exists a classifier with zero empirical error, the problem of learning a linear classifier can be expressed as a problem of satisfying a set of linear inequalities. This problem is efficiently solvable by the perceptron algorithm. Variants of the perceptron algorithm for learning parameters of the max-sum classifier (1) with a chain and tree neighborhood structure were published in Schlesinger and Hlaváč (2002) and Collins (2002). These algorithms exploit the fact that a perceptron can be used whenever the response of the learned classifier can be computed efficiently. This applies for the acyclic neighbourhood structure since the response of the max-sum classifier can be computed by the *dynamic programming* (DP).

The *Support Vector Machines* (SVM) (Vapnik, 1998) are another representative of a discriminative approach for learning linear classifiers which has proved to be successful in numerous applications. Unlike the perceptron algorithm, the SVMs allows learning also from an inconsistent training set. Learning is formulated as minimization of a regularized risk functional which can be further transformed to a convex quadratic programming (QP) task suitable for optimization. The original SVMs are designed for learning the classifiers which estimate a single label. In the recent years, the SVMs have been extended for learning linear classifiers which can estimate a set of interdependent labels. In particular, the Hidden Markov Support Vector Machines (Altun and Hofmann, 2003; Altun et al., 2003) and the Max-Margin Markov Networks (Taskar et al., 2004b) were proposed for learning max-sum classifiers with an acyclic neighboring structure. In this case, learning requires solving a QP task with a huge number of linear constraints proportional to the cardinality of the output space of the classifier. In analogy to the perceptron, this task is tractable if there exists an efficient algorithm that solves the *loss-augmented classification* (LAC) task which involves optimizing (1) with the loss function added to the objective function $F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g})$ (c.f. Section 3.2 for details). For additively decomposable loss functions, the LAC task becomes an instance of the max-sum problem easily solvable by the DP provided the neighbourhood structure is acyclic.

Learning of the Associative Markov Networks (AMN) with an arbitrary neighbourhood structure $\mathcal{E}$ was proposed by Taskar et al. (2004a). The AMN is the max-sum classifier with the quality functions $\mathbf{g}$ restricted in a way similar to the Potts model. The LAC task was approximated by a linear programming (LP) relaxation specially derived for the AMN model. Incorporating an LP relaxation into the SVM QP task, Taskar et al. (2004a) constructed a new compact QP task which has only a polynomial number of constraints. Even though the resulting QP task is polynomially solvable, general purpose QP solvers do not provide a practical solution since they scale poorly with the problem and training set size. Recently, Taskar et al. (2006) proposed a reformulation of the structured learning problem as a convex-concave saddle-point problem which is efficiently solvable by the dual extragradient algorithm. This new framework is applicable for structured classifiers for which a certain projection step can be solved. Taskar et al. (2006) showed that the projection step is tractable for the max-sum classifiers with supermodular functions $\mathbf{g}$ and binary labels $|\mathcal{Y}| = 2$.

Tsochantaridis et al. (2005) proposed a general framework for learning linear classifiers with an interdependent labels. Their approach is based on solving the underlying SVM QP task by a cutting plane algorithm which requires as a subroutine an algorithm solving the LAC task for the particular classifier. The approach cannot be directly applied for the general max-sum classifiers since the

LAC task is not tractable. An alternative approach to the cutting plane algorithm was proposed by Ratliff and Bagnell (2006) who used subgradient methods for optimizing the SVM QP task. This approach also relies on an efficient solution to the LAC task.

An approximated cutting plane algorithm was proposed in Finley and Joachims (2005) to learn a specific structured model for correlation clustering. The inference as well as the corresponding LAC task of this clustering model are NP-complete. The authors suggested to replace an exact solution of the LAC task required in the cutting plane algorithm by its polynomially solvable LP relaxation.

1.2 Contributions

In this article, we build on the previously published approaches which use the perceptron and the SVMs for learning the max-sum classifiers. We propose learning algorithms for three classes of max-sum classifiers for which, up to our knowledge, the discriminative approaches have not been applied yet. Namely, the contributions of the article are as follows:

- We formulate and solve the problem of learning the supermodular max-sum classifier with an arbitrary neighbourhood structure $\mathcal{E}$ and without any restriction on the number of labels $|\mathcal{Y}|$ (Taskar et al., 2006, consider only two labels). We propose a variant of the perceptron algorithm for learning from a consistent training set. For an inconsistent training set, we extend the cutting plane algorithm of Tsochantaridis et al. (2005) such that it maintains the quality functions $\mathbf{g}$ supermodular during the course of the algorithm thus making the LAC task efficiently solvable.

- We formulate and solve the problem of learning the *max-sum classifier with a strictly trivial equivalent* which is a subclass of *max-sum classifiers with a trivial equivalent*. We will show in Section 2 that the latter class can be equivalently characterized by the fact that the LP relaxation (Schlesinger, 1976; Koster et al., 1998; Chekuri et al., 2001; Wainwright et al., 2002) is tight for it. It is known, that the class of problems with a trivial equivalent contains acyclic problems (Schlesinger, 1976) and supermodular problems (Schlesinger and Flach, 2000). We will extend this result to show that problems with a strictly trivial equivalent which we can learn contain the acyclic and the supermodular max-sum problems with a unique solution.

  Learning of the max-sum classifiers with a strictly trivial equivalent leads to an optimization problem with a polynomial number of linear constraints. We propose variants of the perceptron and of the cutting plane algorithm for learning from a consistent and an inconsistent training set, respectively. In this case, the LAC task does not require any specialized max-sum solver since it can be solved exhaustively. Moreover, the QP task to which we transform the learning problem is of the same form as the QP task required for ordinary multi-class SVMs (Crammer and Singer, 2001) which allows using existing optimization packages.

- We formulate the problem of learning the general max-sum classifier, that is, without any restriction on the neighborhood structure $\mathcal{E}$ and the quality functions $\mathbf{g}$. We show that the learning problem can be solved approximately by a variant of the cutting plane algorithm proposed by Finley and Joachims (2005) which uses the LP relaxation to solve the LAC task approximately. In contrast to Taskar et al. (2004a), we do not incorporate the LP relaxation

to the SVM QP task but we keep it separately which allows to exploit existing solvers for LP relaxation of the max-sum problem.

For a simplicity, we will concentrate on the max-sum classifier (1). All the proposed methods, however, are applicable whenever the set of labels $\mathcal{Y}$ is finite and the quality functions are linear in parameters, that is, they are of the form $q_t(x_t,y_t) = \langle \mathbf{w}, \Psi_t(x_t,y_t)\rangle$ and $g_{tt'}(y_t,y_t) = \langle \mathbf{w}, \Psi_{tt'}(y_t,y_{t'})\rangle$ where $\mathbf{w} \in \mathbb{R}^d$ is a parameter vector and $\Psi_t \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$, $\Psi_{tt'} \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^d$ are arbitrary fixed mappings. Furthermore, all the proposed algorithms can be introduced in the form containing only dot products between $\Psi_t(x,y)$ and $\Psi_{tt'}(y,y')$ which allows to use the kernel functions (Vapnik, 1998).

1.3 Structure of the Article

The article is organized as follows. In Section 2, we describe three classes of the max-sum classifiers for which we will later derive learning algorithms. Perceptron algorithm and the SVM framework for learning linear classifiers with an interdependent labels is reviewed in Section 3. In Section 4, we formulate problems of learning the max-sum classifiers from a consistent training set and we propose variants of the perceptron algorithm for their solution. In Section 5, we formulate problems of learning the max-sum classifiers from inconsistent training set using the SVM framework. In Section 6, we propose an extended cutting plane algorithm to solve the learning problem defined in Section 5. Section 7 describes experiments. Finally, in Section 8 we give conclusions.

2. Classes of Max-Sum Problems

In the rest of the article, we consider the max-sum problems $P = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x})$ in which $(\mathcal{T}, \mathcal{E})$ is an arbitrary undirected connected graph and $\mathbf{q}$ are arbitrary quality functions. The three classes of the max-sum problems described below differ in the structure of the finite set of labels $\mathcal{Y}$ and the quality functions $\mathbf{g}$.

2.1 General Max-Sum Problem and LP Relaxation

The first class which we consider is the general max-sum problem with no restrictions imposed on $\mathcal{Y}$ and $\mathbf{g}$. Solving (1) when P is a general max-sum problem is known to be NP-complete. An approximate solution can be found by the linear programming (LP) relaxation proposed independently by Schlesinger (1976), Koster et al. (1998), Chekuri et al. (2001), and Wainwright et al. (2002). We introduce only the basic concepts of the LP relaxation necessary for this article. For more details on the topic we refer to the mentioned publications or to a comprehensive survey in Werner (2007) from which we adopted notation and terminology.

Let $(\mathcal{T} \times \mathcal{Y}, \mathcal{E}_{\mathcal{Y}})$ denote an undirected graph with edges $\mathcal{E}_{\mathcal{Y}} = \{\{(t,y),(t',y')\} \mid \{t,t'\} \in \mathcal{E}, y,y' \in \mathcal{Y}\}$. This graph corresponds to the *trellis diagram* used to visualize Markov chains. Each *node* $(t,y) \in \mathcal{T} \times \mathcal{Y}$ and each *edge* $\{(t,y),(t',y')\} \in \mathcal{E}_{\mathcal{Y}}$ is assigned the numbers $\beta_t(y)$ and $\beta_{tt'}(y,y')$, respectively. Let $\boldsymbol{\beta} \in \mathbb{R}^{|\mathcal{T}||\mathcal{Y}|+|\mathcal{E}||\mathcal{Y}|^2}$ be an ordered tuple which contains elements $\beta_t(y)$, $(t,y) \in \mathcal{T} \times \mathcal{Y}$ and $\beta_{tt'}(y,y')$, $\{(t,y),(t',y')\} \in \mathcal{E}_{\mathcal{Y}}$. Let Λ_P denote a set of *relaxed labelings* which contains vectors $\boldsymbol{\beta}$ satisfying

$$\sum_{y' \in \mathcal{Y}} \beta_{tt'}(y,y') = \beta_t(y)\,, \;\; \{t,t'\} \in \mathcal{E}\,, y \in \mathcal{Y}\,, \qquad \sum_{y \in \mathcal{Y}} \beta_t(y) = 1\,, \;\; t \in \mathcal{T}\,, \qquad \boldsymbol{\beta} \geq \mathbf{0}\,.$$

The LP relaxation of (1) reads

$$\boldsymbol{\beta}^* = \operatorname*{argmax}_{\boldsymbol{\beta}\in\Lambda_P} \left[\sum_{t\in\mathcal{T}} \sum_{y\in\mathcal{Y}} \beta_t(y) q_t(y,x_t) + \sum_{(t,t')\in\mathcal{E}} \sum_{(y,y')\in\mathcal{Y}^2} \beta_{tt'}(y,y') g_{tt'}(y,y') \right]. \tag{3}$$

It can be seen that solving (3) with an additional constraint $\boldsymbol{\beta} \in \{0,1\}^{\mathcal{T}}$ is an integer programming problem equivalent to (1). Further, we will introduce concepts of *equivalent problems*, *equivalent transformations* and *trivial problems* which are tightly connected to the LP relaxation.

A representation of the max-sum problem $P = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x})$ is not minimal since there exists an infinite number of equivalent max-sum problems $P' = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}', \mathbf{g}', \mathbf{x})$ with different quality functions $(\mathbf{q}', \mathbf{g}')$ but the same quality for all labelings: the max-sum problems P and P' are called *equivalent* if $F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g}) = F(\mathbf{x},\mathbf{y};\mathbf{q}',\mathbf{g}')$ for all $\mathbf{y} \in \mathcal{Y}^{\mathcal{T}}$ (Schlesinger, 1976; Wainwright et al., 2002).

Next, we introduce a transformation of a max-sum problem to its arbitrary equivalent. These *equivalent transformations* are originally due to Schlesinger (1976) and recently are also known as *reparametrization* in Wainwright et al. (2002) and Kolmogorov (2006). Let $\varphi_{tt'} \colon \mathcal{Y} \to \mathbb{R}$ and $\varphi_{t't} \colon \mathcal{Y} \to \mathbb{R}$ be a pair of functions introduced for each pair of neighbouring objects $\{t,t'\} \in \mathcal{E}$, that is, we have $2|\mathcal{E}|$ functions in total. The value $\varphi_{tt'}(y)$ is called *potential* at label y of an object t in the direction t'. Note that the potentials correspond to messages in belief propagation (see, for example, Pearl, 1988; Yedidia et al., 2005). We will use $\boldsymbol{\varphi} \in \mathbb{R}^{2|\mathcal{E}||\mathcal{Y}|}$ to denote an ordered tuple which contains elements $\varphi_{tt'}(y)$, $\{t,t'\} \in \mathcal{E}$, $y \in \mathcal{Y}$ and $\varphi_{t't}(y')$, $\{t,t'\} \in \mathcal{E}$, $y' \in \mathcal{Y}$. Let $\mathcal{N}(t) = \{t' \in \mathcal{T} \mid \{t,t'\} \in \mathcal{E}\}$ denote the set of objects neighbouring with the object $t \in \mathcal{T}$. Finally, let $P^{\boldsymbol{\varphi}} = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}^{\boldsymbol{\varphi}}, \mathbf{g}^{\boldsymbol{\varphi}}, \mathbf{x})$ denote the max-sum problem constructed from the max-sum problem $P = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x})$ by the following transformation

$$g^{\boldsymbol{\varphi}}_{tt'}(y,y') = g_{tt'}(y,y') + \varphi_{tt'}(y) + \varphi_{t't}(y'), \quad \{t,t'\} \in \mathcal{E}, y, y' \in \mathcal{Y}, \tag{4a}$$

$$q^{\boldsymbol{\varphi}}_{t}(y,x_t) = q_t(y,x_t) - \sum_{t'\in\mathcal{N}(t)} \varphi_{tt'}(y), \quad t \in \mathcal{T}, y \in \mathcal{Y}. \tag{4b}$$

By substituting (4) to $F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g}) = F(\mathbf{x},\mathbf{y};\mathbf{q}',\mathbf{g}')$ it is easy to show that the problems P and $P^{\boldsymbol{\varphi}}$ are equivalent for arbitrary potentials $\boldsymbol{\varphi}$. Moreover, it has been shown (Schlesinger, 1976; Kolmogorov, 2006) that the converse is also true, that is, if any two max-sum problems are equivalent then they are related by (4) for some potentials $\boldsymbol{\varphi}$.

Now, we can define the class of *trivial max-sum problems*. Node (t,y) is called *maximal* if $q_t(y,x_t) = \max_{y\in\mathcal{Y}} q_t(y,x_t)$ and edge $\{(t,y),(t',y')\}$ is called maximal if $g_{tt'}(y,y') = \max_{y,y'\in\mathcal{Y}} g_{tt'}(y,y')$. The max-sum problem P is called *trivial* if there exists a labeling $\mathbf{y}$ which can be formed only from the maximal nodes and edges. Checking whether a given P is trivial leads to an instance of a *constraint satisfaction problem* CSP (also called *consistent labeling problem*) (Rosenfeld et al., 1976; Haralick and Shapiro, 1979). The CSP is NP-complete in general and it is equivalent to solving a max-sum problem when the quality functions $(\mathbf{q},\mathbf{g})$ take only two values $\{-\infty, 0\}$. Let $U(\mathbf{x};\mathbf{q},\mathbf{g})$ be a *height* (also called *energy*) of the max-sum problem P defined as a sum of qualities of the maximal nodes and the maximal edges, that is,

$$U(\mathbf{x};\mathbf{q},\mathbf{g}) = \sum_{t\in\mathcal{T}} \max_{y\in\mathcal{Y}} q_t(y,x_t) + \sum_{\{t,t'\}\in\mathcal{E}} \max_{y,y'\in\mathcal{Y}} g_{tt'}(y,y'). \tag{5}$$

Comparing (1) and (5) shows that $U(\mathbf{x};\mathbf{q},\mathbf{g})$ is an upper bound on the quality of the optimal labeling, that is, $U(\mathbf{x};\mathbf{q},\mathbf{g}) \geq \max_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g})$. It is easy to see that the upper bound is tight if and only if the max-sum problem is trivial. The following result is central to the LP relaxation:

Theorem 1 *Schlesinger (1976); Werner (2005) Let $\mathcal{C}$ be a class of equivalent max-sum problems. Let C contain at least one trivial problem. Then any problem in C is trivial if and only if it has minimal height.*

This suggests to solve the max-sum problem P by searching for an equivalent $P^{\boldsymbol{\varphi}}$ with the minimal height, which leads to

$$\boldsymbol{\varphi}^* = \operatorname*{argmin}_{\boldsymbol{\varphi}} U(\mathbf{x}; \mathbf{q}^{\boldsymbol{\varphi}}, \mathbf{g}^{\boldsymbol{\varphi}}) \,. \tag{6}$$

The problem (6) can be expressed as an LP task which is known to be a *dual to the LP relaxation* (3). Having $\boldsymbol{\varphi}^*$ one can try to verify whether $P^{\boldsymbol{\varphi}^*}$ is trivial, that is, whether there exists a labeling composed of the maximal nodes and edges of $P^{\boldsymbol{\varphi}^*}$. If such labeling is found then it is the optimal solution of the max-sum problem P. Otherwise, one can use heuristics to find an approximate solution by searching for such labeling which contains as much maximal nodes and edges as possible.

Though (6) is solvable in polynomial time, a general purpose LP solvers are applicable only for small instances. A specialized solvers for LP relaxation were published in Koval and Schlesinger (1976). Recently, Wainwright et al. (2002) and Kolmogorov (2006) proposed a tree-reweighted algorithm based on minimizing a more general upper bound composed of trees spanning the graph $(\mathcal{T}, \mathcal{E})$ of which (5) is a special case.

We propose a learning algorithm for a general max-sum classifier which requires an arbitrary LP relaxation solver as a subroutine. We only require that the LP solver returns an approximate solution $\hat{\mathbf{y}}$ and an upper bound on $\max_{\mathbf{y} \in \mathcal{Y}^{\mathcal{T}}} F(\mathbf{x}, \mathbf{y}; \mathbf{q}, \mathbf{g})$.

2.2 Supermodular Max-Sum Problems

Definition 1 *A function $g_{tt'} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is* supermodular *if*

1. *The set of labels $\mathcal{Y}$ is totally ordered; w.l.o.g. we consider $\mathcal{Y} = \{1, 2, \ldots, |\mathcal{Y}|\}$ endowed with the natural order.*
2. *For each four-tuple $(y_t, y'_t, y_{t'}, y'_{t'}) \in \mathcal{Y}^4$ of labels such that $y_t > y'_t$ and $y_{t'} > y'_{t'}$ the following inequality holds:*

$$g_{tt'}(y_t, y_{t'}) + g_{tt'}(y'_t, y'_{t'}) \geq g_{tt'}(y_t, y'_{t'}) + g_{tt'}(y'_t, y_{t'}). \tag{7}$$

A max-sum problem in which the label set $\mathcal{Y}$ is totally ordered and all the functions $\mathbf{g}$ are supermodular is called a *supermodular max-sum problem*. In addition, we will also consider the max-sum problem in which the inequalities (7) are fulfilled strictly. In this case the corresponding max-sum problem will be called *strictly supermodular*.

A naive approach to check whether a given max-sum problem is supermodular amounts to verifying all $|\mathcal{E}| \cdot |\mathcal{Y}|^4$ inequalities in (7). We will use a more effective way which requires to check only $|\mathcal{E}| \cdot (|\mathcal{Y}| - 1)^2$ inequalities thanks to the following well-known theorem:

Theorem 2 *The function $g_{tt'} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is* supermodular *if for each each pair of labels $(y, y') \in \mathcal{Y}^2$ such that $y + 1 \in \mathcal{Y}$, $y' + 1 \in \mathcal{Y}$ the following inequality holds*

$$g_{tt'}(y, y') + g_{tt'}(y + 1, y' + 1) \geq g_{tt'}(y, y' + 1) + g_{tt'}(y + 1, y'). \tag{8}$$

Proof The proof follows trivially from the equality

$$g_{tt'}(y_t,y_{t'})+g_{tt'}(y'_t,y'_{t'})-g_{tt'}(y_t,y'_{t'})-g_{tt'}(y'_t,y_{t'}) \\ = \sum_{\substack{y_t>z\geq y'_t \\ y_{t'}>z'\geq y'_{t'}}} \Big[g_{tt'}(z,z')+g_{tt'}(z+1,z'+1)-g_{tt'}(z,z'+1)-g_{tt'}(z+1,z') \Big],$$

and the fact that all the summands are non-negative by the condition (8). ■

A similar proposition holds for strictly supermodular problems: a max-sum problem is *strictly supermodular* if for each pair of neighboring objects $\{t,t'\} \in \mathcal{E}$ and for each pair of labels $(y,y') \in \mathcal{Y}^2$ such that $y+1 \in \mathcal{Y}$, $y'+1 \in \mathcal{Y}$ the following inequality holds:

$$g_{tt'}(y,y')+g_{tt'}(y+1,y'+1) > g_{tt'}(y,y'+1)+g_{tt'}(y+1,y'). \tag{9}$$

In this paper, we exploit the fact that the optimal solution of the supermodular problems can be found in a polynomial time (Schlesinger and Flach, 2000). Kolmogorov and Zabih (2002) proposed an efficient algorithm for the binary case $|\mathcal{Y}|=2$ which is based on transforming the supermodular max-sum problem to the max-flow problem from the graph theory. A not widely known extension for a general case $|\mathcal{Y}|>2$ was proposed in Kovtun (2004) and Schlesinger (2005). We will propose learning algorithms which require an arbitrary solver for the supermodular max-sum problem as a subroutine.

2.3 Max-Sum Problems with Strictly Trivial Equivalent

In this section we consider *max-sum problems with a strictly trivial equivalent* which is a subclass of *problems with a trivial equivalent* described in Section (2.1). The main reason for defining this subclass is that these problems are more suitable for learning than problems with a trivial equivalent. It was shown (Schlesinger, 1976; Schlesinger and Flach, 2000) that problems with a trivial equivalent contain two well-known polynomially solvable subclasses of max-sum problems: (i) the problems with an acyclic neighborhood structure $\mathcal{E}$ and (ii) the supermodular problems. We will give a similar result which applies for the problems with a strictly trivial equivalent. In particular, we will show that the class of problems with a strictly trivial equivalent contains all acyclic and supermodular problems which have a unique solution. This shows that learning algorithms for the max-sum classifiers with a strictly trivial equivalent introduced below are applicable for a wide range of polynomially solvable problems.

Max-sum problems with a trivial equivalent are those for which LP relaxation (6) is tight, that is, $U(\mathbf{x};\mathbf{q}^{\boldsymbol{\varphi}^*},\mathbf{g}^{\boldsymbol{\varphi}^*}) = \max_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g})$, and thus $\max_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g})$ can be computed in a polynomial time. The tight LP relaxation, however, does not imply that the optimal solution $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}^{\mathcal{T}}} F(\mathbf{x},\mathbf{y};\mathbf{q},\mathbf{g})$ can be found in a polynomial time. As we mentioned, to find $\mathbf{y}^*$ from $(\mathbf{q}^{\boldsymbol{\varphi}^*},\mathbf{g}^{\boldsymbol{\varphi}^*})$ requires searching for a labeling formed only by the maximal nodes and edges which need not be unique. Finding such labeling leads to the CSP which is NP-complete. We exploit the fact that the labeling can be found trivially if all the maximal nodes and edges are unique. As a result, the optimal solution $\mathbf{y}^*$ can be found in a polynomial time by solving the LP relaxation and finding the maximal nodes and edges.

Definition 2 *Max-sum problem P is* strictly trivial *if:*

1. *There exists a unique maximal node (t,y) for each object $t \in \mathcal{T}$.*
2. *For each maximal edge $\{(t,y),(t',y')\} \in \mathcal{E}_{\mathcal{Y}}$ the nodes (t,y) and (t',y') are maximal.*

It is clear that a strictly trivial problem has a unique solution which is composed of all maximal nodes and edges. Checking whether a given problem is strictly trivial requires only $O(|\mathcal{T}||\mathcal{Y}| + |\mathcal{E}||\mathcal{Y}|^2)$ operations, that is, finding the maximal nodes and edges and checking if they are unique and if they form a labeling.

Recall that the max-sum problem P has a *strictly trivial equivalent* if there exists a strictly trivial problem which is equivalent to P. Finally, we give a theorem which asserts that the class of problems with a strictly trivial equivalent includes at least the acyclic problems and the supermodular problems which have a unique solution.

Theorem 3 *Let $P = (\mathcal{T},\mathcal{E},\mathcal{Y},\mathbf{q},\mathbf{g},\mathbf{x})$ be a max-sum problem and let P have a unique solution. If $(\mathcal{T},\mathcal{E})$ is an acyclic graph or quality functions $\mathbf{g}$ are supermodular then P is equivalent to some strictly trivial problem.*

Proof is given in Appendix A.

3. Discriminative Approach to Learning Structured Linear Classifiers

In this section, we review the discriminative approach to learning linear classifiers which we will later apply to the max-sum classifiers.

Let $\mathbf{x} \in \boldsymbol{\mathcal{X}}$ and $\mathbf{y} \in \boldsymbol{\mathcal{Y}}$ be observations and labels generated according to some fixed unknown distribution $P(\mathbf{x},\mathbf{y})$. We are interested in designing a classifier $f\colon \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$ which estimates labels from observations. Let $L\colon \boldsymbol{\mathcal{Y}} \times \boldsymbol{\mathcal{Y}} \to \mathbb{R}$ be a loss function penalizing a prediction $f(\mathbf{x})$ by a penalty $L(\mathbf{y}, f(\mathbf{x}))$ provided the true output is $\mathbf{y}$. The goal is to find a classifier which minimizes the *expected (Bayesian) risk*

$$R[f] = \int_{\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}}} L(\mathbf{y}, f(\mathbf{x}))\, dP(\mathbf{x},\mathbf{y})\,.$$

The risk $R[f]$ cannot be directly minimized because the distribution $P(\mathbf{x},\mathbf{y})$ is unknown. Instead, we are given a finite training set $\mathcal{L} = \{(\mathbf{x}^j,\mathbf{y}^j) \in \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}} \mid j \in \mathcal{J}\}$ i.i.d. sampled from $P(\mathbf{x},\mathbf{y})$.

The discriminative approach to learning classifiers does not require estimation of a probability distribution. It is based on direct optimization of parameters of a classifier in order to minimize a substitutional risk functional which approximates the desired risk $R[f]$ and can be computed from a finite training set. Let us consider a class of linear classifiers

$$f(\mathbf{x};\mathbf{w}) = \operatorname*{argmax}_{\mathbf{y}\in\boldsymbol{\mathcal{Y}}} \langle \mathbf{w}, \Psi(\mathbf{x},\mathbf{y})\rangle\,. \tag{10}$$

The classifier is determined by a parameter vector $\mathbf{w} \in \mathbb{R}^d$ and some fixed mapping $\Psi\colon \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}} \to \mathbb{R}^d$. A simple proxy for $R[f(\bullet;\mathbf{w})]$ is the *empirical risk*

$$R_{\text{emp}}[f(\bullet;\mathbf{w})] = \frac{1}{m}\sum_{j\in\mathcal{J}} L(\mathbf{y}^j, f(\mathbf{x}^j;\mathbf{w}))\,.$$

Let us assume a class of loss functions which satisfy $L(\mathbf{y},\mathbf{y}')=0$ if $\mathbf{y}=\mathbf{y}'$ and $L(\mathbf{y},\mathbf{y}')>0$ if $\mathbf{y}\neq\mathbf{y}'$. Further, we will distinguish two learning scenarios: (i) learning from a *consistent training set* and (ii) learning from an *inconsistent training set*. The training set $\mathcal{L}$ is *consistent* if there exists a parameter vector $\mathbf{w}^*$ such that the empirical risk of the linear classifier is zero, that is, $R_{\text{emp}}[f(\bullet;\mathbf{w}^*)]=0$. In the opposite case, the training set is *inconsistent*.

In Section 3.1, we review the perceptron algorithm suitable for learning linear classifiers from a consistent training set. The Support Vector Machine (SVM) approach to learning from an inconsistent training set is described in Section 3.2.

3.1 Learning from a Consistent Training Set Using the Perceptron Algorithm

In the case of a consistent training set $\mathcal{L}$, the learning problem is to find parameters $\mathbf{w}^*$ of the linear classifier (10) such that the empirical risk $R_{\text{emp}}[f(\bullet;\mathbf{w}^*)]=0$. This amounts to finding parameters $\mathbf{w}^*$ which satisfy the set of non-linear equations

$$\mathbf{y}^j = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x},\mathbf{y})\rangle\,, \quad j\in\mathcal{J}\,. \tag{11}$$

In the rest of the article, we will assume that the training set does not contain examples with the same observations $\mathbf{x}^j=\mathbf{x}^{j'}$ but different labelings $\mathbf{y}^j\neq\mathbf{y}^{j'}$, that is, we will search for a classifier which returns a unique labeling for each example from the training set. It is convenient to transform (11) to an equivalent problem of solving a set of linear strict inequalities

$$\langle \mathbf{w}, \Psi(\mathbf{x}^j,\mathbf{y}^j)\rangle > \langle \mathbf{w}, \Psi(\mathbf{x}^j,\mathbf{y})\rangle\,, \quad j\in\mathcal{J}, \mathbf{y}\in\mathcal{Y}\setminus\{\mathbf{y}^j\}\,,$$

which, using $\hat{\Psi}(\mathbf{x}^j,\mathbf{y}^j,\mathbf{y}) = \Psi(\mathbf{x}^j,\mathbf{y}^j) - \Psi(\mathbf{x}^j,\mathbf{y})$, can be written in a compact form

$$\langle \mathbf{w}, \hat{\Psi}(\mathbf{x}^j,\mathbf{y}^j,\mathbf{y})\rangle > 0\,, \quad j\in\mathcal{J}, \mathbf{y}\in\mathcal{Y}\setminus\{\mathbf{y}^j\}\,. \tag{12}$$

An efficient method to solve the problem (12) is the *perceptron* algorithm:

Algorithm 1 Perceptron algorithm

1: Set $\mathbf{w} := 0$.

2: Find a violated inequality in (12), that is, find indices $j^*\in\mathcal{J}$, $\mathbf{y}^*\in\mathcal{Y}\setminus\{\mathbf{y}^{j^*}\}$ such that

$$\langle \mathbf{w}, \hat{\Psi}(\mathbf{x}^{j^*},\mathbf{y}^{j^*},\mathbf{y}^*)\rangle \leq 0\,.$$

3: If there is no violated inequality then $\mathbf{w}$ solves (12) and the algorithm halts. Otherwise update the current solution

$$\mathbf{w} := \mathbf{w} + \hat{\Psi}(\mathbf{x}^{j^*},\mathbf{y}^{j^*},\mathbf{y}^*)\,,$$

and go to Step 2.

Provided the training set $\mathcal{L}$ is consistent, that is, the inequalities (12) are satisfiable, the perceptron terminates after a finite number of iterations (Novikoff, 1962). The number of iterations of the perceptron algorithm does not depend on the number of inequalities of (12). This property is crucial for the problems with a very large number of inequalities, which are of interest in our article.

In Step 2, the perceptron algorithm needs to find a violated inequality in (12). This subtask can be solved by computing the classifier responses $\hat{\mathbf{y}}^j = f(\mathbf{x}^j;\mathbf{w})$, $j \in \mathcal{J}$ by evaluating (10). If for some $j \in \mathcal{J}$ the response $\hat{\mathbf{y}}^j \neq \mathbf{y}^j$, that is, the classifier commits an error on the j-th example, then the indices $(j,\hat{\mathbf{y}}^j)$ identify the violated inequality. As a result, the perceptron algorithm is applicable for learning of an arbitrary linear classifier for which the classification task (10) can be evaluated efficiently.

In case of a general max-sum classifier, the classification task is NP-complete so that using the perceptron algorithm is not feasible. In Section 4 we will show, however, how to use the perceptron for learning the strictly supermodular max-sum classifiers and the max-sum classifiers with a strictly trivial equivalent.

3.2 Learning from Inconsistent Training Set Using SVM Approach

In practical applications, the training set is often inconsistent. SVMs (Vapnik, 1998) constitute a popular approach to learning linear classifiers applicable to both consistent and inconsistent training sets. The SVM learning is based on minimizing *regularized risk* which is a sum of the empirical risk $R_{\text{emp}}[f(\bullet;\mathbf{w})]$ and a regularization term. Minimization of the regularization term corresponds to the notion of large margin introduced to prevent over-fitting. Since the empirical risk is not convex for most loss functions used in classification it is replaced by a convex piece-wise linear upper bound which is more suitable for optimization. Originally, SVMs were designed for a binary case $|\boldsymbol{\mathcal{Y}}| = 2$ and $0/1$-loss function $L_{0/1}(\mathbf{y},\mathbf{y}') = [\![\mathbf{y} \neq \mathbf{y}']\!]$ where $[\![\bullet]\!]$ equals 1 if the term inside the brackets is satisfied and it is 0 otherwise. The multi-class variant of SVMs ($|\boldsymbol{\mathcal{Y}}| > 2$) were introduced in Vapnik (1998) and Crammer and Singer (2001) but it still assumes the $0/1$-loss only function.

Taskar et al. (2004b) extended SVMs for learning the max-sum classifiers (1) with an acyclic neighbourhood structure and they proposed using the additive loss function (14) suitable for these problems. Recently, the approach was generalized by Tsochantaridis et al. (2005) who consider an arbitrary structured linear classifier (10) and a general loss function. Learning of structured SVMs classifiers leads to the following convex QP task

$$(\mathbf{w}^*,\boldsymbol{\xi}^*) = \operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}} \left[\frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{j\in\mathcal{J}} \xi_j \right], \tag{13a}$$

subject to

$$\langle \mathbf{w}, \Psi(\mathbf{x}^j,\mathbf{y}^j) - \Psi(\mathbf{x}^j,\mathbf{y}) \rangle \geq L(\mathbf{y}^j,\mathbf{y}) - \xi_j, \qquad j \in \mathcal{J}, \mathbf{y} \in \boldsymbol{\mathcal{Y}}. \tag{13b}$$

The objective function of (13) comprises the regularization term $\frac{1}{2}\|\mathbf{w}\|^2$ and the sum $\frac{1}{m}\sum_{j\in\mathcal{J}}\xi_j$ weighted by the regularization constant $C > 0$. It can be shown (Tsochantaridis et al., 2005) that the sum $\frac{1}{m}\sum_{j\in\mathcal{J}}\xi_j$ is an upper bound on the empirical risk $R_{\text{emp}}[f(\bullet;\mathbf{w})]$ of the linear classifier (10). Thus the learning objective is to minimize an upper bound on the empirical risk and to penalize parameters with high $\|\mathbf{w}\|^2$. The loss function can be an arbitrary function $L\colon \boldsymbol{\mathcal{Y}} \times \boldsymbol{\mathcal{Y}} \to \mathbb{R}$ which satisfies $L(\mathbf{y},\mathbf{y}') = 0$ if $\mathbf{y} = \mathbf{y}'$ and $\infty > L(\mathbf{y},\mathbf{y}') > 0$ if $\mathbf{y} \neq \mathbf{y}'$. In cases which are of interest in our article, the set $\boldsymbol{\mathcal{Y}} = \mathcal{Y}^{\mathcal{T}}$ and a reasonable choice is the *additive loss function*, also known as the Hamming distance, which is defined as

$$L_\Delta(\mathbf{y},\mathbf{y}') = \sum_{t\in\mathcal{T}} [\![y_t \neq y'_t]\!], \tag{14}$$

that is, it counts the number of misclassified objects. The trade-off between the regularization term and the upper bound is controlled by the constant C. A suitable setting is usually found by tuning C on an independent data set or using the cross-validation.

The number of constraints (13b) equals to $m|\mathcal{Y}|$. In the structured learning $|\mathcal{Y}|$ is huge, for example, in the case of the max-sum classifiers it grows exponentially because $|\mathcal{Y}| = |\mathcal{Y}|^{|T|}$. As a result, the QP task (13) is not tractable for general purpose QP solvers. Tsochantaridis et al. (2005) proposed a specialized cutting plane algorithm to approximate (13) by a reduced QP task which has the same objective function (13a) but uses only a small subset of the constraints (13b). The efficiency of the approximation relies on the sparseness of the solution (13) which means that majority of the linear inequality constraints (13b) are inactive and they can be excluded without affecting the solution. To select the constraints, the cutting plane algorithm requires solving the LAC task

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \left[L(\mathbf{y}^j, \mathbf{y}) + \left\langle \mathbf{w}, \Psi(\mathbf{x}^j, \mathbf{y}) \right\rangle \right]. \tag{15}$$

Similarly to the perceptron algorithm, the cutting plane algorithm is applicable for learning of an arbitrary linear classifier for which the LAC task (15) can be solved efficiently.

In case of a general max-sum classifier and an additive loss function the LAC task becomes an instance of a general max-sum problem. Even though a general max-sum problem can be solved only approximately, we will show that it suffices to solve the learning problem (13) with a good precision. Moreover, we will be able to determine how the found solution differs from the optimal one. In case of a supermodular max-sum classifier we will augment the learning problem (13) in such a way that the obtained quality functions will be supermodular and thus the task (15) becomes solvable precisely in a polynomial time. Finally, in case of a max-sum problem with a strictly trivial equivalent, we will give a new formulation of the learning problem which does not require solving the task (15) at all since the number of constraints will be sufficiently small to use an exhaustive search.

4. Learning a Max-Sum Classifier from a Consistent Training Set

In this section, we assume that the training set is consistent with respect to a given max-sum classifier. This means that there exists at least one max-sum classifier in a given class which classifies all training examples correctly. We will introduce variants of the perceptron algorithm which, provided the assumption holds, find a classifier in a finite number of iterations. In practice, however, there is no general way to verify whether the assumption holds unless the perceptron halts. Consequently, if the perceptron algorithm does not halt after a reasonable number of iterations we cannot draw any conclusion about the solution. In Section 5, we will avoid this drawback using the SVM approach able to deal with inconsistent training sets.

To use the perceptron algorithm, we will reformulate the learning problem as an equivalent task of satisfying a set of linear strict inequalities. A keystone of the perceptron algorithm is an efficient procedure to find a violated inequality in the underlying set which amounts to classifying the examples from the training set. Therefore, using the perceptron algorithm is not feasible for the general max-sum problem whose evaluation is NP-complete. Nevertheless, we will formulate the learning problem even for this case because it will serve as a basis for constructing the algorithm for an inconsistent training set. Next, we will derive learning algorithms for strictly supermodular max-sum classifiers and max-sum classifiers with a strictly trivial equivalent. For these two classes, max-

sum classifiers can be evaluated efficiently by polynomial-time algorithms which makes possible to use the perceptron. Moreover, in the case of a max-sum classifier with a strictly trivial equivalent, learning will require finding a violated inequality from only a polynomially-sized set which can be accomplished exhaustively without any max-sum solver.

4.1 General Max-Sum Classifier

Problem 1 *(Learning a general max-sum classifier from a consistent training set). For a given training set* $\mathcal{L} = \{(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \mid j \in \mathcal{J}\}$ *find quality functions* $\mathbf{q}$ *and* $\mathbf{g}$ *such that*

$$\mathbf{y}^j = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^{\mathcal{T}}} F(\mathbf{x}^j, \mathbf{y}; \mathbf{q}, \mathbf{g}), \qquad j \in \mathcal{J}. \tag{16}$$

It can be seen, that the general max-sum classifier (1) can be represented as the linear classifier (10). In particular, the quality functions $\mathbf{q}$ and $\mathbf{g}$ are merged to a single parameter vector $\mathbf{w} = (\mathbf{q}; \mathbf{g}) \in \mathbb{R}^d$ of dimension $d = |\mathcal{T}||\mathcal{Y}||\mathcal{X}| + |\mathcal{E}||\mathcal{Y}|^2$. Further, we have $\boldsymbol{\mathcal{X}} = \mathcal{X}^{\mathcal{T}}$ and $\boldsymbol{\mathcal{Y}} = \mathcal{Y}^{\mathcal{T}}$. The mapping $\Psi\colon \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \to \mathbb{R}^d$ can be constructed of indicator functions in such a way that $F(\mathbf{x}, \mathbf{y}; \mathbf{q}, \mathbf{g}) = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle$.

Following the approach of learning linear classifiers described in Section 3.1, we can reformulate Problem 1 to an equivalent problem of solving a huge set of linear inequalities

$$F(\mathbf{x}^j, \mathbf{y}^j; \mathbf{q}, \mathbf{g}) > F(\mathbf{x}^j, \mathbf{y}; \mathbf{q}, \mathbf{g}), \quad j \in \mathcal{J}, \mathbf{y} \in \mathcal{Y}^{\mathcal{T}} \setminus \{\mathbf{y}^j\}. \tag{17}$$

To find a solution of (17) by the perceptron, we would need an efficient algorithm to compute a response of a general max-sum classifier. Because solving a max-sum problem is NP-complete in general, using the perceptron algorithm is not tractable. In Section 5.1, we will use the formulation (17) as a basis for constructing an algorithm for learning from an inconsistent training set. In this case, it will be possible to use the LP relaxation to approximate the response of a max-sum classifier.

4.2 Strictly Supermodular Max-Sum Classifier

Problem 2 *(Learning strictly supermodular max-sum classifier from a consistent training set). For a given training set* $\mathcal{L} = \{(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \mid j \in \mathcal{J}\}$ *find quality functions* $\mathbf{q}$ *and* $\mathbf{g}$ *such that equations (16) are satisfied and the quality functions* $\mathbf{g}$ *are strictly supermodular, that is,* $\mathbf{g}$ *satisfy the condition (9).*

Similarly to the general case, we reformulate Problem 2 as an equivalent problem of solving a set of strict linear inequalities

$$F(\mathbf{x}^j, \mathbf{y}^j; \mathbf{q}, \mathbf{g}) > F(\mathbf{x}^j, \mathbf{y}; \mathbf{q}, \mathbf{g}), \quad j \in \mathcal{J}, \mathbf{y} \in \mathcal{Y}^{\mathcal{T}} \setminus \{\mathbf{y}^j\}, \tag{18a}$$

$$\left.\begin{aligned} g_{tt'}(y, y') + g_{tt'}(y+1, y'+1) > g_{tt'}(y, y'+1) + g_{tt'}(y+1, y'), \\ \{t, t'\} \in \mathcal{E}, (y, y') \in \{1, \ldots, |\mathcal{Y}| - 1\}^2. \end{aligned}\right\} \tag{18b}$$

The system (18) comprises two sets of linear inequalities (18a) and (18b). The inequalities (18a) enforce an error-less response of a max-sum classifier on the training set. The inequalities (18b), if satisfied, guarantee that the quality functions $\mathbf{g}$ are strictly supermodular (c.f. Section 2.2).

To apply the perceptron algorithm we need an efficient method to find a violated inequality in (18). In the case of the inequalities (18b), it can be accomplished by an exhaustive search since there are only $|\mathcal{E}|(|\mathcal{Y}|-1)^2$ inequalities. To find a violated inequality in (18a), where the number $m(|\mathcal{Y}|^{|\mathcal{T}|}-1)$ grows exponentially, we need to compute the response of a max-sum classifier on the training examples. This can be done efficiently provided the inequalities (18b) are already satisfied as it guarantees that the qualities $\mathbf{g}$ are supermodular. Thus we use the perceptron algorithm to repeatedly solve the inequalities (18b) and, as soon as (18b) are satisfied, we find a violated inequality in (18a) by solving a supermodular max-sum problem. The proposed variant of the perceptron to solve (18) is as follows:

Algorithm 2 Learning strictly supermodular max-sum classifier by perceptron

1: Set $\mathbf{q} := \mathbf{0}$ and $\mathbf{g} := \mathbf{0}$.

2: Find a violated inequality in (18b), that is, find a quadruple $\{t,t'\} \in \mathcal{E}$, $y,y' \in \mathcal{Y}$ such that

$$g_{tt'}(y,y') + g_{tt'}(y+1,y'+1) - g_{tt'}(y,y'+1) - g_{tt'}(y+1,y') \leq 0\,.$$

3: If no such quadruple (t,t',y,y') exists then ($\mathbf{g}$ are already supermodular) go to Step 4. Otherwise use the found (t,t',y,y') to update current $\mathbf{g}$ by

$$\begin{array}{rclrcl} g_{tt'}(y,y') & := & g_{tt'}(y,y')+1\,, & g_{tt'}(y+1,y'+1) & := & g_{tt'}(y+1,y'+1)+1\,, \\ g_{tt'}(y,y'+1) & := & g_{tt'}(y,y'+1)-1\,, & g_{tt'}(y+1,y') & := & g_{tt'}(y+1,y')-1\,, \end{array}$$

and go to Step 2.

4: Find a violated inequality in (18a), that is, find an index $j \in \mathcal{J}$ and a labeling $\mathbf{y}$ such that

$$\mathbf{y}^j \neq \mathbf{y} := \operatorname*{argmax}_{\mathbf{y}' \in \mathcal{Y}^{\mathcal{T}}} F(\mathbf{x}^j, \mathbf{y}'; \mathbf{q}, \mathbf{g})\,.$$

5: If no such $(j, \mathbf{y})$ exist than $(\mathbf{q}, \mathbf{g})$ solve the task (18) and the algorithm halts. Otherwise use the found $(j, \mathbf{y})$ to update current $\mathbf{q}$ and $\mathbf{g}$ by

$$\begin{array}{rclrcll} g_{tt'}(y_t^j, y_{t'}^j) & := & g_{tt'}(y_t^j, y_{t'}^j)+1, & g_{tt'}(y_t, y_{t'}) & := & g_{tt'}(y_t, y_{t'})-1\,, & \{t,t'\} \in \mathcal{E}\,, \\ q_t(y_t^j, x_t^j) & := & q_t(y_t^j, x_t^j)+1, & q_t(y_t, x_t^j) & := & q_t(y_t, x_t^j)-1\,, & t \in \mathcal{T}\,. \end{array}$$

and go to Step 2.

Since Algorithm 2 is nothing but the perceptron algorithm applied to the particular set of linear constraints (18), the Novikoff's theorem (Novikoff, 1962) readily applies. Thus Algorithm 2 terminates in a finite number of iterations provided (18) is satisfiable, that is, if there exists a supermodular max-sum classifier $f(\bullet; \mathbf{q}, \mathbf{g})$ with zero empirical error risk on the training set $\mathcal{L}$.

4.3 Max-Sum Classifier with a Strictly Trivial Equivalent

Problem 3 *(Learning max-sum classifier with a strictly trivial equivalent from a consistent training set). For a given training set* $\mathcal{L} = \{(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \mid j \in \mathcal{J}\}$ *find quality functions* $\mathbf{q}$ *and* $\mathbf{g}$ *such that equations (16) are satisfied and the max-sum problems* $P^j = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x}^j)$, $j \in \mathcal{J}$ *have a strictly trivial equivalent.*

In Problem 3, we again search for a max-sum classifier which classifies all training examples correctly but, in addition, all max-sum problems P^j, $j \in \mathcal{J}$ should have a strictly trivial equivalent. This means that if we compute a response of a max-sum classifier which solves Problem 3 for each training example using LP relaxation we get a unique labeling equal to that in the training set.

Because the equivalent transformations (4) cover the whole class of equivalent problems, the problem $P = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x})$ has a strictly trivial equivalent if and only if there exist potentials $\boldsymbol{\varphi}$ such that $P^{\boldsymbol{\varphi}}$ is strictly trivial. Recall that the strictly trivial problem has unique maximal nodes and edges (cf. Definition 2). Consequently, if the problem P has a strictly trivial equivalent and its optimal solution is $\mathbf{y}^*$ then there must exist potentials $\boldsymbol{\varphi}$ such that the following set of linear inequalities holds

$$q_t^{\boldsymbol{\varphi}}(y_t^*, x_t) > q_t^{\boldsymbol{\varphi}}(y, x_t), \quad t \in \mathcal{T}, y \in \mathcal{Y} \setminus \{y_t^*\}, \tag{19a}$$

$$g_{tt'}^{\boldsymbol{\varphi}}(y_t^*, y_{t'}^*) > g_{tt'}^{\boldsymbol{\varphi}}(y, y'), \quad \{t, t'\} \in \mathcal{E}, (y, y') \in \mathcal{Y}^2 \setminus \{(y_t^*, y_{t'}^*)\}. \tag{19b}$$

Note, that (19) is a set of $|\mathcal{T}||\mathcal{Y}| + |\mathcal{E}||\mathcal{Y}|^2$ strict inequalities which are linear in $(\mathbf{q}, \mathbf{g})$ and $\boldsymbol{\varphi}$ as can be seen after substituting (4) to (19). By replicating (19) for max-sum problems $P^j = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x}^j)$, $j \in \mathcal{J}$ whose unique solutions are required to be $\mathbf{y}^j$, $j \in \mathcal{J}$ we obtain an equivalent formulation of Problem 3 which requires satisfaction of a set of strict linear inequalities

$$q_t^{\boldsymbol{\varphi}^j}(y_t^j, x_t^j) > q_t^{\boldsymbol{\varphi}^j}(y, x_t^j), \quad j \in \mathcal{J}, t \in \mathcal{T}, y \in \mathcal{Y} \setminus \{y_t^j\}, \tag{20a}$$

$$g_{tt'}^{\boldsymbol{\varphi}^j}(y_t^j, y_{t'}^j) > g_{tt'}^{\boldsymbol{\varphi}^j}(y, y'), \quad j \in \mathcal{J}, \{t, t'\} \in \mathcal{E}, (y, y') \in \mathcal{Y}^2 \setminus \{(y_t^j, y_{t'}^j)\}. \tag{20b}$$

The system (20) is to be solved with respect to the quality functions $(\mathbf{q}, \mathbf{g})$ and the potentials $\boldsymbol{\varphi}^j$, $j \in \mathcal{J}$ introduced for each P^j, $j \in \mathcal{J}$. Note that the number of inequalities in the problem (20) is substantially smaller compared to the learning of a general max-sum classifier. In particular, (20) contains only $m|\mathcal{T}|(|\mathcal{Y}|-1) + m|\mathcal{E}|(|\mathcal{Y}|^2-1)$ inequalities compared to $m(|\mathcal{Y}|^{|\mathcal{T}|}-1)$ for a general max-sum classifier. Therefore a violated inequality in (20) can be easily selected by an exhaustive search. The proposed variant of the perceptron to solve (20) is as follows:

Algorithm 3 Learning the max-sum classifier with a strictly equivalent by perceptron

1: Set $\mathbf{g} := 0$, $\mathbf{q} := 0$, $\boldsymbol{\varphi}^j := 0$, $j \in \mathcal{J}$.

2: Find a violated inequality in (20a), that is, find a triplet $j \in \mathcal{J}$, $t \in \mathcal{T}$, $y \in \mathcal{Y} \setminus \{y_t^j\}$ such that

$$q_t(y_t^j, x_t^j) - \sum_{t' \in \mathcal{N}(t)} \varphi_{tt'}^j(y_t^j) \le q_t(y, x_t^j) - \sum_{t' \in \mathcal{N}(t)} \varphi_{tt'}^j(y).$$

3: If no such triplet (j, t, y) exists then go to Step 4. Otherwise update $\mathbf{q}$ and $\boldsymbol{\varphi}^j$ by

$$\begin{aligned} \varphi_{tt'}^j(y_t^j) &:= \varphi_{tt'}^j(y_t^j) - 1, \quad \varphi_{tt'}^j(y) := \varphi_{tt'}^j(y) + 1, \quad t' \in \mathcal{N}(t), \\ q_t(y_t^j, x_t^j) &:= q_t(y_t^j, x_t^j) + 1, \quad q_t(y, x_t^j) := q_t(y, x_t^j) - 1. \end{aligned}$$

4: Find a violated inequality in (20b), that is, find a quintuple $j \in \mathcal{J}$, $\{t,t'\} \in \mathcal{E}$, $(y,y') \in \mathcal{Y}^2 \setminus \{(y^j_t, y^j_{t'})\}$ such that

$$g_{tt'}(y^j_t, y^j_{t'}) + \varphi^j_{tt'}(y^j_t) + \varphi^j_{t't}(y^j_{t'}) \leq g_{tt'}(y,y') + \varphi^j_{tt'}(y) + \varphi^j_{t't}(y') \,.$$

5: If no such quintuple (j,t,t',y,y') exists and no update was made in Step 3 then the current $(\mathbf{q}, \mathbf{g}, \boldsymbol{\varphi}^j, j \in \mathcal{J})$ solves the task (20) and the algorithm halts. Otherwise update $\mathbf{g}$ and $\boldsymbol{\varphi}^j$ by

$$\begin{array}{rcrrcr} \varphi^j_{tt'}(y^j_t) & := & \varphi^j_{tt'}(y^j_t) + 1 \,, & \varphi^j_{t't}(y^j_{t'}) & := & \varphi^j_{t't}(y^j_{t'}) + 1 \,, \\ \varphi^j_{tt'}(y) & := & \varphi^j_{tt'}(y) - 1 \,, & \varphi^j_{t't}(y') & := & \varphi^j_{t't}(y') - 1 \,, \\ g_{tt'}(y^j_t, y^j_{t'}) & := & g_{tt'}(y^j_t, y^j_{t'}) + 1 \,, & g_{tt'}(y,y') & := & g_{tt'}(y,y') - 1 \,. \end{array}$$

and go to Step 2.

By the Novikoff's theorem, Algorithm 3 terminates in a finite number of iterations provided (20) is satisfiable, that is, if there exists a max-sum classifier $f(\bullet; \mathbf{q}, \mathbf{g})$ with zero empirical risk on $\mathcal{L}$ and all max-sum problems $P^j = (\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x}^j)$, $j \in \mathcal{J}$ have a strictly trivial equivalent.

5. Learning a Max-Sum Classifier from an Inconsistent Training Set

In this section, we formulate problems of learning max-sum classifiers from an inconsistent training set using the SVM framework described in Section 3.2. We will formulate the learning problems for a general max-sum classifier, a supermodular max-sum classifier, and a max-sum classifier with a strictly trivial equivalent. In all cases, learning will amount to solving an instance of a convex QP task. In Section 6, we will propose an extended version of the cutting plane algorithm (Tsochantaridis et al., 2005) which can solve these QP tasks efficiently.

In Section 4, we showed that learning of max-sum classifiers can be expressed as satisfying a set of strict linear inequalities. In the case of an inconsistent training set, however, these linear inequalities become unsatisfiable. Therefore we augment linear inequalities with non-negative slack variables, which relaxes the problem and makes it always satisfiable. In analogy to the SVM, we will minimize the Euclidean norm of an optimized parameters and the sum of slack variables. The problems are formulated in such a way that the sum of slack variables is a piece-wise linear upper bound on the empirical risk for a certain loss function. We will consider two different loss functions. In case of a general max-sum classifier and a supermodular max-sum classifier, we will use the *additive loss function* $L_\Delta(\mathbf{y}, \mathbf{y}') = \sum_{t \in \mathcal{T}} L_t(y_t, y'_t)$ where $L_t \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is any function which satisfies $L_t(y,y') = 0$ for $y = y'$ and $L_t(y,y') > 0$ otherwise. The Hamming distance $L_\Delta(\mathbf{y}, \mathbf{y}') = \sum_{t \in \mathcal{T}} [\![\mathbf{y} \neq \mathbf{y}']\!]$ is the particular case of the additive loss which seems to be a reasonable choice in many (not necessarily all) applications. In case of a max-sum classifier with a strictly trivial equivalent, we will consider the *0/1-loss function* $L_{0/1}(\mathbf{y}, \mathbf{y}') = [\![\mathbf{y} \neq \mathbf{y}']\!]$. The 0/1-loss function penalizes equally all incorrect predictions regardless of how many labels are misclassified. The additive loss is preferable to the 0/1-loss function in most structured classification problems. On the other hand, there are applications for which the 0/1-loss function is a natural choice, for example, problems with small number of objects like the one presented in Section 7.2.

5.1 General Max-Sum Classifier

Problem 4 *(Learning a general max-sum classifier from an inconsistent training set). For a given training set* $\mathcal{L} = \left\{(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \,\middle|\, j \in \mathcal{J}\right\}$ *and a regularization constant* C *find quality functions* $\mathbf{q}$ *and* $\mathbf{g}$ *solving the following QP task*

$$(\mathbf{g}^*, \mathbf{q}^*, \boldsymbol{\xi}^*) = \operatorname*{argmin}_{\mathbf{g},\mathbf{q},\boldsymbol{\xi}} \left[\frac{1}{2}\left(\|\mathbf{g}\|^2 + \|\mathbf{q}\|^2\right) + \frac{C}{m}\sum_{j\in\mathcal{J}} \xi_j\right], \tag{21a}$$

subject to

$$F(\mathbf{x}^j, \mathbf{y}^j; \mathbf{q}, \mathbf{g}) - F(\mathbf{x}^j, \mathbf{y}; \mathbf{q}, \mathbf{g}) \geq L_\Delta(\mathbf{y}^j, \mathbf{y}) - \xi_j\,, \quad j \in \mathcal{J}, \mathbf{y} \in \mathcal{Y}^{\mathcal{T}}\,. \tag{21b}$$

The QP task (21) fits to the formulation (13) of structured SVM learning with the max-sum classifier (1) plugged in. The number of optimized parameters $(\mathbf{q};\mathbf{g}) \in \mathbb{R}^d$ equals to $d = |\mathcal{X}||\mathcal{Y}||\mathcal{T}| + |\mathcal{Y}|^2|\mathcal{E}|$. The main difficulty in solving (21) stems from the huge number $n = m|\mathcal{Y}|^{|\mathcal{T}|}$ of the linear inequalities (21b) which define the feasible set.

For a later use, it is convenient to rewrite the QP task (21) using a compact notation

$$\begin{aligned} (\mathbf{w}^*, \boldsymbol{\xi}^*) &= \operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}} Q_P(\mathbf{w}, \boldsymbol{\xi}) = \operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}} \left[\frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{j\in\mathcal{J}} \xi_j\right], \\ \text{s.t.} \quad & \langle \mathbf{w}, \mathbf{z}_i \rangle \geq b_i - \xi_j\,, \quad j \in \mathcal{J}, i \in I_j\,, \end{aligned} \tag{22}$$

where $I_1 \cup \cdots \cup I_m = I = \{1, \ldots, n\}$, $I_u \cap I_v = \{\emptyset\}$, $u \neq v$, denote disjoint sets of indices such that each $i \in I$ has assigned an unique pair $(j, \mathbf{y})$, $j \in \mathcal{J}$, $\mathbf{y} \in \mathcal{Y}^{\mathcal{T}}$; the vector $\mathbf{w} = (\mathbf{q};\mathbf{g}) \in \mathbb{R}^d$ comprises both the optimized quality functions and $(\mathbf{z}_i \in \mathbb{R}^d, b_i \in \mathbb{R})$, $i \in I$, are constructed correspondingly to inscribe the inequalities (21b).

In Section 6, we will introduce a variant of the cutting plane algorithm to solve the QP task (21) (or (22), respectively). The cutting plane algorithm requires a subroutine which solves the LAC task (15). Using the compact notation, the LAC task reads $u_j = \operatorname{argmax}_{i\in I_j}(b_i - \langle \mathbf{w}, \mathbf{z}_i \rangle)$. Since this is NP-complete in general, we will use an LP relaxation which solves the task approximately. We will show that it is possible to assess the quality of the found solution $(\mathbf{w}, \boldsymbol{\xi})$ in terms of the objective of the learning task, that is, it is possible to bound the difference $Q_P(\mathbf{w}, \boldsymbol{\xi}) - Q_P(\mathbf{w}^*, \boldsymbol{\xi}^*)$. Further, we will prove that when the LAC task is solved exactly then the cutting plane algorithm finds an arbitrary precise solution $Q_P(\mathbf{w}, \boldsymbol{\xi}) - Q_P(\mathbf{w}^*, \boldsymbol{\xi}^*) \leq \varepsilon$, $\varepsilon > 0$, after a finite number of iterations. This guarantee does not apply for learning of the general max-sum classifier when an LP relaxation providing only an approximate solution is used. Though there is no theoretical guarantee, we will experimentally show that using an LP relaxation is sufficient to find a practically useful solution.

5.2 Supermodular Max-Sum Classifier

Problem 5 *(Learning a supermodular max-sum classifier from an inconsistent training set). For a given training set* $\mathcal{L} = \left\{(\mathbf{x}^j, \mathbf{y}^j) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \,\middle|\, j \in \mathcal{J}\right\}$ *and a regularization constant* C *find quality functions* $\mathbf{q}$ *and* $\mathbf{g}$ *solving the following QP task*

$$(\mathbf{g}^*, \mathbf{q}^*, \boldsymbol{\xi}^*) = \operatorname*{argmin}_{\mathbf{g},\mathbf{q},\boldsymbol{\xi}} \left[\frac{1}{2}\left(\|\mathbf{g}\|^2 + \|\mathbf{q}\|^2\right) + \frac{C}{m}\sum_{j\in\mathcal{J}} \xi_j\right], \tag{23a}$$

subject to

$$F(\mathbf{x}^j,\mathbf{y}^j;\mathbf{q},\mathbf{g}) - F(\mathbf{x}^j,\mathbf{y};\mathbf{q},\mathbf{g}) \geq L_\Delta(\mathbf{y}^j,\mathbf{y}) - \xi_j\,, \quad j \in \mathcal{J}, \mathbf{y} \in \mathcal{Y}^\mathcal{T}\,, \tag{23b}$$

$$\left.\begin{aligned} g_{tt'}(y,y') + g_{tt'}(y+1,y'+1) \geq g_{tt'}(y,y'+1) + g_{tt'}(y+1,y')\,, \\ \{t,t'\} \in \mathcal{E}\,, (y,y') \in \{1,\ldots,|\mathcal{Y}|-1\}^2\,. \end{aligned}\right\} \tag{23c}$$

Compared to the task (21) of learning a general max-sum classifier, the task (23) defined for the supermodular max-sum classifier contains additional linear constraints (23c). The added constraints (23c), when satisfied, guarantee that the found quality function $\mathbf{g}$ is supermodular. The total number of constraints increases to $n = m|\mathcal{Y}|^{|\mathcal{T}|} + |\mathcal{E}|(|\mathcal{Y}|-1)^2$. A compact form of the QP task (23) reads

$$\begin{aligned} (\mathbf{w}^*,\boldsymbol{\xi}^*) = \operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}} & \left[\frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{j\in\mathcal{J}}\xi_j\right], \\ \text{s.t.} \quad \langle \mathbf{w},\mathbf{z}_i\rangle &\geq b_i\,, \qquad i \in I_0\,, \\ \langle \mathbf{w},\mathbf{z}_i\rangle &\geq b_i - \xi_j\,, \quad j \in \mathcal{J}\,, i \in I_j\,, \end{aligned} \tag{24}$$

where $b_i = 0$, $i \in I_0$, and $\mathbf{z}_i$, $i \in I_0$, account for the added supermodular constraints (23c).

In Section 6, we introduce a variant of the cutting plane algorithm which maintains the constraints (23c) satisfied. Thus the quality functions are always supermodular and the LAC task can be solved precisely by efficient polynomial-time algorithms. As a result, we can guarantee that the cutting plane algorithm finds a solution with an arbitrary finite precision in a finite number of iterations.

5.3 Max-sum Classifier with Strictly Trivial Equivalent

Problem 6 *(Learning a max-sum classifier with a strictly trivial equivalent from an inconsistent training set). For a given training set* $\mathcal{L} = \{(\mathbf{x}^j,\mathbf{y}^j) \in \mathcal{X}^\mathcal{T} \times \mathcal{Y}^\mathcal{T} \,\big|\, j \in \mathcal{J}\}$ *and regularization constant* C *find quality functions* $\mathbf{q}$ *and* $\mathbf{g}$ *that solve the following QP task*

$$(\mathbf{g}^*,\mathbf{q}^*,\boldsymbol{\xi}^*,\boldsymbol{\varphi}^{j*}, j\in\mathcal{J}) = \operatorname*{argmin}_{\mathbf{g},\mathbf{q},\boldsymbol{\xi},\boldsymbol{\varphi}^j, j\in\mathcal{J}} \left[\frac{1}{2}\big(\|\mathbf{g}\|^2 + \|\mathbf{q}\|^2 + \sum_{j\in\mathcal{J}}\|\boldsymbol{\varphi}^j\|^2\big) + \frac{C}{m}\sum_{j\in\mathcal{J}}\xi_j\right], \tag{25a}$$

subject to

$$\begin{aligned} q_t^{\boldsymbol{\varphi}^j}(y_t^j,x_t^j) - q_t^{\boldsymbol{\varphi}^j}(y,x_t^j) &\geq 1-\xi_j\,, \quad j\in\mathcal{J}\,, t\in\mathcal{T}\,, y\in\mathcal{Y}\setminus\{y_t^j\}\,, \\ g_{tt'}^{\boldsymbol{\varphi}^j}(y_t^j,y_{t'}^j) - g_{tt'}^{\boldsymbol{\varphi}^j}(y,y') &\geq 1-\xi_j\,, \quad j\in\mathcal{J}\,, \{t,t'\}\in\mathcal{E}\,, (y,y')\in\mathcal{Y}^2\setminus\{(y_t^j,y_{t'}^j)\}\,, \\ \xi_j &\geq 0\,, \qquad j\in\mathcal{J}\,. \end{aligned} \tag{25b}$$

The problem (25) is derived from the problem (20) which was formulated for a consistent training set. In the consistent case, it is required that each max-sum problem P^j has a strictly trivial equivalent whose unique solution equals to the desired labeling $\mathbf{y}^j$ given in the training set. In the case of an inconsistent training set, we allow some max-sum problems to violate this requirement. To this end, each subset of linear inequalities in (25b) which corresponds to the given max-sum problem P^j is relaxed by a single non-negative slack variable ξ_j. If a slack variable $\xi_j \geq 1$ then either the solution of the max-sum problem P^j differs from $\mathbf{y}^j$ or P^j has no trivial equivalent. The number of

such max-sum problems is upper bounded by $\sum_{j\in\mathcal{J}}^{m}\xi_j$ which is included into the objective function of (25a). Thus it corresponds to minimization of an upper bound on the empirical risk with the $0/1$-loss functions. The objective function also contains the Euclidean norm of the quality functions $(\mathbf{q},\,\mathbf{g})$ and the potentials $\boldsymbol{\varphi}^j$. Including the potentials $\boldsymbol{\varphi}^j$ into the objective function is somewhat arbitrary since it penalizes the transformation of max-sum problems to their trivial equivalents. An advantage of including the potentials is the fact that the dual representation of the task (25) has a simpler form which corresponds to the QP task of an ordinary multi-class SVM. Another variant is to remove the potentials from the objective function which corresponds to including a set of linear constraints to the dual task of (25) making the problem more difficult.

Compared to the previous learning problems, the number of variables $d=|\mathcal{X}||\mathcal{Y}||\mathcal{T}|+|\mathcal{Y}|^2|\mathcal{E}|+2m|\mathcal{E}||\mathcal{Y}|$ in (25) is increased by $2m|\mathcal{E}||\mathcal{Y}|$, however, the number of linear constraints $n=m|\mathcal{E}|(|\mathcal{Y}|^2-1)+m|\mathcal{T}|(|\mathcal{Y}|-1)+m$ is drastically smaller. In particular, n grows only polynomially compared to the exponential growth in the previous cases.

The QP task (25) can be rewritten into the compact form (22), that is, the same QP task as required when learning the general max-sum classifier. Unlike the general case, the optimized parameters $\mathbf{w}$ now comprise $\mathbf{w}=(\mathbf{q};\mathbf{g};\boldsymbol{\varphi}^1;\ldots;\boldsymbol{\varphi}^m)\in\mathbb{R}^d$ and the vectors $\mathbf{z}_i$, $i\in I$, are constructed correspondingly. However, as mentioned above, the main difference is much smaller $n=|I|$. As a result, the LAC task required by the cutting plane algorithm can be easily accomplished by an exhaustive search.

6. Algorithm for Quadratic Programming Tasks

In this section, we propose an algorithm to solve the QP tasks (21), (23) and (25) required for learning the max-sum classifiers from inconsistent training sets. We extend the cutting plane algorithm by Tsochantaridis et al. (2005) and its approximate version by Finley and Joachims (2005) in two directions. First, we will consider a more general QP task (23) which contains linear inequalities both with and without slack variables. Moreover, the inequalities without slack variables are required to be satisfied during the whole course of the algorithm. This extension is necessary for learning a supermodular max-sum classifier. Second, we propose to use a different stopping conditions to halt the algorithm. The original algorithm by Tsochantaridis et al. (2005) halts the optimization as soon as the linear constraints of a QP task are violated by a prescribed constant $\bar{\varepsilon}>0$ at most. We will use a stopping condition which is based on the duality gap. This allows us to control the precision of the found solution directly in terms of the optimized objective function. Moreover, the stopping condition can be easily evaluated even when the LAC task is solved only approximately by an LP relaxation which is useful for learning general max-sum classifiers. Finally, we will prove that the proposed algorithm converges in a finite number of iterations even in the case of a general max-sum classifier where the LAC task is NP-complete. We point out that the proof is similar to that of Tsochantaridis et al. (2005) but is technically simpler and it applies for the extended cutting plane algorithm proposed here. A general QP task which covers all the QP tasks (21), (23) and (25) reads

$$
\begin{aligned}
&(\mathbf{w}^*,\boldsymbol{\xi}^*)=\operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}} Q_P(\mathbf{w},\boldsymbol{\xi})=\operatorname*{argmin}_{\mathbf{w},\boldsymbol{\xi}}\left[\frac{1}{2}\|\mathbf{w}\|^2+\frac{C}{m}\sum_{j\in\mathcal{J}}\xi_j\right],\\
&\text{s.t.}\qquad \begin{array}{lll}\langle\mathbf{w},\mathbf{z}_i\rangle & \geq & b_i\,, \qquad i\in I_0\,,\\ \langle\mathbf{w},\mathbf{z}_i\rangle & \geq & b_i-\xi_j\,, \quad j\in\mathcal{J}, i\in I_j\,,\end{array}
\end{aligned}
\tag{26}
$$

where $I = I_0 \cup I_1 \cup \cdots \cup I_m = \{1,\ldots,n\}$, $I_i \cap I_j = \{\emptyset\}$, $i \neq j$ are index sets. Note that we obtain the QP task (21) and (25) by setting $I_0 = \{\emptyset\}$. The Wolf dual of (26) reads

$$\begin{aligned} \boldsymbol{\alpha}^* &= \operatorname*{argmax}_{\boldsymbol{\alpha}} Q_D(\boldsymbol{\alpha}) = \operatorname*{argmax}_{\boldsymbol{\alpha}} \left[\langle \mathbf{b}, \boldsymbol{\alpha} \rangle - \frac{1}{2} \langle \boldsymbol{\alpha}, \mathbf{H}\boldsymbol{\alpha} \rangle \right], \\ \text{s.t.} \quad \sum_{i \in I_j} \alpha_i &= \frac{C}{m}, \quad j \in \mathcal{J}, \\ \alpha_i &\geq 0, \quad i \in I, \end{aligned} \tag{27}$$

where $\mathbf{H}$ is a positive semi-definite matrix $[n \times n]$ which contains the dot products $H_{i,j} = \langle \mathbf{z}_i, \mathbf{z}_j \rangle$. The primal variables $(\mathbf{w}^*, \boldsymbol{\xi}^*)$ can be computed from the dual variables $\boldsymbol{\alpha}^*$ by

$$\mathbf{w}^* = \sum_{i \in I} \alpha_i^* \mathbf{z}_i, \quad \xi_j^* = \max_{i \in I_j} (b_i - \langle \mathbf{w}^*, \mathbf{z}_i \rangle), \ j \in \mathcal{J}.$$

Note that the vectors $\mathbf{z}_i$, $i \in I$ appear in (27) only as dot products and thus the kernel functions can be potentially used.

It is not tractable to solve the QP task directly neither in the primal form nor in the dual form due to the exponential number of constraints or variables, respectively. The cutting plane algorithm alleviates the problem by exploiting the sparseness of maximal margin classifiers. Sparseness means that only a small portion of constraints of the primal task are active in the optimal solution, or equivalently, a small portion of dual variables are non-zero.

We denote $\overline{I} = I_0 \cup \overline{I}_1 \cup \cdots \cup \overline{I}_m$ disjoint subsets of selected indices from $I = I_0 \cup I_1 \cup \cdots \cup I_m$, that is, $\overline{I}_j \subseteq I_j$, $j \in \mathcal{J}$, while I_0 is always the same. The reduced task is obtained from (27) by considering only the selected variables $\{\alpha_i \mid i \in \overline{I}\}$ while the remaining variables $\{\alpha_i \mid i \in I \setminus \overline{I}\}$ are fixed to zero, that is, the reduced dual QP task reads

$$\overline{\boldsymbol{\alpha}} = \operatorname*{argmax}_{\boldsymbol{\alpha}} \left[\sum_{i \in \overline{I}} \alpha_i b_i - \frac{1}{2} \sum_{i \in \overline{I}} \sum_{j \in \overline{I}} \alpha_i \alpha_j H_{ij} \right], \tag{28a}$$

$$\begin{aligned} \text{s.t.} \quad \sum_{i \in \overline{I}_j} \alpha_i &= \frac{C}{m}, \quad j \in \mathcal{J}, \\ \alpha_i &\geq 0, \quad i \in \overline{I}, \\ \alpha_i &= 0, \quad i \in I \setminus \overline{I}. \end{aligned} \tag{28b}$$

To solve the reduced task (28), it is enough to maintain explicitly just the selected variables and thus its size scales only with $|\overline{I}|$. The cutting plane algorithm tries to select the subsets $\overline{I} = I_0 \cup \overline{I}_1 \cup \cdots \cup \overline{I}_m$ such that (i) $|\overline{I}|$ is sufficiently small to make the reduced task (28) solvable by standard optimization packages and (ii) the obtained solution well approximates the original task. The proposed extension of the cutting plane algorithm is as follows:

Algorithm 4 A cutting plane algorithm for QP task (26)

1: Select arbitrarily $\left(\overline{I}_j := \{u_j\}, u_j \in I_j\right)$, $j \in \mathcal{J}$. Set the desired precision $\varepsilon > 0$.

2: For selected indices $\overline{I} = I_0 \cup \overline{I}_1 \cup \cdots \cup \overline{I}_m$ solve the reduced task (28) to obtain $\overline{\boldsymbol{\alpha}}$ and compute the primal variable $\overline{\mathbf{w}} := \sum_{i \in \overline{I}} \overline{\alpha}_i \mathbf{z}_i$.

3: For all $j \in \mathcal{J}$ do:

3a: Solve the LAC task

$$u_j := \operatorname*{argmax}_{i \in I_j} \left(b_i - \langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle \right) , \tag{29}$$

and set $\overline{\xi}_j := b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle$. If (29) is solved only approximately (when using LP relaxation) then set $\overline{\xi}_j$ to the smallest available upper bound on $b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle$.

3b: Set $\overline{I}_j := \overline{I}_j \cup \{u_j\}$ provided the following condition holds

$$\frac{C}{m}(b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle) - \sum_{i \in \overline{I}_j} \overline{\alpha}_i (b_i - \langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle) > \frac{\varepsilon}{m} . \tag{30}$$

4: If the current solution $(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\alpha}})$ satisfies the stopping condition

$$Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_D(\overline{\boldsymbol{\alpha}}) \leq \varepsilon , \tag{31}$$

or if the condition (30) was violated for all $j \in \mathcal{J}$ then halt. Otherwise go to Step 2.

Note that Algorithm (4) explicitly maintains only the selected variables $\{\alpha_i \mid i \in \overline{I}\}$ and the corresponding pairs $\{(b_i, \mathbf{z}_i) \mid i \in \overline{I}\}$. In Step 3a, the algorithm requires a subroutine to solve the LAC task (29). We consider two different variants of the algorithm. First, the task (29) can be solved precisely which applies to learning of supermodular max-sum classifiers and max-sum classifiers with a strictly trivial equivalent. Second, the task (29) can be solved only approximately, which applies to the general case when the max-sum problem is solved by an LP relaxation. The key property of an LP relaxation which we exploit here is that it provides an upper bound on the optimal solution of a max-sum problem, that is, an upper bound on the quantity $b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle$.

In Step 3b, the condition (30) is evaluated to decide whether adding the constraint $\langle \mathbf{w}, \mathbf{z}_{u_j} \rangle \geq b_{u_j} - \xi_j$ to the reduced QP task (28) brings a sufficient improvement or not. In Lemma 1 introduced below, we show that adding at least one constraint guarantees a minimal improvement regardless of whether the LAC problem is solved precisely or approximately. The algorithm halts when no constraint is added in Step 3b. This situation occurs only if the algorithm has converged to a solution which satisfies the stopping condition (31) provided the LAC task is solved exactly (cf. Theorem 4 and its proof).

Let us discuss the stopping condition (31). Note that the primal $(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}})$ and the dual variables $\overline{\boldsymbol{\alpha}}$ are feasible during entire progress of the algorithm. This holds even in the case when an LP relaxation is used to solve (29) since the $\overline{\xi}_j$, $j \in \mathcal{J}$ are set to upper bounds on their optimal values. Having feasible primal and dual variables, we can use the weak duality theorem to write

$$Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_P(\mathbf{w}^*, \boldsymbol{\xi}^*) \leq Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_D(\overline{\boldsymbol{\alpha}}) ,$$

which implies that any solution satisfying the stopping condition (31) differs from the optimal solution by at most ε. Note that the precision parameter $\bar{\varepsilon}$ used in the original stopping condition of Tsochantaridis et al. (2005) can also be related to the duality gap. Namely, it can be shown that $\bar{\varepsilon}$ approximate solution satisfies $Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_D(\overline{\boldsymbol{\alpha}}) \leq C\bar{\varepsilon}$.

Finally, we show that the algorithm converges in a finite number of iterations. The proof is based on Lemma 1 which asserts that a minimal improvement $\Delta_{\min} > 0$ of the dual objective function is guaranteed provided at least one new variable was added in Step 3(b) to the reduced task, that is, the condition (30) was satisfied for at least one $j \in \mathcal{J}$.

Lemma 1 *Provided at least one new variable was added in Step 3(b) of Algorithm 4 the improvement in the dual objective function $Q_D(\boldsymbol{\alpha})$ obtained after solving the reduced task is not less that*

$$\Delta_{\min} = \min\left\{\frac{\varepsilon}{2m}, \frac{\varepsilon^2}{8C^2R^2}\right\}, \quad \textit{where} \quad R = \max_{j\in\mathcal{J}} \max_{i\in I_j} \|\mathbf{z}_i\|\,.$$

Proof of Lemma 1 is given in Appendix B.

Using Lemma 1, it is easy to show that Algorithm 4 halts after a finite number of iterations. Note that the proof applies even for the case when the LAC task (29) is solved only approximately.

Theorem 4 *Let us assume that the primal QP task (26) is bounded and feasible. Algorithm 4 halts for arbitrary $\varepsilon > 0$ after at most T iterations, where*

$$T = \left(Q_D(\boldsymbol{\alpha}^*) - Q_D(\overline{\boldsymbol{\alpha}}^1)\right) \max\left\{\frac{2m}{\varepsilon}, \frac{8C^2R^2}{\varepsilon^2}\right\}, \qquad R = \max_{j\in\mathcal{J}} \max_{i\in I_j} \|\mathbf{z}_i\|\,,$$

and $\overline{\boldsymbol{\alpha}}^1$ is the solution of the reduced task obtained after the first iteration of Algorithm 4.

Proof Algorithm 4 halts if either the stopping condition (31) holds or no new variable was added in Step 3(b). Provided the algorithm does not halt in Step 4, the dual objective functions $Q_D(\boldsymbol{\alpha})$ is improved by at least $\Delta_{\min} > 0$ which is given in Lemma 1. Since the difference $Q_D(\boldsymbol{\alpha}^*) - Q_D(\overline{\boldsymbol{\alpha}}^1)$ is bounded from above the algorithm cannot pass through the Step 4 infinite number times and we can write

$$T \leq \frac{Q_D(\boldsymbol{\alpha}^*) - Q_D(\overline{\boldsymbol{\alpha}}^1)}{\Delta_{\min}} = \left(Q_D(\boldsymbol{\alpha}^*) - Q_D(\overline{\boldsymbol{\alpha}}^1)\right) \max\left\{\frac{2m}{\varepsilon}, \frac{8C^2R^2}{\varepsilon^2}\right\} < \infty\,.$$

■

The bound on a maximal number of iterations (or the bound on $\Delta_{\min}$, respectively) given in Theorem 4 is obtained based on the worst case analysis. As a result, the bound is an over-pessimistic estimate of the number of iterations usually required in practice. Because the bound is not useful for computing practical estimates of the number of iterations, we do not derive its particular variants for the QP tasks (21), (23) and (25).

Finally, we give a theorem which asserts that the stopping condition (31) is always satisfied after Algorithm 4 halts provided the LAC task (29) is solved exactly, that is, the found solution achieves the desired precision $Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_P(\mathbf{w}^*, \boldsymbol{\xi}^*) \leq \varepsilon$.

Theorem 5 *Let us assume that the LAC task (29) is solved exactly and the assumptions of Theorem 4 hold. In this case the condition (31) holds after Algorithm 4 halts.*

Proof We show that the assumption that no new variable was added in Step 3(b) and the condition (31) is violated leads to a contradiction. If no variable was added then the condition (30) is violated for all $j \in \mathcal{J}$. Summing up the violated conditions (30) yields

$$\sum_{j\in\mathcal{J}} \left[\frac{C}{m}\left(b_{u_j} - \langle\overline{\mathbf{w}}, \mathbf{z}_{u_j}\rangle\right) - \sum_{i\in\overline{I}_j} \overline{\alpha}_i \left(b_i - \langle\overline{\mathbf{w}}, \mathbf{z}_i\rangle\right)\right] \leq \sum_{j\in\mathcal{J}} \frac{\varepsilon}{m}\,. \tag{32}$$

The vector $\overline{\boldsymbol{\alpha}}$ is the optimal solution of the reduced task which includes all the variables $\{\alpha_i \,|\, i \in I_0\}$. Therefore by the complementary slackness (Karush-Kuhn-Tucker conditions) we have

$$\sum_{i \in I_0} \overline{\alpha}_i (b_i - \langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle) = 0 \,. \tag{33}$$

By adding (33) to (32) and using $\overline{\mathbf{w}} = \sum_{i \in \overline{I}} \overline{\alpha}_i \mathbf{z}_i$, $\overline{\xi}_j = b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle$ we get

$$\begin{aligned}
\sum_{j \in \mathcal{J}} \left[\frac{C}{m} \left(b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle \right) - \sum_{i \in \overline{I}_j} \overline{\alpha}_i \left(b_i - \langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle \right) \right] - \sum_{i \in \overline{I}_0} \overline{\alpha}_i \left(b_i - \langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle \right) &\leq \varepsilon \\
\frac{C}{m} \sum_{j \in \mathcal{J}} \left(b_{u_j} - \langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle \right) - \sum_{i \in \overline{I}} \overline{\alpha}_i \left(b_i - \langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle \right) &\leq \varepsilon \\
\frac{C}{m} \sum_{j \in \mathcal{J}} \overline{\xi}_j - \sum_{i \in \overline{I}} \overline{\alpha}_i b_i + \langle \overline{\mathbf{w}}, \overline{\mathbf{w}} \rangle &\leq \varepsilon \,.
\end{aligned}$$

The last inequality can be equivalently written as $Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_D(\overline{\boldsymbol{\alpha}}) \leq \varepsilon$ which is in contradiction to the assumption that (31) is violated. ■

To summarize, in case when the LAC task (29) is solved exactly Algorithm 4 finds a solution with an arbitrary finite precision $\varepsilon > 0$ in a finite number of iterations. In case when the task (29) is solved only approximately by an LP relaxation Algorithm 4 always halts after a finite number of iterations but there is no guarantee that the desired precision was attained. The attained precision, however, can be determined by evaluating the duality gap $Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_D(\overline{\boldsymbol{\alpha}}) \geq Q_P(\overline{\mathbf{w}}, \overline{\boldsymbol{\xi}}) - Q_P(\mathbf{w}^*, \boldsymbol{\xi}^*)$.

In next sections, we will apply Algorithm 4 to the three particular instances of the QP tasks (21), (23) and (25).

6.1 General Max-Sum Classifier

In this section, we discuss optimization of the QP task (21) (or its compact form (22), respectively) using Algorithm 4.

In Step 1 of Algorithm 4, we have to select the initial subsets $(\overline{I}_j = \{u_j\}, u_j \in I_j), j \in \mathcal{J}$. A simple way is to use $u_j = (j, \mathbf{y}^j)$, which is equivalent to selecting the primal constraints $\langle \mathbf{w}, \mathbf{z}_i \rangle \geq b_i - \xi_j$ with $\mathbf{z}_i = \mathbf{0}$ and $b_i = 0$. Note that the task (21) does not contain the primal constraints without slack variables which implies that the subset $I_0 = \{\emptyset\}$.

The LAC task (29) required in Step 2a of Algorithm 4 amounts to solving an instance of a general max-sum problem

$$\begin{aligned}
\hat{\mathbf{y}}^j &= \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^{\mathcal{T}}} \left[L_\Delta(\mathbf{y}, \mathbf{y}^j) - F(\mathbf{x}^j, \mathbf{y}^j; \mathbf{q}, \mathbf{g}) + F(\mathbf{x}^j, \mathbf{y}; \mathbf{q}, \mathbf{g}) \right] \\
&= \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^{\mathcal{T}}} \left[\sum_{t \in \mathcal{T}} \left(q_t(y_t, x_t^j) + [\![y_t^j \neq y_t]\!] \right) + \sum_{\{t,t'\} \in \mathcal{E}} g_{tt'}(y_t, y_{t'}) \right] .
\end{aligned} \tag{34}$$

The obtained labeling $\hat{\mathbf{y}}^j$ is then used to construct the vector $\mathbf{z}_{(j,\hat{\mathbf{y}}^j)} = \Psi(\mathbf{x}^j, \mathbf{y}^j) - \Psi(\mathbf{x}^j, \hat{\mathbf{y}}^j)$. Because the task (34) is NP-complete in general, we use an LP relaxation discussed in Section 2.1. This

means that the found $\hat{\mathbf{y}}^j$ is not optimal but it is usually good enough to satisfy the condition (30) which then leads to a guaranteed improvement of the optimized dual objective (cf. Lemma 1). LP relaxation algorithms also provides an upper bound UB_j on the optimal solution of (34) which is essential to compute feasible $\overline{\xi}_j$, $j \in \mathcal{J}$ in Step 3(a) of Algorithm 4. In particular, $\overline{\xi}_j$ is computed as $\overline{\xi}_j = \mathrm{UB}_j - F(\mathbf{x}^j, \mathbf{y}^j; \mathbf{q}, \mathbf{g})$.

6.2 Supermodular Max-Sum Classifier

In this section, we discuss optimization of the QP task (23) (or its compact form (24), respectively) using Algorithm 4.

The initialization in Step 1 of Algorithm 4 can be carried out in the same way as described in Section 6.1. Since the QP task (24) involves the primal constraints $\langle \mathbf{w}, \mathbf{z}_i \rangle \geq b_i$, $i \in I_0$, without slack variables the corresponding $\mathbf{z}_i$, $i \in I_0$, must be included in the reduced QP task during the entire progress of Algorithm 4. The size $|I_0| = |\mathcal{E}|(|\mathcal{Y}|-1)^2$ grows only quadratically which allows to use standard optimization packages.

The LAC task (29) is of the same form as that for a general max-sum problem, that is, it requires solving (34). Unlike the general case, the task (34) can be solved exactly by a polynomial-time algorithms provided the quality functions $\mathbf{g}$ obey the supermodularity condition. The requirement of supermodularity is expressed by the primal conditions $\langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle \geq 0$, $i \in I_0$ which are always included in the reduced task (28) and thus they should be always satisfied. There is one numerical issue, however, concerning a finite precision of QP solvers used to optimize the reduced task. It is possible that the primal constraints on supermodularity are slightly violated even if the reduced task is solved with a high precision. This might potentially (even though we have not observed the problem in practice) cause problems when solving the LAC task which relies on the supermodularity of $\mathbf{g}$. An easy solution to avoid the problem is to add the primal constraints $\langle \overline{\mathbf{w}}, \mathbf{z}_i \rangle \geq 0$, $i \in I_0$ with $\overline{\mathbf{w}} = \sum_{i \in \overline{I}} \alpha_i \mathbf{z}_i$ to the dual reduced task (28). In particular, we can solve the reduced task (28) subject to the constraints (28b) augmented by additional constraints $\sum_{i \in \overline{I}} \alpha_i H_{ij} \geq 0\,, \quad j \in I_0$. Note that adding of the additional constraints changes just the feasible set but it does not change the solution. Solving the reduced task with the added constraints by using any QP solver which produces a feasible solution with a finite precision (e.g., *interior point methods*) guarantees that the supermodularity constraints are satisfied.

6.3 Max-Sum Classifier with Strictly Trivial Equivalent

Unlike the previous two cases, the QP task (25) (or its compact form (22) respectively) contains only $n = m|\mathcal{E}|(|\mathcal{Y}|^2-1) + m|\mathcal{T}|(|\mathcal{Y}|-1) + m$ linear constraints. The form of the QP task (25) is the same as required for learning of an ordinary multi-class SVM (Crammer and Singer, 2001) which makes it possible to use existing optimization packages. The problem can be also solved by the proposed Algorithm 4. In this case, an initialization of the subset $(\overline{I}_j = \{u_j\}, u_j \in I_j), j \in \mathcal{J}$ performed in Step 1, can simply select the indices which correspond to the constraints $\xi_j \geq 0$, $j \in \mathcal{J}$. The LAC task (29) required in Step 3a can be solved exhaustively which means that we do not need any max-sum solver during the learning stage.

7. Experiments

Even though this article focuses primarily on theory, we present some examples to demonstrate functionality of the proposed learning algorithms. The examples are not meant to be a comprehensive evaluation.

7.1 Image Segmentation

We consider the problem of learning a classifier for segmentation of color images. We compare a general max-sum classifier, a supermodular max-sum classifier and a standard multi-class SVMs classifier (Crammer and Singer, 2001) as a baseline approach. In particular, we used the formulations given in Problem 4 and Problem 5 for learning the general and supermodular classifiers from the inconsistent training sets. To solve the LAC task required by Algorithm 4, we used an LP relaxation solver based on the Augmented Directed Acyclic Graph (ADAG) algorithm (Schlesinger 1976; see also the tutorial by Werner 2007).

We used the following three sets of color images (see Figure 1):

Microsoft Research (MSR) database: The original database [1] contains 240 images of 9 objects along with their manual segmentation. We selected a subset of 32 images each of which contains a combination of the following four objects: *cow*, *grass*, *water* and *void*. All images are of size 213×320 pixels. We created 3 random splits of the images into 10 training, 9 validation and 13 testing images. Reported results are averages taken over these three splits.

Shape-Sorter: We collected snapshots of a toy shape-sorter puzzle placed on a carpet. The snapshots are taken under varying view-angle and lighting conditions. The images contain 6 objects of different colors which we manually segmented. We split the images into a training set containing 14 images of size 100×100 pixels, validation and testing sets each containing 4 images of size 200×200 pixels.

Landscape: The data set is created from a single landscape image of size 280×600 which was manually segmented into *sky*, *border line* and *ground* areas. We divided the landscape image into 20 non-overlapping sub-windows of size 70×120 pixels. Finally, we split the 20 images into 5 training, 4 validation and 11 testing images.

In this experiment we define the max-sum classifier as follows. The set of objects $\mathcal{T} = \{(i,j) \mid i \in \{1,\dots,H\}, j \in \{1,\dots,W\}\}$ corresponds to the pixels of the input image of size $H \times W$. We consider a 4-neighborhood structure of image pixels, that is, $\mathcal{E}$ contains undirected edges between all pixels $(i,j) \in \mathcal{T}$ and $(i',j') \in \mathcal{T}$ which satisfy $|i-i'|+|j-j'|=1$. Each pixel $t \in \mathcal{T}$ is characterized by its observable state x_t and a label y_t. The observable state $x_t \in \mathcal{X} = \mathbb{R}^3$ is a vector containing RGB color values of t-th pixel. The label $y_t \in \mathcal{Y} = \{1,\dots,N\}$ assigns t-th pixel to one of N segments. We assume that the quality functions $q\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ and $g\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ do not depend on pixel $t \in \mathcal{T}$ (so called homogeneous model). The quality function $q(x,y) = \langle \mathbf{w}_y, x \rangle$ is linear in parameter $\mathbf{w}_y \in \mathbb{R}^3$, $y \in \mathcal{Y}$, as well as the function $g(y,y')$ represented by a vector $\mathbf{g} \in \mathbb{R}^{N^2}$. Hence the learned parameter vector $\mathbf{w} = (\mathbf{w}_1;\dots;\mathbf{w}_N;\mathbf{g}) \in \mathbb{R}^d$ is of dimension $d = 3N + N^2$. For the baseline approach we used a linear multi-class SVM which is equivalent to a max-sum classifier with a constant quality function $g(y,y')$, that is, the interrelation between labels is neglected.

1. Database B1 can be found at https://research.microsoft.com/vision/cambridge/recognition/default.htm.

	Multi-class SVM	General max-sum	Supermodular max-sum
MSR-database	21.74%	14.16%	14.03%
Shape-Sorter	1.31%	0.91%	0.92%
Landscape	6.22%	0.69%	0.65%

Table 1: Performance of max-sum classifiers compared to an ordinary SVM classifier on three segmentation data sets. The table contains average percentages of misclassified pixels per image.

We stopped the cutting plane Algorithm 4 when either the precision $(Q_P(\mathbf{w},\boldsymbol{\xi}) - Q_D(\boldsymbol{\alpha}))/Q_P(\mathbf{w},\boldsymbol{\xi}) \leq 10^{-2}$ was attained or no improving constraint was found after solving the LAC task. For the supermodular max-sum classifier the required precision was always attained since the LAC can be solved exactly. For the general max-sum classifier the algorithm failed to converge only in a few cases when the regularization constant C was set too high. The average time required for learning on an ordinary desktop PC was around 11 hours for the MSR-database, 3 hours for the Shape-Sorter data set and 6 minutes for the Landscape data set. The bottleneck is solving the LAC task by the ADAG max-sum solver which takes approximately 95% of the training time. Though we have not fully exploited this possibility, the training time for the supermodular max-sum classifier can be considerably reduced by using min-cut/max-flow algorithms to solve the LAC task. The min-cut/max-flow algorithms optimized for a grid neighbourhood structure can achieve near real-time performance (Boykov and Kolmogorov, 2004). For this reason the supermodular max-sum classifier is favourable when the training time or the classification time is of importance.

The only free parameter of the learning algorithm is the regularization constant C which we tuned on the validation images. The classifier with the best performance on the validation set was then assessed on independent testing images. Due to a different size of images it is convenient to present errors in terms of the normalized additive loss function $L(\mathbf{y},\mathbf{y}') = \frac{100}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}[\![y_t \neq y'_t]\!]$ corresponding to percentage of misclassified pixels in the image. Obtained results are summarized in Table 1. It is seen that the max-sum classifiers significantly outperformed the multi-class SVM on all data sets. Performances of the general and the supermodular max-sum classifiers are almost identical. This shows a good potential of the learning algorithm to extend applicability of the polynomially solvable class of supermodular max-sum problems. Note that selecting a proper supermodular function by hand is difficult due to an unintuitive form of the supermodularity conditions (7). Figure 1 shows examples of testing images and their labeling estimated by the max-sum classifier.

7.2 Learning the Rules of Sudoku

In this section, we demonstrate the algorithm for learning the max-sum classifier with a strictly trivial equivalent. We will consider a problem of learning the game rules of a logic-based number placement puzzle Sudoku[2]. Figures 2(a) and 2(b) show an example of the Sudoku puzzle and its solution, respectively. The aim of the puzzle is to fill in a 9×9 grid such that each column, each row and each of 9 non-overlapping 3×3 boxes contains the numbers from 1 to 9. A player starts

2. For more details and references see http://en.wikipedia.org/wiki/Sudoku .

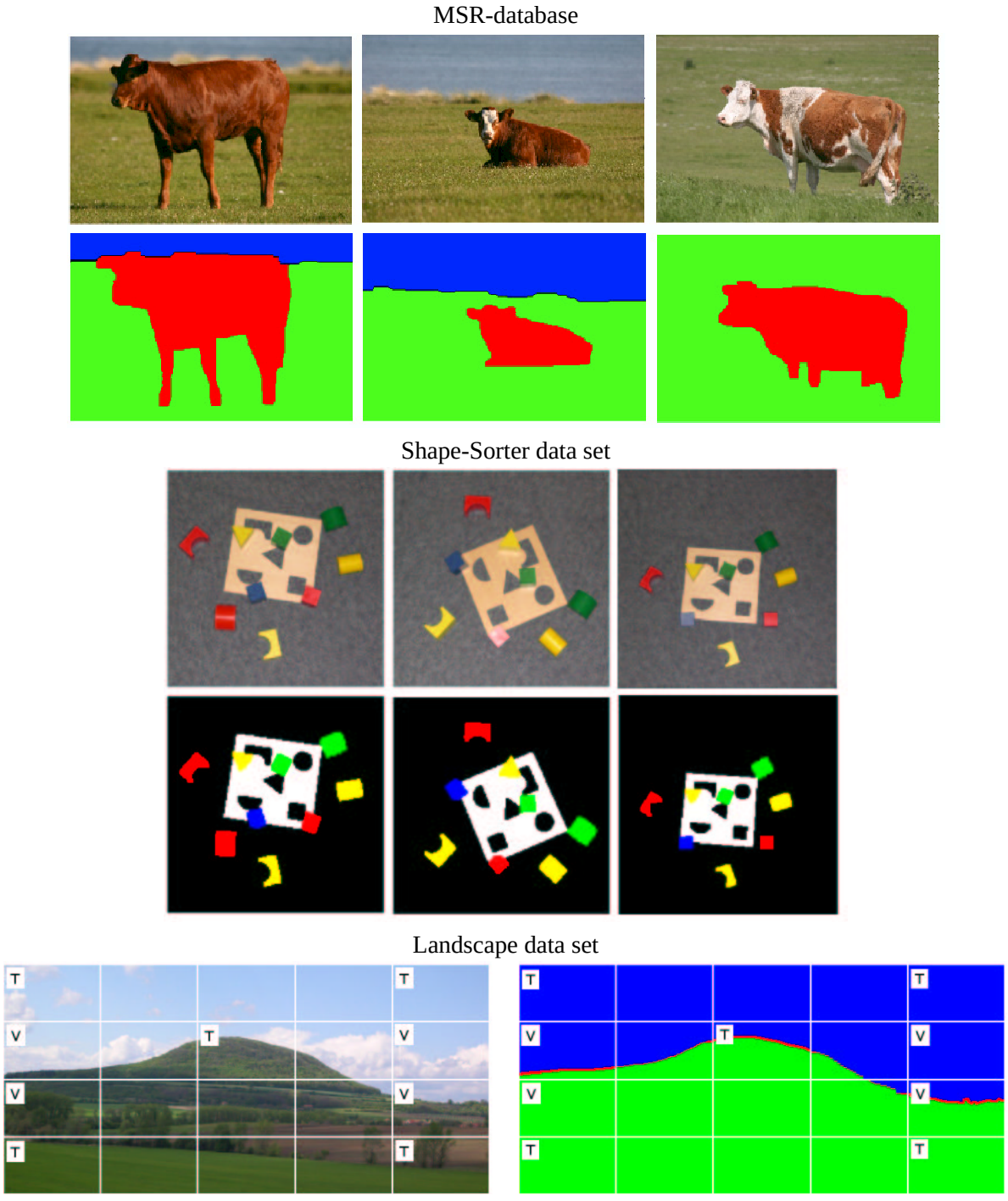


Figure 1: A sample from three data sets used in experiments. The figures show testing images and their labelings estimated by the max-sum classifier. For the Landscape data set the training and the validation sub-images are marked.

					8			
	1	9	5	6		2		
2	5			1		3	6	
9					2		8	1
	8	2	6		9			
5	7		1					2
	2	1		9			4	3
		5		7	6	8		
8	9		3					

(a)

7	6	3	4	2	8	1	9	5
4	1	9	5	6	3	2	7	8
2	5	8	9	1	7	3	6	4
9	3	4	7	5	2	6	8	1
1	8	2	6	3	9	4	5	7
5	7	6	1	8	4	9	3	2
6	2	1	8	9	5	7	4	3
3	4	5	2	7	6	8	1	9
8	9	7	3	4	1	5	2	6

(b)

(c)

Figure 2: Example of Sudoku puzzle: (a) input puzzle; (b) solution; (c) handwritten Sudoku grid created from the USPS database.

from an incompletely filled grid which is constructed such a way that the proper puzzle has a unique solution.

Our goal is to learn a max-sum classifier (artificial player) able to solve arbitrary Sudoku puzzle. A part of the rules is a priori known to the system while the rest is learned from examples of incomplete puzzle and its correct solution. In a standard scenario, the learned classifier is validated on a testing set. In this particular case, however, it is possible to show that the found max-sum classifier solves correctly all the Sudoku puzzles (the number of Sudoku solutions is approximately 6.6×10^{27}, that is, the number of puzzles is much larger).

Note, that the Sudoku puzzle can be naturally expressed as solving the *constraint satisfaction problem* CSP which is a subclass of a max-sum labeling problem when the quality functions attain only two values $\{0, -\infty\}$. From this viewpoint, a max-sum classifier is a richer model than necessarily needed for Sudoku puzzle.

Let us define the learning problem using the notation of this paper. Solving the puzzle is equivalent to solving the max-sum problem $(\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x})$. The set of objects $\mathcal{T} = \{(i, j) \mid i \in \{1, \ldots, 9\}, j \in \{1, \ldots, 9\}\}$ corresponds to the cells of a 9×9 grid. The neighbourhood structure

$$\mathcal{E} = \{\{(i, j), (i', j')\} \mid i = i' \vee j = j' \vee (\lceil i/3 \rceil = \lceil i'/3 \rceil \wedge \lceil j/3 \rceil = \lceil j'/3 \rceil)\},$$

contains pairs of cells whose relation plays a role in the game rules, that is, the cells in rows, columns, and the 3×3 boxes. The set of observations $\mathcal{X} = \{\square, 1, \ldots, 9\}$ represents the input filling of cells where the special symbol $\square$ means an empty cell. The set of labels $\mathcal{Y} = \{1, \ldots, 9\}$ corresponds to numbers which a player can fill in. The quality functions $\mathbf{q}$ and $\mathbf{g}$ are assumed to be homogeneous, that is, $q_t(x, y) = q(x, y)$ and $g_{tt'}(y, y') = g(y, y')$. Moreover, the structure of quality function $q(y, x)$ is assumed to be $q(y, x) = q(y)$ for $y = x$ and $q(y, x) = q_\square$ for $x = \square$ or $x \neq y$. The particular values of quality functions $(\mathbf{q}; \mathbf{g}) \in \mathbb{R}^{9 \times 9 + 10}$, which specify the rules of the game, are unknown to the system.

Using the setting introduced above, it is easy to see that the solution of a max-sum problem $(\mathcal{T}, \mathcal{E}, \mathcal{Y}, \mathbf{q}, \mathbf{g}, \mathbf{x})$ is a correct solution of the Sudoku puzzles if for any triplet $(y, y', y'') \in \mathcal{Y}^3$ such

$q(y,x)$

$q_\square$	y	1	2	3	4	5	6	7	8	9
-1870	$q(y)$	867	295	696	234	224	339	228	455	378

$g(y,y')$

y/y'	1	2	3	4	5	6	7	8	9
1	-552	19	47	66	65	73	61	66	73
2	84	-500	67	57	71	56	79	60	65
3	78	30	-437	48	48	47	68	50	54
4	73	-28	63	-266	55	53	76	56	60
5	23	67	57	47	-397	46	68	49	52
6	74	66	58	49	49	-448	69	49	54
7	-26	9	28	49	48	47	-342	48	54
8	36	77	60	49	49	46	67	-441	52
9	79	20	57	48	46	48	65	51	-441

Table 2: The quality functions **q** and **g** learned from examples. The learned functions **q** and **g** satisfy the conditions (35) which guarantee the optimal solution.

that $y' \neq y''$ the quality functions satisfy the following conditions

$$q_\square < q(y) \qquad \text{and} \qquad g(y,y) < g(y',y'')\,. \tag{35}$$

The first condition just forces the max-sum classifier to use the numbers from filled in cells as labels. The second condition, in compliance with to the Sudoku rules, ensures that neighbouring cells will not be assigned the same label. The conditions (35) allows us to verify whether the found classifier is an optimal one.

Because a proper Sudoku puzzle have a unique solution it is natural to require that the max-sum classifier also guarantees this condition. The conditions (35) show that the ground truth quality function $g(y,y')$ is not supermodular. For this reasoning, Algorithm 3 for learning of the max-sum classifier with a strictly trivial solution seems to be a reasonable choice.

We created 18 distinct training sets each containing a single example of an incomplete puzzle and its correct solution. In all cases, the proposed Algorithm 3 converged to a solution after the number of iterations ranging 12×10^3 to 95×10^3 approximately, which took less than one minute on an ordinary desktop PC. The conditions (35) were satisfied for all the solutions found by an algorithm, that is, an optimal max-sum classifier was found just from a single example (using more general **q** and **g** would probably require more examples). Table 2 shows an example of the found quality functions **q** and **g**. In this case, it is even possible to precisely compute the output of the max-sum classifier by branch-and-bound algorithm since the depth searching tree is limited due the requirement on a unique solution of the Sudoku and modest complexity tractable for human players. Note, however, that no max-sum solver was required during the course of the learning algorithm.

	Multi-class SVM	Max-Sum classifier
Linear	8.81%	7.20%
Gaussian kernel	5.80%	0.04%

Table 3: Recognition of Sudoku grids created from the USPS database.

7.3 Handwritten Character Recognition with Structured Outputs

In this section, we compare the max-sum classifier with a strictly trivial equivalent learned by the Perceptron Algorithm 3 against an ordinary multi-class SVM classifier. To this end, we modified the Sudoku problem from Section 7.2 in two ways. Firstly, the input observations are handwritten digits taken from the USPS database. In particular, the input observation $x \in \mathcal{X} = \mathbb{R}^{256}$ is a vector containing pixels of a grey-scale image 16×16 which depicts a digit from 1 to 9. Secondly, the input of the classifier is a fully solved Sudoku grid, that is, $\mathcal{Y} = \{1,\dots,9\}$, as the multi-class SVM cannot solve an incomplete puzzle unlike the max-sum classifier. The neighbourhood structure $(\mathcal{T},\mathcal{E})$ is the same as in the experiment from Section 7.2. Figure 2(c) shows and example of input observations and Figure2(b) shows a corresponding labeling to be estimated by the max-sum classifier.

Similarly to the experiment in Section 7.2, we considered a quality function $g(y,y')$ of a general form $\mathbf{g} \in \mathbb{R}^{|\mathcal{Y}|^2}$. However, we used two different forms of the quality functions $q(x,y)$. First, a linear function $q(x,y) = \langle \mathbf{w}_y, x\rangle$ which lead to learning a parameter vector $\mathbf{w} = (\mathbf{w}_1;\dots;\mathbf{w}_{|\mathcal{Y}|};\mathbf{g}) \in \mathbb{R}^d$ of dimension $d = 256|\mathcal{Y}| + |\mathcal{Y}|^2 = 2385$. Second, we applied the Gaussian kernel function defined as $k(x,x') = \exp(-\sigma\|x-x'\|^2)$ for some $\sigma > 0$. In this case, the quality function is $q(x,y) = \langle \mathbf{v}_y, \mathbf{k}(x)\rangle$ where $\mathbf{k}(x) = (k(x,x_t^j) \mid t \in \mathcal{T}, j = 1,\dots,m)$ denotes a vector of kernel functions centered in all training observations. The corresponding parameter vector $\mathbf{w} = (\mathbf{v}_1;\dots;\mathbf{v}_{|\mathcal{Y}|};\mathbf{g}) \in \mathbb{R}^d$ was of dimension $d = m|\mathcal{T}||\mathcal{Y}| + |\mathcal{Y}|^2 = 7371$.

We created a set of 30 examples of Sudoku grids $\mathbf{x} = (x_t \in \mathcal{X} \mid t \in \mathcal{T})$ and their solutions $\mathbf{y} = (y_t \in \mathcal{Y} \mid t \in \mathcal{T})$. Note that a single grid $\mathbf{x}$ contains $9 \times 9 = 81$ digits from the USPS database. We generated 3 random splits of the 30 examples into 10 training, 10 validation and 10 testing examples. The Perceptron Algorithm 3 required a few seconds to converge to a solution with zero training error when the linear kernel was used and around 3 minutes for the Gaussian kernel. The optimal kernel width σ was tuned on the validation data. In addition, we also tuned the regularization constant C for the multi-class SVM. The model with the best performance on the validation set was then assessed on the testing data. Table 3 shows the average classification performance computed over the 3 random splits.

It is seen that the max-sum classifier significantly outperforms the multi-class SVM regardless of the used kernel functions. The classification error 5.80% achieved for the multi-class SVM with the Gaussian kernel is slightly higher than 4.00% reported in Schölkopf et al. (1995). The reason is that we trained on a smaller number of examples (namely, 810 compared to 7291). On the other hand, the smaller training set is sufficient to achieve nearly error-less performance when the structure in the output space is considered. In particular, the error of the max-sum classifier with the Gaussian kernel is 0.04%. Note that without considering the structure a human recognition rate is 2.5% and the best published machine performance is 2.6% (Schölkopf et al., 1995).

8. Conclusions

In this article we have examined discriminative learning of max-sum classifiers. Learning of a max-sum classifier leads to satisfying a set of linear inequalities or solving a QP task with linear inequality constraints. A characteristic feature of these tasks is a huge number of linear constraints which is proportional to the number of possible responses (labelings) of a max-sum classifier. Efficient optimization methods for solving these tasks are the perceptron and the cutting plane algorithm, respectively. These methods manage to solve large problems provided the response of a max-sum classifier can be evaluated efficiently. Direct application of these methods is not tractable because computing a response of a general max-sum classifier is NP-complete.

We have proposed variants of the perceptron and the cutting plane algorithm for learning supermodular max-sum classifiers whose response can be computed efficiently in polynomial time. We have augmented the optimization tasks by additional linear constraints which guarantee that a max-sum classifier is supermodular. The perceptron and the cutting plane algorithm are modified such that added constraints on supermodularity are maintained satisfied during the course of optimization. This modification allows to compute the response of a max-sum classifier efficiently thus making the learning problem tractable.

We have defined a class of max-sum classifiers with a strictly trivial equivalent which are solvable exactly in polynomial time by an LP relaxation. We have showed that this class covers at least acyclic and supermodular max-sum classifiers with a unique solution. Another favorable property of this class is that the learning problems contain only polynomially-sized sets of linear constraints. As a result, the perceptron and the cutting plane algorithms do not require to call any max-sum solver during the course of optimization.

We have proposed a variant of the cutting plane algorithm which can approximately solve the learning problem formulated for the general max-sum classifier. The response of the max-sum classifier is approximated by an LP relaxation for which specialized optimization algorithms exist. Using an approximate response prohibits a guarantee that the cutting plane algorithm finds a solution with an arbitrary precision. This is not an obstacle, however, for using the algorithm in practice as we demonstrated experimentally.

Acknowledgments

Authors are deeply grateful to Prof. Michail I. Schlesinger for inspiring this work by many important ideas which he delivered in personal communication and during his lectures on structural pattern recognition. Our special thanks go to Tomáš Werner for reading this article and giving us many valuable comments.

The authors would like to acknowledged the support from the EU INTAS project PRINCESS 04-77-7347. The main part of this work has been done while the first author was with the Center for Machine Perception. The first author was also supported by Marie Curie Intra-European Fellowship grant SCOLES (MEIF-CT-2006-042107) during his current fellowship in Fraunhofer-FIRST.IDA institute.

Appendix A.

In this appendix we prove Theorem 3 introduced in Section 2.3. Prior to proving the theorem, we will introduce the concept of *local consistency* (Schlesinger, 1976) and the theorem asserting that the problems with a trivial equivalent contain the acyclic and supermodular problems (Schlesinger, 1976; Schlesinger and Flach, 2000).

Definition 3 *The maximal node* (t,y) *is called* locally consistent *if for each neighboring object* $t' \in \mathcal{N}(t)$ *there exists a maximal edge* $\{(t,y),(t',y')\}$. *The maximal edge* $\{(t,y),(t',y')\}$ *is called* locally consistent *if the nodes* (t,y) *and* (t',y') *are maximal.*

The maximal nodes and maximal edges which do not satisfy Definition 3 will be called *locally inconsistent*.

Theorem 6 *Schlesinger (1976); Schlesinger and Flach (2000) Let* $P = (\mathcal{T},\mathcal{E},\mathcal{Y},\mathbf{q},\mathbf{g},\mathbf{x})$ *be a max-sum problem. If* $(\mathcal{T},\mathcal{E})$ *is an acyclic graph or quality functions* $\mathbf{g}$ *are supermodular then* P *is equivalent to a trivial problem.*

Recall, that P with the optimal solution $\mathbf{y}^* \in \mathcal{Y}^{\mathcal{T}}$ has a trivial equivalent $P^{\boldsymbol{\varphi}}$ if there exist potentials $\boldsymbol{\varphi}$ such that the following set of linear inequalities holds

$$\begin{array}{rcll} q_t^{\boldsymbol{\varphi}}(y_t^*,x_t) & \geq & q_t^{\boldsymbol{\varphi}}(y,x_t)\,, & t \in \mathcal{T}\,, y \in \mathcal{Y} \setminus \{y_t^*\}\,, \\ g_{tt'}^{\boldsymbol{\varphi}}(y_t^*,y_{t'}^*) & \geq & g_{tt'}^{\boldsymbol{\varphi}}(y_t,y_{t'})\,, & \{t,t'\} \in \mathcal{E}\,, (y,y') \in \mathcal{Y}^2 \setminus \{(y_t^*,y_{t'}^*)\}\,. \end{array} \tag{36}$$

The system (36) differs from the definition of problems with a strictly trivial equivalent (19) just by using the non-strict inequalities. Finally, we will prove two auxiliary lemmas:

Lemma 2 *Let* P *be an acyclic problem which has a unique solution* $\mathbf{y}^*$ *and let* $P^{\boldsymbol{\varphi}}$ *be a trivial equivalent of* P. *Then only two cases can occur: (i)* $P^{\boldsymbol{\varphi}}$ *is strictly trivial or (ii) there is at least one maximal locally inconsistent node or edge.*

Proof We will show that violating both the assertions (i) and (ii) contradicts the assumption that $\mathbf{y}^*$ is unique. Assuming $P^{\boldsymbol{\varphi}}$ is not strictly trivial implies that there exists a maximal node (t,y_t^0) such that $y_t^0 \neq y_t^*$. Let us construct a labeling $\mathbf{y}^0$ such that y_t^0 belongs to $\mathbf{y}^0$. The remaining labels are determined by repeating the following procedure $(|\mathcal{T}|-1)$ times:

- Let $t' \in \mathcal{T}$ be an object whose label $y_{t'}^0$ has been already determined and let $t'' \in \mathcal{N}(t')$ be an object whose label $y_{t''}^0$ has not been determined yet. Then set up the label $y_{t''}^0$ such that $\{(t',y_{t'}^0),(t'',y_{t''}^0)\}$ is a maximal edge.

The constructed labeling $\mathbf{y}^0$ is the optimal solution of P because it is composed of maximal nodes and edges. Note that this simple construction of the optimal labeling is possible because the graph $(\mathcal{T},\mathcal{E})$ is acyclic. Thus we have $\mathbf{y}^0 \neq \mathbf{y}^*$ because $y_t^0 \neq y_t^*$ which implies that $\mathbf{y}^*$ is not unique. ■

Lemma 3 *Let* P *be a supermodular problem with an unique solution* $\mathbf{y}^*$ *and let* $P^{\boldsymbol{\varphi}}$ *be a trivial equivalent of* P. *Then only two cases can occur: (i)* $P^{\boldsymbol{\varphi}}$ *is strictly trivial or (ii) there is at least one maximal locally inconsistent node or edge.*

Proof We will show that violating both the assertions (i) and (ii) contradicts the assumption that $\mathbf{y}^*$ is unique. Let $\mathcal{Y}_t^0 = \{(t,y) \mid q_t^{\boldsymbol{\varphi}}(y,x_t) = \max_{y' \in \mathcal{Y}} q_t^{\boldsymbol{\varphi}}(y',x_t)\}$ denote a set of all maximal nodes corresponding to the object $t \in \mathcal{T}$. Let us construct the labeling $\mathbf{y}^h = (y_t^h \mid t \in \mathcal{T})$ composed of the highest maximal nodes; (t,y_t^h) is the highest maximal node if $(t,y_t^h) \in \mathcal{Y}_t^0$ and $y_t^h > y$ for all $(t,y) \in \mathcal{Y}_t^0 \setminus \{t,y_t^0\}$. Recall, that the labels are fully ordered for the supermodular problems.

Now, we show that the labeling $\mathbf{y}^h$ is optimal since all its edges $\{(t,y_t^h),(t',y_{t'}^h)\} \in \mathcal{E}_{\mathcal{Y}}$ are also maximal. Let us assume that there exists an edge $\{(t,y_t^h),(t',y_{t'}^h)\}$ which is not maximal. Then, by assumption of local consistency, there exist edges $\{(t,y_t^h),(t',y_{t'})\}$ and $\{(t,y_t),(t',y_{t'}^h)\}$ which are maximal. Note that $y_t < y_t^h$ and $y_{t'} < y_{t'}^h$ because (t,y_t^h) and $(t',y_{t'}^h)$ are the highest nodes. From the condition of supermodularity (7) and (4a) we have

$$\begin{aligned} 0 \leq g_{tt'}(y_t^h,y_{t'}^h) + g_{tt'}(y_t,y_{t'}) - g_{tt'}(y_t^h,y_{t'}) - g_{tt'}(y_t,y_{t'}^h) \\ = g_{tt'}^{\boldsymbol{\varphi}}(y_t^h,y_{t'}^h) + g_{tt'}^{\boldsymbol{\varphi}}(y_t,y_{t'}) - g_{tt'}^{\boldsymbol{\varphi}}(y_t^h,y_{t'}) - g_{tt'}^{\boldsymbol{\varphi}}(y_t,y_{t'}^h)\,. \end{aligned}$$

This condition, however, cannot be satisfied if the edge $\{(t,y_t^h),(t',y_{t'}^h)\}$ is not maximal which is a contradiction. Similarly, we can show that the labeling $\mathbf{y}^l$ composed of the lowest maximal nodes (defined analogically) is also optimal.

Finally, the assumption that $P^{\boldsymbol{\varphi}}$ is not strictly trivial implies that for some object $t \in \mathcal{T}$ there exists a maximal node (t,y_t^0) such that $y_t^0 \neq y_t^*$. W.l.o.g. we can select (t,y_t^0) which is either the highest maximal or the lowest maximal node. Thus $y_{t^*}^0$ belongs either to the labeling $\mathbf{y}^h$ or $\mathbf{y}^l$ which are optimal and differ from $\mathbf{y}^*$. This contradicts the assumption that $\mathbf{y}^*$ is unique. ■

Proof of Theorem 3: Let P be an acyclic or supermodular max-sum problem with the unique solution $\mathbf{y}^* \in \mathcal{Y}^{\mathcal{T}}$. Let $\boldsymbol{\varphi}$ be the potentials such that the max-sum problem $P^{\boldsymbol{\varphi}}$ is a trivial equivalent of P. The existence of such $P^{\boldsymbol{\varphi}}$ is guaranteed by Theorem 6. Then, by Lemma 2 and Lemma 3, $P^{\boldsymbol{\varphi}}$ is either strictly trivial or there exists a maximal node (t,y_t^0) or a maximal edge $\{(t,y_t^0),(t',y_{t'}^0)\}$ which are locally inconsistent, that is, (t,y_t^0) and $\{(t,y_t^0),(t',y_{t'}^0)\}$ do not belong to $\mathbf{y}^*$. We will introduce a procedure which changes the potentials $\boldsymbol{\varphi}$ in such a way that the inconsistent maximal node (t,y_t^0) or inconsistent maximal edge $\{(t,y_t^0),(t',y_{t'}^0)\}$, respectively, become non-maximal while other maximal (non-maximal) nodes or edges remain maximal (non-maximal). Repeating this procedure for all inconsistent maximal nodes and edges makes the inequalities (36) satisfied strictly, that is, the problem $P^{\boldsymbol{\varphi}}$ becomes strictly trivial which is to be proven. The procedures for elimination of inconsistent nodes and edges read:

Elimination of the inconsistent maximal node (t,y_t^0)**:** Let $t' \in \mathcal{N}(t)$ be such a neighbor of t that the set of edges $\mathcal{E}_{tt'}(y_t^0) = \{\{(t,y_t^0),(t',y_{t'})\} \mid y_{t'} \in \mathcal{Y}\}$ does not contain any maximal edge. Let ε be a number computed as

$$\varepsilon = \frac{1}{2}\left[\max_{(y,y') \in \mathcal{Y}^2} g_{tt'}^{\boldsymbol{\varphi}}(y,y') - \max_{y' \in \mathcal{Y}} g_{tt'}^{\boldsymbol{\varphi}}(y_t^0,y')\right].$$

Since $\mathcal{E}_{tt'}(y_t^0)$ does not contain maximal edges this implies $\varepsilon > 0$. Adding ε to the potential $\varphi_{tt'}(y_t^0) := \varphi_{tt'}(y_t^0) + \varepsilon$ decreases the quality $q_t^{\boldsymbol{\varphi}}(y_t^0,x_t) = q_t(y_t^0,x_t) - \sum_{t'' \in \mathcal{N}(t)} \varphi_{tt''}(y_t^0)$ by ε and increases the qualities $g_{tt'}^{\boldsymbol{\varphi}}(y,y') = g_{tt'}(y,y') + \varphi_{tt'}(y) + \varphi_{t't}(y')$, $\{(t,y),(t',y')\} \in \mathcal{E}_{tt'}(y_t^0)$ by ε. This change makes the node (t,y_t^0) non-maximal while the edges $\mathcal{E}_{tt'}(y_t^0)$ remain non-maximal as before. The qualities of other nodes and edges remain unchanged.

Elimination of the inconsistent maximal edge $\{(t,y_t^0),(t',y_{t'}^0)\}$**:** W.l.o.g. let $(t',y_{t'}^0)$ be a non-maximal node. Notice, that all edges from $\mathcal{E}_{t't}(y_{t'}^0) = \{(t,y_t),(t',y_{t'}^0)\} \mid y_t \in \mathcal{Y}\}$ are locally inconsistent and they cannot be a part of the optimal solution. Let ε be a number computed as

$$\varepsilon = \frac{1}{2}\left[\max_{y\in\mathcal{Y}} q_{t'}^{\boldsymbol{\varphi}}(y,x_{t'}) - q_{t'}^{\boldsymbol{\varphi}}(y_{t'}^0,x_{t'})\right].$$

Because $(t',y_{t'}^0)$ is non-maximal $\varepsilon > 0$. Subtracting ε from the potential $\varphi_{t't}(y_{t'}^0) := \varphi_{t't}(y_{t'}^0) - \varepsilon$ decreases the qualities $g_{tt'}^{\boldsymbol{\varphi}}(y,y') = g_{tt'}(y,y') + \varphi_{tt'}(y) + \varphi_{t't}(y')$, $\{(t,y),(t',y')\} \in \mathcal{E}_{t't}(y_{t'}^0)$ by ε and increases the quality $q_{t'}^{\boldsymbol{\varphi}}(y_{t'}^0,x_{t'}) = q_{t'}(y_{t'}^0,x_{t'}) - \sum_{t''\in\mathcal{N}(t')}\varphi_{tt''}(y_{t'}^0)$ by ε. This change makes all edges from $\mathcal{E}_{t't}(y_{t'}^0)$ non-maximal while the node $(t',y_{t'}^0)$ remains non-maximal as before. The qualities of other nodes and edges remain unchanged. ■

Appendix B.

In this appendix we prove Lemma 1 given in Section 6.

Proof of Lemma 1: We show that if a new variable was added then we can construct a vector $\overline{\boldsymbol{\beta}}$ such that optimizing the dual objective $Q_D(\boldsymbol{\alpha})$ over a line segment between the current solution $\overline{\boldsymbol{\alpha}}$ and the vector $\overline{\boldsymbol{\beta}}$ yields a guaranteed improvement. Since the reduced QP task solved in Step 2 optimizes in the space of all selected variables which contains the line segment between $\overline{\boldsymbol{\alpha}}$ and $\overline{\boldsymbol{\beta}}$, the obtained improvement cannot be smaller.

Let us assume an optimization of the dual objective $Q_D(\boldsymbol{\alpha})$ of the QP task (27) w.r.t. a line segment between the current solution $\overline{\boldsymbol{\alpha}}$ and an arbitrary feasible vector $\overline{\boldsymbol{\beta}}$. The problem is equivalent to searching for the maximum of an univariate quadratic function

$$\begin{aligned} Q_L(\tau) &= Q_D\Big((1-\tau)\overline{\boldsymbol{\alpha}} + \tau\overline{\boldsymbol{\beta}}\Big) \\ &= (1-\tau)\langle \mathbf{b}, \overline{\boldsymbol{\alpha}}\rangle + \tau\langle \mathbf{b}, \overline{\boldsymbol{\beta}}\rangle - \frac{1}{2}(1-\tau)^2\langle \overline{\boldsymbol{\alpha}}, \mathbf{H}\overline{\boldsymbol{\alpha}}\rangle - \tau(1-\tau)\langle \overline{\boldsymbol{\beta}}, \mathbf{H}\overline{\boldsymbol{\alpha}}\rangle - \frac{1}{2}\tau^2\langle \overline{\boldsymbol{\beta}}, \mathbf{H}\overline{\boldsymbol{\beta}}\rangle\,, \end{aligned}$$

over the closed interval $0 \le \tau \le 1$. The maximum is attained at the vector

$$\overline{\boldsymbol{\alpha}}_{\text{new}} = (1-\overline{\tau})\overline{\boldsymbol{\alpha}} + \overline{\tau}\overline{\boldsymbol{\beta}} \tag{37}$$

where

$$\overline{\tau} = \operatorname*{argmax}_{0\le\tau\le 1} Q_L(\tau)\,. \tag{38}$$

The derivative of $Q_L(\tau)$ reads

$$\frac{\partial Q_L(\tau)}{\partial \tau} = \langle \mathbf{b}, \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}\rangle + (1-\tau)\langle \overline{\boldsymbol{\alpha}}, \mathbf{H}\overline{\boldsymbol{\alpha}}\rangle - (1-2\tau)\langle \overline{\boldsymbol{\beta}}, \mathbf{H}\overline{\boldsymbol{\alpha}}\rangle - \tau\langle \overline{\boldsymbol{\beta}}, \mathbf{H}\overline{\boldsymbol{\beta}}\rangle\,.$$

An objective function is improved, that is, $Q_D(\overline{\boldsymbol{\alpha}}_{\text{new}}) - Q_D(\overline{\boldsymbol{\alpha}}) > 0$, iff the vector $\overline{\boldsymbol{\beta}}$ satisfies

$$\left.\frac{\partial Q_L(\tau)}{\partial \tau}\right|_{\tau=0} = \langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}}\rangle > 0\,. \tag{39}$$

Provided (39) holds, the maximum of $Q_L(\tau)$ w.r.t. $0 \le \tau \le 1$ is attained within the open interval $0 < \tau < 1$ or on its boundary $\tau = 1$. The maximum of $Q_L(\tau)$ w.r.t. unbounded τ can be found by solving $\frac{\partial Q_L}{\partial \tau} = 0$ for τ. If the resulting τ exceeds 1 then the optimum of the line segment optimization is attained at $\overline{\tau} = 1$. Thus we can write the solution of (38) in a closed form

$$\overline{\tau} = \min \left\{ 1, \frac{\langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}} \rangle}{\langle \overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}, \mathbf{H}(\overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}) \rangle} \right\} . \tag{40}$$

Analytical formulas for improvement can be derived substituting (37) and (40) to $\Delta = Q_D(\overline{\boldsymbol{\alpha}}_{\text{new}}) - Q_D(\overline{\boldsymbol{\alpha}})$. For $\overline{\tau} < 1$ we get

$$\Delta = \frac{\langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}} \rangle^2}{2\langle \overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}, \mathbf{H}(\overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}) \rangle} , \tag{41}$$

and for $\overline{\tau} = 1$ we get

$$\Delta \quad = \quad \langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}} \rangle - \frac{1}{2}\langle \overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}, \mathbf{H}(\overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}) \rangle \ge \frac{1}{2}\langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}} \rangle . \tag{42}$$

The last inequality in (42) follows from $\langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}} \rangle \ge \langle \overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}, \mathbf{H}(\overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}) \rangle$ which holds for $\overline{\tau} = 1$ as seen from (40).

Let us consider that a new variable with index u_j was added in Step 3(b), that is, the condition (30) was satisfied. Using $\langle \overline{\mathbf{w}}, \mathbf{z}_{u_j} \rangle = [\mathbf{H}\boldsymbol{\alpha}]_{u_j}$ we can rewrite the condition (30) as

$$\frac{C}{m}[\mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}}]_{u_j} - \sum_{i \in I_j} \overline{\alpha}_i [\mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}}]_i > \frac{\varepsilon}{m} . \tag{43}$$

Let us construct the feasible vector $\overline{\boldsymbol{\beta}} = (\overline{\beta}_1, \ldots, \overline{\beta}_n)^T$ as follows

$$\overline{\beta}_i = \begin{cases} \frac{C}{m} & \text{if} \quad i = u_j , \\ 0 & \text{if} \quad i \in I_j \setminus \{u_j\} , \\ \overline{\alpha}_i & \text{if} \quad i \in I \setminus I_j . \end{cases} \tag{44}$$

As was shown above, optimization over the line segment yields an improvement provided (39) holds. Substituting $\overline{\boldsymbol{\beta}}$ constructed by (44) into the formula (39) we get

$$\frac{dQ_L}{d\tau}\Big|_{\tau=0} = \langle \overline{\boldsymbol{\beta}} - \overline{\boldsymbol{\alpha}}, \mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}} \rangle = \frac{C}{m}[\mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}}]_{u_j} - \sum_{i \in I_j} \overline{\alpha}_i [\mathbf{b} - \mathbf{H}\overline{\boldsymbol{\alpha}}]_i > \frac{\varepsilon}{m} , \tag{45}$$

where the last inequality follows from (43). This implies that optimizing w.r.t. line segment between $\overline{\boldsymbol{\alpha}}$ and the vector $\overline{\boldsymbol{\beta}}$ yields positive improvement. Now, we derive a lower bound on this improvement. Combining (45) with (42) we immediately get

$$\Delta \ge \frac{\varepsilon}{2m} \quad \text{for} \quad \overline{\tau} = 1 . \tag{46}$$

Before deriving the bound for $\bar{\tau} < 1$, we must find an upper bound on the denominator of (41). Let us define $\mathcal{A}_j = \left\{\boldsymbol{\alpha} \,|\, \sum_{i \in I_j} \alpha_i = \frac{C}{m}\,, \alpha_i \geq 0\,, \forall i \in I_j\right\}$. Then we can write

$$\begin{aligned}
\langle \overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}, \mathbf{H}(\overline{\boldsymbol{\alpha}} - \overline{\boldsymbol{\beta}}) \rangle &\leq \max_{j \in \mathcal{J}} \left\| \sum_{i \in I_j} \overline{\alpha}_i \mathbf{z}_i - \sum_{i \in I_j} \overline{\beta}_i \mathbf{z}_i \right\|^2 \\
&\leq \left(2 \max_{j \in \mathcal{J}} \max_{\boldsymbol{\alpha} \in \mathcal{A}_j} \left\| \sum_{i \in I_j} \alpha_i \mathbf{z}_i \right\| \right)^2 \qquad (47) \\
&= \frac{4C^2}{m^2} \max_{j \in \mathcal{J}} \max_{i \in I_j} \|\mathbf{z}_i\|^2 \,.
\end{aligned}$$

Combining (45) and (47) with (41) yields

$$\Delta \geq \frac{\varepsilon^2}{8C^2R^2} \quad \text{for} \quad \bar{\tau} = 1 \quad \text{and} \quad R = \max_{j \in \mathcal{J}} \max_{i \in I_j} \|\mathbf{z}_i\| \,. \qquad (48)$$

Taking the minimum of improvements (46) and (48) gives the bound on the minimal improvement. ■

References

Y. Altun and T. Hofmann. Large margin methods for label sequence learning. In *European Conference on Speech Communication and Technology (EuroSpeech)*, 2003.

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *International Conference on Machine Learning*. ACM Press, New York, 2003.

D. Anguelov, B. Taskar, V. Chatabashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmentation of 3D scan data. In *International Conference on Computer Vision and Pattern Recognition*, pages 167–176. IEEE Computer Society, Washington, DC, 2005.

J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48: 259–302, 1986.

Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, Sept. 2004.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, Nov. 2001.

C. Chekuri, S. Khanna, J. Naor, and L. Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Symposium on Discrete Algorithms*, pages 109–118, 2001.

P.B. Chou and C.M. Brown. The theory and practice of bayesian image labeling. *International Journal on Computer Vision*, 4(3):185–210, June 1990.

M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Conference on Empirical Methods in Natural Language Processing*, pages 1–8, 2002.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, Dec. 2001.

T. Finley and T. Joachims. Supervised clustering with support vector machines. In *International Conference on Machine Learning*, pages 217–224. ACM Press, New York, 2005.

R. M. Haralick and L. G. Shapiro. The consistent labeling problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):173–184, 1979.

G. E. Hinton and T. J. Sejnowski. Learning and relearning in boltzmann machines. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 282–317. MIT Press, Cambridge, 1986.

M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal of Computing*, 22:1087–1116, 1993.

V. Kolmogorov. Convergent tree-reweighted message passing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *European Conference on Computer Vision*, pages 65–81. Springer-Verlag, 2002.

A. Koster, C. P. M. Hoesel, and A. W. J. Kolen. The partial constraint satisfaction problem: Facets and lifting theorems. *Operations Research Letters*, 23(3–5):89–97, 1998.

V.A. Koval and M.I. Schlesinger. Dvumernoe programmirovanie v zadachakh analiza izobrazheniy (two-dimensional programming in image analysis problems). *USSR Academy of Science, Automatics and Telemechanics*, 8:149–168, 1976. In Russian.

I. Kovtun. *Segmentaciya Zobrazhen na Usnovi Dostatnikh Umov Optimalnosti v NP-povnikh Klasakh Zadach Strukturnoi Rozmitki (Image Segmentation Based on Sufficient Conditions Of Optimality in NP-complete Classes of Structural Labeling Problems)*. PhD thesis, IRTC ITS Nat. Academy of Science Ukraine, Kiev, 2004. In Ukrainian.

A. Novikoff. On convergence proofs of perceptrons. In *Symposium on Mathematical Theory of Automata*, volume 12, pages 615–622. Polytechnic Institute of Brooklyn, 1962.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, USA, 1988.

N.D. Ratliff and J.A. Bagnell. Subgradient methods for maximum margin structured learning. In *ICML Workshop on Learning in Structured Output Spaces*, 2006.

A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(6):420–433, June 1976.

D. Schlesinger. *Structurelle Ansätze für die Stereoreconstruction*. PhD thesis, Technische Universität Dresden, Fakultät Informatik, Institut für Künstliche Intelligenz, July 2005. In German.

M.I. Schlesinger. Sintaksicheskiy analiz dvumernykh zritelnikh signalov v usloviyakh pomekh (Syntactic analysis of two-dimensional visual signals in noisy conditions). *Kibernetika*, 4:113–130, 1976. In Russian.

M.I. Schlesinger and B. Flach. Some solvable sublcasses of structural recognition problems. In *Czech Pattern Recognition Workshop*. Czech Pattern Recognition Society, 2000.

M.I. Schlesinger and V. Hlaváč. *Ten Lectures on Statistical and Structural Pattern Recognition*. Kluwer Academic Publishers, 2002.

B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In U.M. Fayyad and R. Uthurusamy, editors, *International Conference on Knowledge Discovery & Data Mining*. AAAI Press, Menlo Park, 1995.

B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *International Conference on Machine Learning*. ACM Press, New York, 2004a.

B. Taskar, C. Guestrin, and D. Koller. Maximum-margin markov networks. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004b.

B. Taskar, S. Lacoste-Jullien, and M.I. Jordan. Structured prediction, dual extragradient and bregman projections. *Journal of Machine Learning Research*, 7:1627–1653, Jul. 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, Sep. 2005.

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., 1998.

M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on hypertrees: message passing and linear programming approaches. In *Conference on Communication, Control and Computing*, 2002.

T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, July 2007.

T. Werner. A linear programming approach to max-sum problem: A review. Research Report CTU–CMP–2005–25, Center for Machine Perception, K13133 FEE Czech Technical University, Prague, Czech Republic, December 2005.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

Journal of Machine Learning Research 9 (2008) 105-130 Submitted 1/07; Revised 12/07; Published 1/08

Active Learning by Spherical Subdivision

Falk-Florian Henrich HENRICH@MATH.TU-BERLIN.DE
Technische Universität Berlin
Institut für Mathematik
Sekr. MA 8–3
Straße des 17. Juni 136
D-10623 Berlin, Germany

Klaus Obermayer OBY@CS.TU-BERLIN.DE
Technische Universität Berlin
Institut für Informatik
FR 2–1, NI
Franklinstraße 28/29
D-10587 Berlin, Germany

Editor: David Cohn

Abstract

We introduce a computationally feasible, "constructive" active learning method for binary classification. The learning algorithm is initially formulated for separable classification problems, for a hyperspherical data space with constant data density, and for great spheres as classifiers. In order to reduce computational complexity the version space is restricted to spherical simplices and learning procedes by subdividing the edges of maximal length. We show that this procedure optimally reduces a tight upper bound on the generalization error. The method is then extended to other separable classification problems using products of spheres as data spaces and isometries induced by charts of the sphere. An upper bound is provided for the probability of disagreement between classifiers (hence the generalization error) for non-constant data densities on the sphere. The emphasis of this work lies on providing mathematically exact performance estimates for active learning strategies.

Keywords: active learning, spherical subdivision, error bounds, simplex halving

1. Introduction

Active learning methods seek a solution to inductive learning problems by incorporating the selection of training data into the learning process. In these schemes, the labeling of a data point occurs only after the algorithm has explicitly asked for the corresponding label, and the goal of the "active" data selection is to reach the same accuracy as standard "passive" algorithms—but with less labeled data points. In many practical tasks, the acquisition of unlabeled data can be automated, while the actual labeling must often be done by humans and is therefore time consuming and costly. In these cases, active learning methods—which usually trade labeling costs against the computational burden required for optimal data selection—can be a valuable alternative.

There are two approaches to active learning. So-called query filtering methods (Freund et al., 1997; Opper et al., 1992) operate on a given pool of unlabeled data and select—at every learning step—a "most informative" data point for subsequent labeling. So-called constructive methods lit-

erally "construct" an unlabeled datum and ask the user to provide a label. There is strong empirical evidence for many learning scenarios and for different selection procedures, that active learning methods can reduce the number of labeled training data which are needed to reach a predefined generalization performance (see Balcan et al., 2006; Fine et al., 2002; Freund et al., 1997; Tong and Koller, 2001; Warmuth et al., 2002). In addition, theoretical work has shown that active learning strategies can achieve an exponential reduction of the generalization error *with high probability* (see Balcan et al., 2006; Freund et al., 1997). In this contribution, we put the emphasis on a mathematically rigorous derivation of *hard* upper bounds for the generalization error. This is in contrast to other studies which give bounds in probability (see Freund et al., 1997) or discuss asymptotic behavior (see Bach, 2007).

We use a geometrical approach to active learning which is based on the concept of a version space (see Mitchell, 1982; Tong and Koller, 2001). Loosely speaking, given a set of predictors and a set of labeled training data, "version space" denotes the set of models whose predictions are consistent with the training data. If a version space can be defined, active learning strategies should evaluate data points which allow the learning machine to maximally reduce the "size" of its version space at every learning step. Some active learning algorithms try to halve the volume of the version space (see Tong and Koller, 2001). In contrast to this, our approach is to reduce the *maximal distance* between pairs of points that belong to the version space. We prefer distance over volume simply because it is impossible to compute the exact volume of version space in high dimensions. A detailed discussion of this problem can be found at the end of Section 7. For special types of data spaces, our method of maximal length reduction coincides with volume reduction. In this case, we observe a rapid (that is, exponential) progress in learning, as is explicitly shown in Section 4 and Proposition 9 of Section 6 for data which is arbitrarily distributed on an n-dimensional torus.

In the following we will consider the simple case of a separable, binary classification problem. Assuming the knowledge of the data density μ on the data space M, the generalization distance $d^G(c_1,c_2)$ of two classifiers $c_1,c_2 : M \to \mathbb{Z}_2 := \{0,1\}$ is given by the integral

$$d^G(c_1,c_2) := \frac{1}{\mathrm{Vol}(M)} \int_{D(c_1,c_2)} \omega,$$

where $D(c_1,c_2) := \{x \in M \mid c_1(x) \neq c_2(x)\}$ is the set of points which are classified differently by the two classifiers, and ω denotes the volume form of M (see Section 2). If we further assume the labeling of the data to be generated by an unknown classifier $c^* : M \to \mathbb{Z}_2$, the important question is: *Can we give tight upper bounds on the generalization error $d^G(c,c^*)$ of some classifier c, and can we reduce this bound during learning in an optimal way?* Inspired by this question, we introduce a constructive active learning algorithm which reduces such a bound by successive subdivisions of the version space on a hypersphere. This then allows us to compute exact and tight generalization error bounds for several classes of data densities. After deriving the bounds for the case of an uniform distribution on the hypersphere, we use the notion of Riemannian isometries to extend the algorithm and the error bounds to a set of selected data densities on other data spaces including $\mathbb{R}^n$. In a second step, we extend our results to product manifolds in order to obtain sharp error bounds for a larger set of data densities. Finally, we provide bounds for arbitrary densities on $\mathbb{R}^n$, hyperspheres and products thereof.

The article is organized as follows: After introducing the geometric setup of active learning for binary classification (Section 2), we formulate a learning method for data distributed on the unit sphere $S^n \subset \mathbb{R}^{n+1}$ (Section 3). Thereafter, we extend the basic algorithm to a broader class

of separable, binary classification problems using isometries induced by charts of the sphere and products of hyperspheres (Sections 4, 5). This will include perceptrons (that is, linear classifiers with bias on $\mathbb{R}^n$) as a special case. Upper bounds are provided for the generalization error for non-constant data densities for binary classification problems on the sphere (Section 6) and for linear classifiers without bias in $\mathbb{R}^n$ (Section 7). Finally, Section 8 provides an empirical evaluation of the geometric method. As our focus lies on a theoretical analysis of active learning algorithms, applications of the proposed algorithm to concrete problems have to be of second importance.

2. A Geometric Setup for Active Learning

In the sequel, we will apply some standard constructions from differential geometry. We refer to Appendix B for a quick introduction to the terminology.

Let M be an n-dimensional compact manifold, the *data space*, equipped with a Riemannian metric g. One might object to this type of data space, since the compactness assumption seems to rule out the most important instance of data space, the Euclidean space $\mathbb{R}^n$. However, this is not the case, because $\mathbb{R}^n$, or any submanifold therein, can be embedded into S^n, the n-sphere, by the inverse of stereographic projection (see Section 4). Recently, spherical data spaces have received some attention in machine learning (see Lebanon and Lafferty, 2004; Belkin and Niyogi, 2004; Minh et al., 2006).

We assume the existence of an *unknown binary classifier* $c^* : M \to \mathbb{Z}_2 := \{0,1\}$ and a given set of labeled data points $\{(x_1,y_1),\ldots,(x_I,y_I))\}$, $(x_i,y_i) \in M \times \mathbb{Z}_2$, with correct labels, that is, $c^*(x_i) = y_i$. Now, the *binary classification problem* asks for an approximation $c : M \to \mathbb{Z}_2$ which minimizes the *generalization error*, that is, the probability of misclassification of data points. This can be formalized as follows.

The Riemannian volume form ω which belongs to the metric g is given in local coordinates $x : M \supset U \to \mathbb{R}^n$ by

$$\omega = \sqrt{\det(g)}dx_1 \wedge \ldots \wedge dx_n, \tag{1}$$

where U is some open chart domain in M. We assume that the Riemannian volume form ω (see Equation 1) represents up to a scaling factor $\mathrm{Vol}(M) := \int_M \omega$ the p.d.f. of the data points $x \in M$. This allows us to interpret the probability of disagreement between two classifiers c_1, c_2 as a distance measure[1] on the set of all classifiers:

$$d^G(c_1,c_2) := \frac{1}{\mathrm{Vol}(M)} \int_{D(c_1,c_2)} \omega, \tag{2}$$

where $D(c_1,c_2) := \{x \in M \mid c_1(x) \neq c_2(x)\}$ is called the *disagreement area*. In these terms, the generalization error of c is given by $d^G(c,c^*)$.

3. Subdivisions of the Sphere

In order to be able to compute upper bounds on d^G, we need to impose restrictions on the data space M as well as on the set of classifiers.

1. Depending on the regularity conditions imposed on the classifiers, it might happen that $d^G(c_1,c_2) = 0$ while $c_1 \neq c_2$ on a set of measure zero.

We start with the following simple setup: Let $M = S^n = \{x \in \mathbb{R}^{n+1} \mid \langle x,x\rangle = 1\}$, the n-sphere with its canonical Riemannian metric and denote by C the set of *hemisphere classifiers*

$$c : S^n \to \mathbb{Z}_2, \quad c(x) := \begin{cases} 1 & : \quad \langle x,p\rangle \geq 0 \\ 0 & : \quad \langle x,p\rangle < 0 \end{cases} .$$

Here, $p \in S^n$ is the center of the hemisphere, and $\langle .,. \rangle$ denotes the Euclidean scalar product in $\mathbb{R}^{n+1}$. This setup implies that the data are uniformly distributed on the sphere. The use of closed hemispheres as classifiers is appropriate for spherical data, since hemispheres are the direct analog to half spaces in Euclidean geometry. A simple, yet crucial, observation is the duality $C = S^n$. By a slight abuse of notation, we use the symbol c to denote both, the classifier and the center of the hemisphere. Concerning the generalization distance of two classifiers we have the following proposition.

Proposition 1 *For hemispheres $c_1, c_2 \in C$*

$$d^G(c_1,c_2) = \frac{1}{\pi} d(c_1,c_2),$$

where $d(c_1,c_2) := \arccos(\langle c_1,c_2\rangle)$ is the geodesic distance on S^n.

Proof The disagreement area $D(c_1,c_2)$ consists of two congruent lunes on the sphere. The area of a lune is proportional to its opening angle $\alpha = d(c_1,c_2)$. Since the total volume of the unit sphere $V := \int_{S^n} \omega$ with respect to its canonical Riemannian metric and volume form is not equal to one, we have to normalize:

$$d^G(c_1,c_2) = \frac{1}{V} \int_{D(c_1,c_2)} \omega = \frac{1}{V} \left(\frac{\alpha}{\pi} V \right) = \frac{1}{\pi} d(c_1,c_2).$$

■

We assume the true classifier c^* to be some unknown hemisphere. If $(x,1)$ is a labeled data point, it follows that c^* is contained in the closed hemisphere around x. Thus, given a labeled set $\{(x_1,y_1),\ldots,(x_I,y_I))\}$, c^* is contained in the intersection V of the corresponding hemispheres. The *version space* of a labeled set (see Mitchell, 1982; Tong and Koller, 2001) is defined as the set of classifiers which are consistent with the given labeled data. In our case, the version space coincides with the intersection V. Geometrically, V is a convex spherical polytope, a high-dimensional generalization of a spherical polygon whose edges are segments of great circles and whose $(n-1)$-dimensional facets are segments of $(n-1)$-dimensional great spheres within S^n.

In theory, one can compute the vertices of the version space of any finite labeled set. An iterative algorithm would reduce this polytope by taking intersections with hemispheres corresponding to new data points. Unfortunately, during this process, the number of vertices of the polytope grows exponentially. Thus, the computational costs render its explicit computation impossible even for low dimensions.

One possibility to reduce the immense computational complexity of polytopes is to work with *spherical simplices*. This motivates the following active learning algorithm:

Definition 2 (Simplex algorithm)

1. *Specify some maximal edge length $\varepsilon > 0$ as termination criterion. Choose a random orthogonal matrix. Ask for the labels of the $n+1$ column vectors. This results in a set of admissible classifiers $S \subset S^n$ which is an equilateral simplex on S^n.*
2. *Select one of the edges of maximal length of the current simplex S. If its length is less than ε, go to step 7. Otherwise, compute its midpoint m.*
3. *Compute a unit normal u of the plane in $\mathbb{R}^{n+1}$ spanned by m together with all vertices of S which do not belong to the edge through m.*
4. *Ask for the label l of u. If $l = 0$, replace u by $-u$.*
5. *Replace the old simplex S by the part* in direction of *u, that is, the simplex which has as vertices m as well as all vertices of S with exception of that end point e of the chosen longest edge for which $\langle u, e \rangle < 0$.*
6. *Repeat with step 2.*
7. *Return the simplex' center of mass $c \in S$ as the learned classifier.*

Figure 1 illustrates one iteration of the simplex algorithm on the sphere S^2. The "random orthogonal

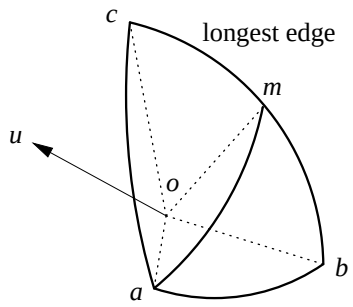


Figure 1: The drawing above shows one iteration of the simplex algorithm for the two-dimensional case, that is, for spherical triangles on the unit sphere S^2. The current version space is the spherical triangle (a,b,c). The longest edge (b,c) is about to be cut at its midpoint m. Together with the origin o, the vertex a and the point m define a plane in $\mathbb{R}^3$ one of whose unit normal vectors is $u \in S^2$. Depending on the label of u, the new triangle will be either (a,b,m) or (a,m,c).

matrix" that occurs in step one of the algorithm is meant to be drawn from the uniform distribution on the Lie group

$$O(n) = \{A \in GL(n) \mid AA^t = I\}$$

of orthogonal real matrices. A practical approach to the problem of generating such matrices can be found in Stewart (1980). The worst case generalization error after each iteration can be computed

by evaluating the maximum spherical distance of the chosen classifier to the vertices of the spherical simplex. To be explicit, the following statement holds:

Proposition 3 *If S is the current simplex from the simplex algorithm (see Definition 2) with vertices $v_1, \ldots, v_{n+1} \in S^n$, and $c \in S$ some classifier,*

$$d^G(c^*, c) \leq \max_{i,j} d(v_i, v_j) = \max_{i,j} \arccos\langle v_i, v_j \rangle.$$

This bound is tight and attainable if we allow any element of the version space to be the learned classifier. Moreover, if $c \in S$ denotes the center of mass, then

$$d^G(c^*, c) \leq \max_{i} d(c, v_i) = \max_{i} \arccos\langle c, v_i \rangle$$

is a tight and attainable upper bound for the generalization error.

Proof Within a simplex, the maximal distance of two points is realized by pairs of vertices. Now the first inequality follows from proposition 1. If all elements of the simplex are admissible classifiers, the bound is tight. The second inequality follows from the convexity of the simplex. ■

Clearly, the maximal edge length of the simplex S converges to zero. In Appendix A, we derive $O((n+1)^3)$ as a rough complexity estimate for one iteration of the simplex algorithm (see Definition 2).

Another question concerning the convergence of the algorithm is: How many iterations are needed until the maximum edge length of the initial simplex starts to drop? To this end, we have the following proposition whose proof is given in Appendix A.

Proposition 4 *Let S be the initial equilateral simplex from the simplex algorithm (see Definition 2). Let $k \in \mathbb{N}$ be the number of steps needed until the maximum of the edge lengths drops. Then*

$$n \leq k \leq \frac{n(n+1)}{2},$$

and these bounds are tight and attainable.

4. Extensions by Isometries

We will now extend our results to other data spaces by applying the concept of isometries. The easiest method to obtain isometries from the n-sphere to other data spaces is to consider charts of the sphere together with the induced metric. Being isometries, they preserve the geometry, and any generalization bounds derived for the sphere can be applied without modifications. Combining isometries with the product construction of Section 5, we end up with a large family of data densities on $\mathbb{R}^n$ to which our results are directly applicable. In Section 6, we will loosen our preconditions even further and consider the general case of arbitrary data densities.

We begin with the discussion of the stereographic chart of the n-sphere.

Stereographic chart. The stereographic projection

$$\sigma_N : S^n \setminus \{N\} \to \mathbb{R}^n,$$
$$(x_1, \ldots, x_{n+1}) \mapsto \frac{(x_1, \ldots, x_n)}{1 - x_{n+1}},$$

where $N = (0,\ldots,0,1)$ denotes the north pole, is an isometry of $S^n \setminus \{N\}$ to $(\mathbb{R}^n, g)$, where the Riemannian metric g is given by

$$g_x(v,w) = \frac{4}{(1+\|x\|^2)^2}\langle v,w\rangle.$$

See Figure 2 for an illustration of the two-dimensional case. Hence, we can *identify* $S^n \setminus \{N\}$ with

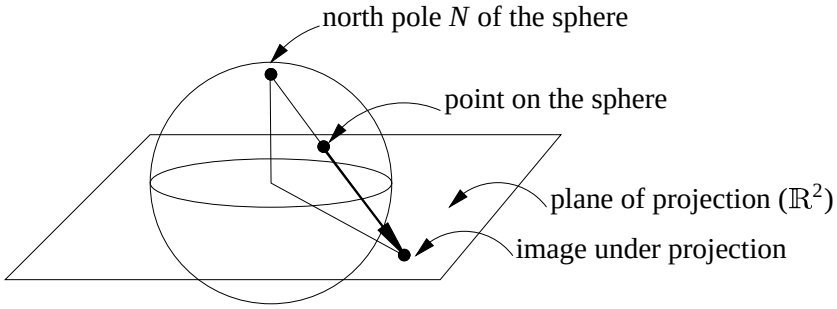


Figure 2: The stereographic projection $S^2 \setminus \{N\} \to \mathbb{R}^2$ is a diffeomorphism from the sphere with the north pole removed to the plane. It distorts lengths but preserves angles. If one considers a ray from the north pole to some point on the plane, the stereographic projection will map the intersection point of this ray with the sphere onto its intersection point with the plane.

$(\mathbb{R}^n, g)$, and the induced Riemannian volume form is

$$\omega_x = \frac{2^n}{(1+\|x\|^2)^n} dx_1 \wedge \ldots \wedge dx_n,$$

where $dx_1 \wedge \ldots \wedge dx_n$ denotes the Euclidean volume on $\mathbb{R}^n$ (the determinant). If the given data density on $\mathbb{R}^n$ is (up to a constant factor) equal to ω, the data can be considered to lie on S^n with constant density, and our error bounds hold. When viewed under stereographic projection, our spherical classifiers fall in three categories: If the boundary of the hemisphere, which is a great $(n-1)$-sphere within S^n, contains the north pole, its projection is a hyperplane through the origin in $\mathbb{R}^n$. The equatorial great sphere $\{x \in S^n \subset \mathbb{R}^{n+1} \mid x_{n+1} = 0\}$ is projected onto $S^{n-1} \subset \mathbb{R}^n$. All other great spheres become spheres intersecting $S^{n-1} \subset \mathbb{R}^n$ orthogonally. Hence, any data which is separable by these classes of hypersurfaces in $\mathbb{R}^n$ is separable by great spheres on S^n and vice versa.

Gnomonic chart. The gnomonic projection

$$\varphi : S^n \supset \{x_{n+1} > 0\} \to \mathbb{R}^n,$$
$$(x_1,\ldots,x_{n+1}) \mapsto \frac{(x_1,\ldots,x_n)}{x_{n+1}},$$

generates on $\mathbb{R}^n$ the metric

$$g = \frac{1}{(1+x_1^2+\ldots+x_n^2)^2} \begin{pmatrix} s_1 & & -x_i x_j \\ & \ddots & \\ -x_i x_j & & s_n \end{pmatrix},$$

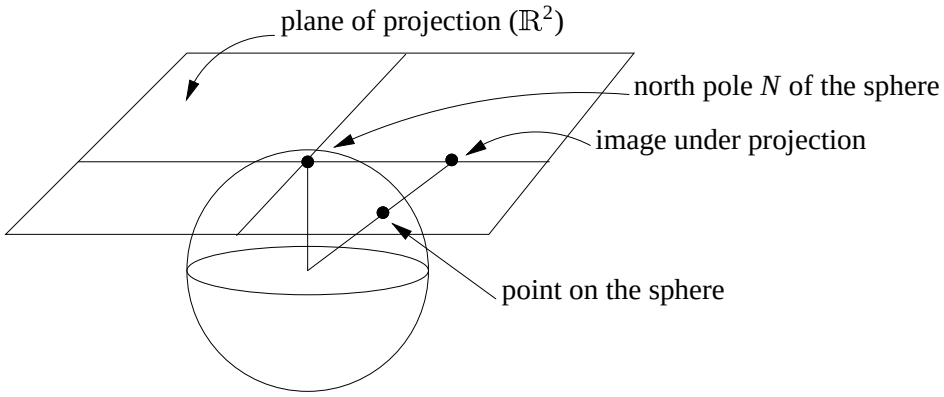


Figure 3: Illustration of the gnomonic projection for the case $S^2 \supset \{x_3 > 0\} \to \mathbb{R}^2$. In this case, we embed $\mathbb{R}^2$ at height one into $\mathbb{R}^3$, such that it touches the sphere at the north pole. Rays are sent from the center of the sphere. Their intersection points with the sphere are mapped to the corresponding intersection points with the plane. The gnomonic projection distorts lengths as well as angles, but maps great circles to straight lines.

where we have used the abbreviation $s_i := 1 + x_1^2 + \ldots + x_n^2 - x_i^2$. Figure 3 illustrates the gnomonic projection in two dimensions. As was the case with the stereographic chart, gnomonic projection is an isometry from the upper half-sphere to $(\mathbb{R}^n, g)$. Using Equation 1, we have for $x_2 = \ldots = x_n = 0$

$$\sqrt{\det(g)} = \frac{1}{(1+x_1^2)^{\frac{n+1}{2}}}.$$

Since the scaling function of the volume form ω has to be rotationally symmetric, it follows that

$$\omega_x = \frac{1}{(1+\|x\|^2)^{\frac{n+1}{2}}}\, dx_1 \wedge \ldots \wedge dx_n. \tag{3}$$

Note that our separating great spheres are projected to affine hyperplanes in $\mathbb{R}^n$. Therefore, the classical approach to the binary classification problem using *linear classifiers with bias* can be considered a special case of our spherical setup. More precisely, the strict error bounds derived for our algorithm apply to linear classifiers with bias on $\mathbb{R}^n$ if and only if the data density on $\mathbb{R}^n$ is given by Equation 3. For an analysis that applies to a greater variety of densities we refer to Section 6.

As a byproduct, formula 3 also clarifies the arguments given in the proof of Theorem 4 of Freund et al. (1997) which estimates the information gain of queries made by the Query by Committee algorithm. In the proof, the scaling factor of the volume form of the gnomonic chart is estimated using infinitesimals. The explicit formula for the volume form given above makes this more lucid.

5. Products of Spheres

We now extend the simplex algorithm (see Definition 2) to other data manifolds and other sets of classifiers using a simple product construction. The main purpose of this section is to obtain

building blocks for data manifolds and data densities which later can be combined to produce more sophisticated examples. We consider product data manifolds of the type

$$(M,g) = (S^{n_1},g^1) \times \ldots \times (S^{n_k},g^k), \tag{4}$$

where each factor (S^{n_j},g^j) is a unit sphere of dimension n_j with its standard metric g^j. For a point $x = (x_1,\ldots,x_k) \in M$ and tangent vectors $X = (X_1,\ldots,X_k), Y = (Y_1,\ldots,Y_k) \in T_xM$ we have

$$g_x(X,Y) = \sum_{j=1}^{k} g^j_{x_j}(X_j,Y_j).$$

The Riemannian volume form of the product is given in local coordinates by

$$\omega = \prod_{j=1}^{k} \sqrt{\det(g^j)} = \bigwedge_{j=1}^{k} \omega^j,$$

where ω^j denotes the volume form of the jth factor. On the product manifold M, we consider classifiers which are products of hemispheres, that is, $C = C^1 \times \ldots \times C^k$, where the $C^j = S^{n_j}$ are the individual sets of hemispheres defined in Section 3 and a classifier $c \in C$ is given by

$$c : M \to \mathbb{Z}_2, \quad c(x) := \begin{cases} 1 & : \quad \langle x, c_j \rangle \geq 0 \, \forall \, j \\ 0 & : \quad \text{otherwise} \end{cases} .$$

Due to the simplicity of the product structure, we arrive at the following formula for the generalization metric:

Proposition 5 *For products of hemispheres $c^1, c^2 \in C$,*

$$d^G(c^1,c^2) = \frac{1}{2^{k-1}} \left(1 - \frac{1}{\pi^k} \prod_{j=1}^{k} (\pi - d_j(c^1_j, c^2_j)), \right)$$

where d_j is the geodesic distance on S^{n_j}.

Proof Since each component of a classifier is a hemisphere,

$$\mathrm{Vol}(c^1_j \cap c^2_j) = \mathrm{Vol}(S^{n_j}) \frac{\pi - d_j(c^1_j, c^2_j)}{2\pi}.$$

Furthermore, the volume of a product of such hemispheres is given by

$$\mathrm{Vol}(c^1) = \prod_{j=1}^{k} \mathrm{Vol}(c^1_j) = \prod_{j=1}^{k} \frac{\mathrm{Vol}(S^{n_j})}{2} = \frac{\mathrm{Vol}(M)}{2^k}.$$

Inserting this into

$$\mathrm{Vol}(D(c^1,c^2)) = \mathrm{Vol}(c^1) + \mathrm{Vol}(c^2) - 2\mathrm{Vol}(c^1 \cap c^2)$$

where $D(c^1,c^2)$ denotes the disagreement area, yields the proposition. ■

This leads to the extended spherical simplex algorithm:

Definition 6 (Extended simplex algorithm)

1. *Specify some maximal edge length* $\varepsilon > 0$ *as termination criterion. For each factor* S^{n_j}*, create an initial simplex using a random orthogonal matrix whose columns are interpreted as a basis for* $\mathbb{R}^{n_j+1}$*. This results in a product S of equilateral simplices.*

2. *Find one of the edges of maximal length of each factor of the current simplex product S. If all the respective lengths are less than* ε*, go to step 7. Otherwise, compute the midpoints* $m_1, \ldots, m_k$*.*

3. *Compute the corresponding unit normals* u_j*.*

4. *Ask for the labels* l_j *of* u_j*. If* $l_j = 0$*, replace* u_j *by* $-u_j$*.*

5. *Replace the old simplex product S by the product of the parts* in direction of u_j*.*

6. *Repeat with step 2.*

7. *For each factor, compute its center of mass* c_j*, and return* $(c_1, \ldots, c_k) \in S$ *as the learned classifier.*

In parallel to the case of a single sphere, the minimization of maximal edge lengths forces convergence. Note, that if k denotes the number of factors in the product of spheres, then k training points are needed to carry out one iteration of the algorithm. The worst case generalization error after each step is bounded as follows.

Proposition 7 *If S is the current product simplex from the extended simplex algorithm (see Definition 6) with maximal edge lengths* $d_1, \ldots, d_k$ *of its factors, and* $c \in S$ *some classifier,*

$$d^G(c^*, c) \leq \frac{1}{2^{k-1}} \left(1 - \frac{1}{\pi^k} \prod_{j=1}^{k} (\pi - d_j) \right).$$

This bound is tight and attainable if we allow any element of the version space to be the learned classifier.

Proof In analogy to the case of a single sphere, this follows from proposition 5. ■

The complexity estimate for one iteration of the extended simplex algorithm is

$$O((n_1+1)^3 + \ldots + (n_k+1)^3).$$

Here, $n_1, \ldots, n_k$ denote the dimensions of the k individual factors of the product data space. This estimate can be deduced directly from the complexity analysis of the simplex algorithm given in Appendix A.

Products combined with isometries. We now apply the isometries discussed in Section 4 to product manifolds. For each factor in Equation 4, we may choose one of stereographic or gnomonic projection.

$$\begin{array}{rccc} M^{n_1+\ldots+n_k} = & S^{n_1} & \times\ldots\times & S^{n_k}, \\ & f_1\downarrow & \cdots & \downarrow f_k \\ \mathbb{R}^{n_1+\ldots+n_k} = & \mathbb{R}^{n_1} & \times\ldots\times & \mathbb{R}^{n_k}. \end{array}$$

This results in product densities on $\mathbb{R}^n = \mathbb{R}^{n_1+\ldots+n_k}$, represented by the volume form

$$\omega = \prod_{j=1}^{k} \omega^j$$

with factors ω^j given by either

$$\omega_x^j = \frac{2^{n_j}}{(1+\|x\|^2)^{n_j}} dx_1 \wedge \ldots \wedge dx_{n_j}$$

(stereographic projection) or

$$\omega_x^j = \frac{1}{(1+\|x\|^2)^{\frac{n_j+1}{2}}} dx_1 \wedge \ldots \wedge dx_{n_j}$$

(gnomonic projection). Similarly, the projected separating hypersurfaces are products of the individual projections. One could now go on to produce *many* more families of compatible densities by working with different charts, but instead we turn our attention to an important special case.

The *n*-torus. A particular case is the gnomonic projection of the *n*-torus

$$T^n = \underbrace{S^1 \times \ldots \times S^1}_{n \text{ factors}},$$

which yields a scaled version of the Cauchy distribution on $\mathbb{R}^n$:

$$\omega_x = \prod_{i=1}^{n} \frac{1}{1+x_i^2} dx_1 \wedge \ldots \wedge dx_n.$$

Here, we take S^1 to be the unit circle which results in T^n having total mass $(2\pi)^n$. Since there is one circle S^1 per axis, the projected classifiers are *axis parallel corners* in $\mathbb{R}^n$. One iteration of the algorithm consumes n labeled data points, because T^n is made up of n individual factors. At each step of the extended simplex algorithm, the version space is a *hypercube* of equal edge length l. Therefore, we do not need to compare edge lengths. One step results in halving all the edges, and the volume is divided by 2^n. Hence, if V_i denotes the volume after i iterations, we have

$$V_i = \frac{\pi}{2^{in}}.$$

In analogy, the maximal generalization error G_i of the center of mass classifier after i steps is given by

$$G_i = \frac{\sqrt{n}\pi}{2^i}.$$

Thus, we observe an exponential decrease in volume and in the tight upper bound for the generalization error.

6. Aspherical Data Manifolds

Up to now, our methods apply only to such data manifolds which are isometric to products of subsets of unit spheres. We now loosen this assumption by looking at all oriented Riemannian data manifolds M which admit an orientation preserving[2] diffeomorphism

$$\Phi : M \to S^n,$$

that is, Φ is a smooth bijective map which has a smooth inverse. The Riemannian volume form belonging to the metric g on M induces a volume form $\widetilde{\omega}$ on S^n which, in general, is not equal to the spherical volume ω. More precisely,

$$\widetilde{\omega} = f\omega$$

with some smooth positive scaling function $f : S^n \to \mathbb{R}^+$. Note that *all* volume forms (or smooth positive densities) on S^n can be written in this form. This results in the formula

$$d^G(c_1, c_2) := \frac{1}{\widetilde{\mathrm{Vol}}(S^n)} \int_{D(c_1,c_2)} f\omega$$

for the generalization metric. An illustation of a non-uniform density on the sphere is given in Figure 4 in Section 8 where the reader also finds empirical results for this case.

Concerning the set of admissible classifiers, let us keep the assumption that the Φ-image of the data is separable by hemisphere classifiers. If we know that the true classifier c^* lies in some subset $S \subset S^n$ the worst case generalization error of some classifier $c \in S$ is bounded from above by

$$d^G(c, c^*) \leq \sup_{\widetilde{c} \in S} d^G(\widetilde{c}, c^*).$$

Therefore, the simplex algorithm (see Definition 2) will still converge, as it reduces an upper bound of the generalization error. Its rate of convergence will depend on the properties of the density.

Nevertheless, we can force a simple upper bound by assuming the deviation of the induced volume form from spherical volume to be small:

$$\sup_{x \in S^n} |1 - f(x)| < \varepsilon.$$

This implies

Proposition 8 *Let ω be the canonical volume form of S^n. Denote by $d^{\widetilde{G}}$ the generalization distance induced by the scaled volume form $\widetilde{\omega} = f\omega$ where $f : S^n \to \mathbb{R}^+$ is some positive smooth scaling function. If $\sup_{x \in S^n} |1 - f(x)| < \varepsilon$ for some $\varepsilon > 0$ then*

$$d^{\widetilde{G}}(c_1, c_2) \leq \frac{(1+\varepsilon)\mathrm{Vol}(S^n)}{\pi \widetilde{\mathrm{Vol}}(S^n)} d(c_1, c_2),$$

where d is the canonical geodesic distance of S^n. In this formula, $\mathrm{Vol}(S^n) := \int_{S^n} \omega$ and $\widetilde{\mathrm{Vol}}(S^n) := \int_{S^n} \widetilde{\omega}$ denote the volumina of S^n with respect to ω and $\widetilde{\omega} := f\omega$.

2. A map between oriented Riemannian manifolds is orientation preserving if its functional determinant is positive.

Proof Using the definition of the generalization distance in Equation 2 and applying proposition 1, we compute

$$\begin{aligned} d^{\widetilde{G}}(c_1,c_2) &= \frac{1}{\widetilde{\mathrm{Vol}}(S^n)} \int_{D(c_1,c_2)} f\omega \\ &\leq \frac{1}{\widetilde{\mathrm{Vol}}(S^n)} \int_{D(c_1,c_2)} (1+\varepsilon)\omega \\ &= \frac{(1+\varepsilon)\mathrm{Vol}(S^n)}{\widetilde{\mathrm{Vol}}(S^n)} d^G(c_1,c_2) \\ &= \frac{(1+\varepsilon)\mathrm{Vol}(S^n)}{\pi\widetilde{\mathrm{Vol}}(S^n)} d(c_1,c_2). \end{aligned}$$

■

Inserting the above upper bound into proposition 3 we obtain

Proposition 9 *Let S be the current simplex from the simplex algorithm (see Definition 2) with vertices $v_1,\ldots,v_{n+1} \in S^n$, and $c \in S \subset S^n$ an arbitrary classifier, not necessarily the center of mass. If $c^* \in S^n$ denotes the unknown true classifier, the generalization error of c is bounded by*

$$d^{\widetilde{G}}(c,c^*) \leq \frac{(1+\varepsilon)\mathrm{Vol}(S^n)}{\pi\widetilde{\mathrm{Vol}}(S^n)} \max_{i,j} d(v_i,v_j),$$

where $d(v_i,v_j)$ denotes the spherical distance of the vertices.

The usefulness of the above proposition depends on how much the scaled volume form $f\omega$ deviates from the canonical spherical volume ω. In the case of the n-torus, the same arguments as given at the end of Section 4 yield an exponential decrease of the volume of the version space as well as of the upper bound for the generalization error—*regardless of the data density under consideration*. The only difference is the newly introduced constant ε which may affect the absolute rate but not the functional form of convergence.

7. Linear Classifiers without Bias

We now return to the case of linear classifiers on the Euclidean space $\mathbb{R}^n$ which commonly appear in the machine learning literature. The corresponding separating hypersurfaces are linear subspaces, that is, hyperplanes through the origin, of $\mathbb{R}^n$.

Consider the data space $M = \mathbb{R}^n$. Then, the set C of *linear classifiers without bias* consists of maps

$$c : \mathbb{R}^n \to \mathbb{Z}_2, \quad c(x) := \begin{cases} 1 & : \quad \langle x,p\rangle \geq 0 \\ 0 & : \quad \langle x,p\rangle < 0 \end{cases},$$

where $p \in S^n$ is the unit normal vector of an oriented $(n-1)$-dimensional plane through the origin. Therefore, we can identify C with the unit n-sphere, $C = S^n$. In the following, we use c to denote both, the classifier and its corresponding unit normal.

Consider some data density $f : \mathbb{R}^n \to \mathbb{R}$. Since the origin is always classified as $+1$ by all classifiers we may remove it from the data space, and the remaining data space is given by

$$M = \mathbb{R}^n \setminus \{0\}.$$

Using polar coordinates (s,r), we can write M as the product[3]

$$M = S^{n-1} \times \mathbb{R}^+, \quad s = \frac{x}{\|x\|}, r = \|x\|.$$

For a given $s \in S^{n-1}$, we will call the subset $F_s := \{(s,r) \mid r > 0\}$ the *fiber* over s. Since the data is assumed to be separable by at least one element of C any two points belonging to the same fiber have the same label.

The area $D(c_1,c_2)$ of disagreement between two classifiers c_1, c_2 is given by

$$D(c_1,c_2) = \{(s,r) \in M \mid \langle c_1, rs\rangle\langle c_2, rs\rangle < 0\} = \{(s,r) \in M \mid \langle c_1, s\rangle\langle c_2, s\rangle < 0\}.$$

The generalization distance is given by

$$d^G(c_1,c_2) = \int_{D(c_1,c_2)} f dx = \int_{\{s \in S^{n-1} \mid F_s \subset D(c_1,c_2)\}} \left(\int_{F_s} f(s,.) dr \right) ds, \tag{5}$$

where dx, dr, ds denote the canonical volume forms on $\mathbb{R}^n$, F_s, and S^{n-1}. It may happen that different fibers F_s have different mass in the sense that

$$S^{n-1} \to \mathbb{R}, \quad s \mapsto \int_{F_s} f(s,.) dr$$

is a non-constant function. If we rule out this case we end up with the following proposition:

Proposition 10 *The generalization distance of any two linear classifiers c_1, c_2 is given by*

$$d^G(c_1,c_2) = \frac{\lambda}{\pi} d(c_1,c_2),$$

where $d(c_1,c_2) = \arccos\langle c_1, c_2\rangle$ is the geodesic distance on S^n, if and only if the fiber mass is equal to a positive constant,

$$\int_{F_s} f(s,.) dr = \lambda > 0 \ \ \forall s \in S^n.$$

Proof This follows by applying proposition 1 to Equation 5. ■

The precondition of proposition 10 does *not* assume that the density f is rotationally invariant on $\mathbb{R}^{n+1}$. Instead, it assumes the *accumulated* density to be invariant on the sphere. Linear classification problems on non-constant densities which fulfill this condition map to classification problems with hemisphere classifiers for the uniform density on the sphere. Consequently, all results derived for the spherical simplex algorithm (see Definition 2) apply, including the hard upper bounds on the generalization error. In particular, we deduce the following result from Proposition 3

3. We use the convention $\mathbb{R}^+ := \{r \in \mathbb{R} \mid r > 0\}$.

Proposition 11 *If S is the current simplex from the simplex algorithm (see Definition 2) with vertices $v_1, \ldots, v_{n+1} \in S^n$, c^* denotes the unknown true classifier and $c \in S$ an arbitrary classifier,*

$$d^G(c^*, c) \leq \lambda \max_{i,j} d(v_i, v_j) = \lambda \max_{i,j} \arccos\langle v_i, v_j \rangle.$$

As in Proposition 10, $\lambda > 0$ denotes the fiber mass. This bound is tight and attainable if we allow any element of the version space to be the learned classifier. Moreover, if $c \in S$ denotes the center of mass, then

$$d^G(c^*, c) \leq \lambda \max_{i} d(c, v_i) = \lambda \max_{i} \arccos\langle c, v_i \rangle.$$

is a tight and attainable upper bound for the generalization error.

Relation to SVM methods. Proposition 10 also sheds new light on some active learning strategies that use Support Vector Machines (SVM). A SVM classifier can be interpreted as an approximation of the center of the largest inscribable hypersphere of the version space on S^{n-1} (see Herbrich, 2002). Let us denote this center by $p^* \in V \subset S^{n-1}$, where $V \subset S^{n-1}$ is the version space (a spherical polytope, see Section 3) on the hypersphere.

Tong and Koller (2001) argue that, despite its dependence on the particular shape of the version space, the center of the maximal inscribable hypersphere often lies close to "the center of the version space". Motivated by these insights, they propose the following pool-based strategy for selecting an unlabeled data point $x \in S^{n-1}$ to be labeled: Choose x such that the (spherical) distance from the $(n-2)$-dimensional great sphere

$$X := \{s \in S^{n-1} \mid \langle x, s \rangle = 0\}$$

to p^* is minimal. After the data point x is labeled, the version space will be cut into two pieces along the great sphere $X \subset S^{n-1}$. The goal of this strategy is to reduce the volume of the spherical polytope V as quickly as possible. Similar strategies can be found in Warmuth et al. (2002). Up to now, a closed formula for the volume of a n-spherical simplex *is not known*, not to speak of polytopes, hence, there is *no way* of computing the exact volume of a version space on a hypersphere. We refer to Milnor (1994) for a detailed discussion of this topic. Nevertheless, Monte-Carlo methods can be applied to obtain volume estimates.

Proposition 10 can now be applied. The SVM algorithm works with the Euclidean scalar product on $\mathbb{R}^n$, and therefore implicitly assumes the canonical Riemannian metric and volume form on the unit sphere. Proposition 10 tells us that the SVM approximation is theoretically justified if and only if the given data density induces (up to a constant factor) the uniform density on the sphere.

8. Experimental Results

The following results were obtained from a C++ implementation of the simplex algorithm (see Definition 2). The numerically most sensitive operation of the algorithm is the computation of a normal vector $u \in S^n$ to the hyperplane $H \subset \mathbb{R}^{n+1}$ whose intersection $I = H \cap S^n$ with S^n cuts the current simplex S into two pieces. In higher dimensions, say $n > 50$,

$$\int_{S^n \setminus \{x \in S^n \mid d(x,I) < \delta\}} \omega$$

becomes very small even for small values $\delta > 0$.[4] This means, nearly all the mass of S^n is concentrated within a thin tube of radius δ around I, which makes it hard to compute normal vectors. We avoid numerical problems by using the following procedure:

1. Select the basis of H given by the midpoint of the longest edge and all vertices of the simplex excluding the end points of the longest edge. Construct a corresponding orthonormal basis using the modified Gram-Schmidt algorithm (see Meyer, 2000).

2. Choose random points $x \in S^n$ until the projected length $\sqrt{\sum_i \langle x, h_i \rangle}$, where h_i are the orthonormal basis vectors of H, is less than a given constant $\varepsilon < 1$.

3. Use x to construct a unit vector which is orthogonal to all basis vectors h_i.

In order to test the performance of the simplex algorithm on spheres of different dimensions a series of numerical experiments was conducted, where the following quantities were measured.

Maximal edge length: If $(v_1, \ldots, v_{n+1})$ denote the vertices of the current simplex S of the simplex algorithm, the maximal edge length

$$\max_{i,j} d(v_1, v_j)$$

is an upper bound on the generalization error, regardless which point of the simplex is chosen as the learned classifier.

Maximal distance from center of mass: Let $c \in S$ denote the center of mass of S. Then the maximal distance between c and any other classifier from S is given by

$$\max_i d(c, v_i).$$

This yields a tight upper bound on the generalization error if we choose the center of mass as the learned classifier.

Approximate generalization error for the center of mass classifier c: We estimated the generalization error of c using the empirical average of the individual errors for 50,000 randomly selected "test" data points. Test data were sampled (i) from a uniform density on the sphere and (ii) from an aspherical density with two distinct "clusters" at opposite poles. The aspherical density was constructed by mapping the uniform distribution from an open parameter cuboid onto the sphere using n-spherical coordinates.

Figure 4 illustrates the relation between a sample drawn from this density on the sphere S^2 and its stereographic projection onto the plane $\mathbb{R}^2$. Densities of this kind are typical for binary classification problems. Any density with two peaks at $p_1, p_2 \in \mathbb{R}^n$ can be identified with a density on S^n with peaks at opposite poles: Firstly, we apply a translation to move the midpoint of the line $\overline{p_1 p_2}$ to the origin. Then we use a rotation to place p_1, p_2 on the x_1-axis. After a scaling, $p_1 = (-1, 0, \ldots, 0)$ and $p_2 = (+1, 0, \ldots, 0)$. Now inverse stereographic projection will map the peaks onto opposite poles.

4. For a comprehensive discussion of these effects we refer to Gromov (1999).

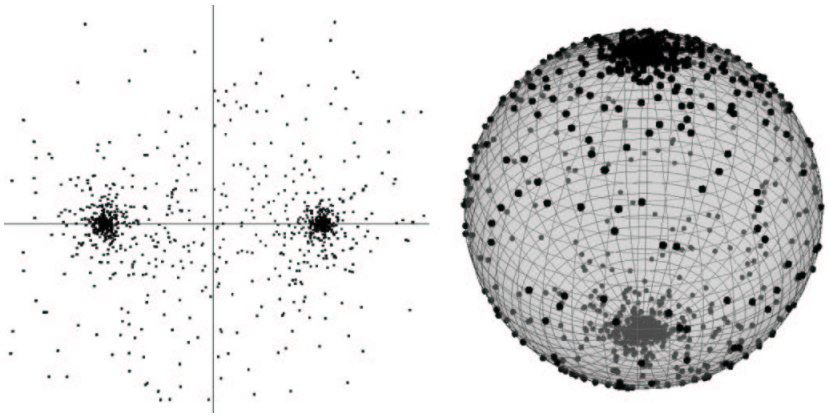


Figure 4: A data sample from a density with two peaks on $\mathbb{R}^2$ (left) and S^2 (right). The data points on the plane are the stereographic projections of the points on the sphere. On S^2, the two peaks are located at opposite poles.

The above quantities were computed for each step of the simplex algorithm. Averages and variances were calculated for 1,000 simulations, and averages were normalized to lie within the interval $[0,1]$. For every simulation, a true classifier was drawn from the uniform distribution on the sphere. Figures 5 and 6 show the resulting learning curves for the spheres $S^9 \subset \mathbb{R}^{10}$, $S^{29} \subset \mathbb{R}^{30}$, $S^{49} \subset \mathbb{R}^{50}$, and $S^{79} \subset \mathbb{R}^{80}$. The average maximal edge length as a function of the number of selected training data

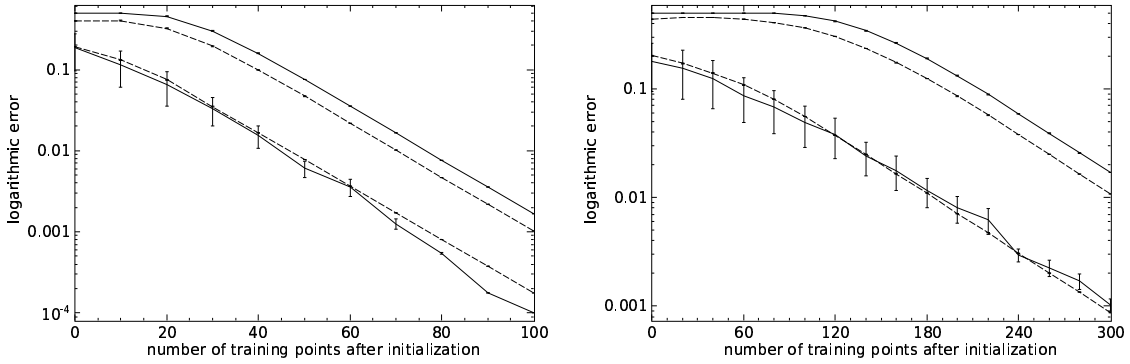


Figure 5: Learning curves on $S^9 \subset \mathbb{R}^{10}$ (left) and on $S^{29} \subset \mathbb{R}^{30}$ (right). The figures show the average maximal edge length (upper solid line), the average maximal distance from the simplex's center of mass (upper dashed line), and the average approximate generalization errors for the uniform (lower dashed line) and aspherical (lower solid line) data densities as a function of the number of selected training examples. Error bars indicate variances, however, only the approximate generalization error for the aspherical data density shows large fluctuations between simulation runs. Proposition 4 yields the bounds $9 \leq k_9 \leq 45$ and $29 \leq k_{29} \leq 435$ for the number k of steps needed before the maximal edge length starts to drop on S^9 and S^{29}.

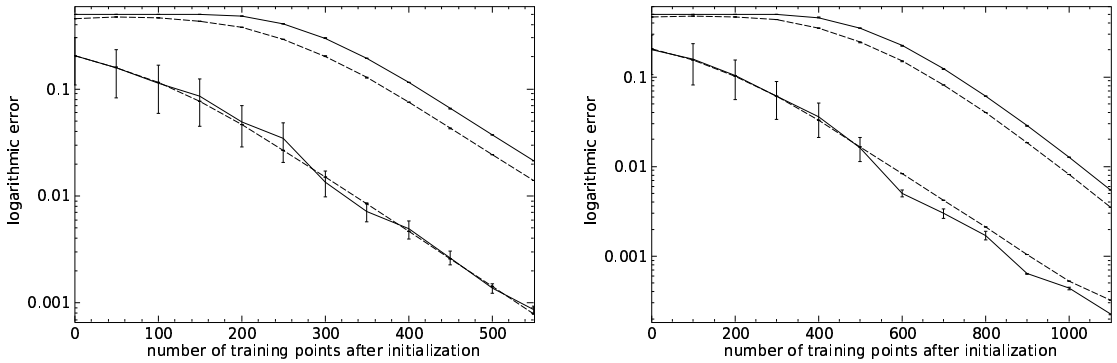


Figure 6: Learning curves on $S^{49} \subset \mathbb{R}^{50}$ (left) and $S^{79} \subset \mathbb{R}^{80}$ (right). For details see legend of Figure 5. Proposition 4 yields the bounds $49 \leq k_{49} \leq 1225$ and $79 \leq k_{79} \leq 3160$ for the number k of steps needed before the maximal edge length starts to drop on S^{49} and S^{79}.

shows an initial plateau, until the values begin to decrease in an approximately exponential fashion. The length of the plateau increases with the dimensionality of the sphere and is a direct result of Proposition 4. The average maximal distance from the center of mass rises initially (see Figure 7 for a magnified version of the initial segment of the learning curve), until a sudden drop occurs, again followed by a roughly exponential decrease. This can be explained as follows. The simplex algorithm is initialized with an equilateral simplex. During the first learning steps, the center of mass moves towards those vertices whose adjacent edges are cut already. This results in a slight increase of the maximal distance of the vertices from the center of mass. The simplex becomes a "thin pyramid" with small base, and the following drop in the plots then corresponds to a cut of a line connecting the apex to the base. The ratio between the edges connecting the apex to the base and the edges which are contained within the base is given by $\frac{2}{n-1}$, where n is the dimensionality of the sphere. Since this number tends to zero for $n \to \infty$ the sudden drop disappears in higher dimensions.

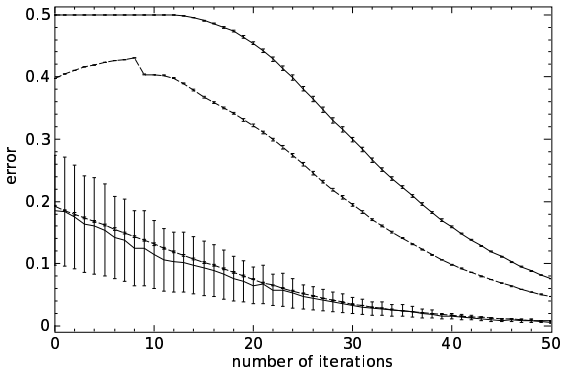


Figure 7: Initial learning curves on $S^9 \subset \mathbb{R}^{10}$ (see Figure 5, left), now plotted on a linear scale. For details see legend of Figure 5.

If the data is drawn from the spherical distribution, the approximate generalization error changes smoothly with the number of selected training data, and its variance is very small. For data distributed according to the aspherical (two cluster) density, the average approximate generalization error is similar, but the variance increases dramatically. Nevertheless, the numerical experiments show that the spherical simplex algorithm performs well even in the case of non-uniform densities.

Experimental results on product manifolds. So far, we restricted the experiments to single spheres instead of products, because the simulation of the extended simplex algorithm on

$$M = S^{n_1} \times \ldots \times S^{n_k},$$

is equivalent to the parallel execution of several copies of the basic algorithm. Nevertheless, it might be interesting to consider the special case of the n-torus T^n (see Section 5):

$$T^n = \underbrace{S^1 \times \ldots \times S^1}_{n \text{ factors}}.$$

The product structure of the torus reflects the fact that data is distributed independently on each factor. For the standard product density on T^n, volume and distances can be computed explicitly. Therefore, we consider only the non-uniform case. We consider a von Mises density (see Devroye, 1986) on the unit circle:

$$f : S^1 = \mathbb{R}/2\pi \to \mathbb{R}, \quad f(x) = \frac{\exp(\kappa \cos(x-\mu))}{2\pi I_0(\kappa)}.$$

with center $\mu \in [0, 2\pi]$ and width $\kappa \geq 0$. The symbol I_0 in the equation above denotes the modified Bessel function of the first kind of order zero. A technique for simulating the von Mises distribution can be found in Best and Fisher (1979). In order to obtain a density with two peaks on opposite poles of the circle, we superimpose two copies of f with $\mu = 0$ and $\mu = \pi$. This construction is applied to every factor S^1 of the torus $T^n = S^1 \times \ldots \times S^1$.

We implemented the extended simplex algorithm on the n-torus. Due to the product structure, numerical problems like those described at the beginning of Section 8 do not arise. For the case $n = 2$, the torus can be embedded into $\mathbb{R}^3$ using

$$S^1 \times S^1 \to \mathbb{R}^3, \quad (s,t) \mapsto \begin{pmatrix} (2+\cos t)\cos s \\ (2+\cos t)\sin s \\ \sin t \end{pmatrix}.$$

Using this mapping, we can visualize the iterations of the extended simplex algorithm on von Mises distributed data. Figure 8 shows a data sample as well as several iterations of the algorithm on the embedded torus. Figure 9 depicts the stereographic projection of a sample drawn from the von Mises distribution together with the projected classifier in $\mathbb{R}^2$.

Finally, we conducted experiments on the n-torus in order to obtain learning curves analogous to those on the n-sphere. As was shown in Section 5 distances and volumina on the n-torus can be computed explicitly provided the data density is uniform. Therefore, we focus on the approximal generalization error for data distributed according to the modified von Mises density described above. The approximation was done by evaluating the performance of the classifier on a data sample of 50,000 test points after each training step. The resulting values were averaged over 1,000

Figure 8: The extended simplex algorithm on the 2-torus. The large dot on the upper part of the torus represents the true classifier. Small dots depict positively (light gray) and negatively (dark) classified points drawn from the modified von Mises distribution. The meaning of the nested regions is the following (light to dark): positively classified area of the true classifier, version space after initialization (step one of the extended simplex algorithm), version space after first iteration, version space after second iteration.

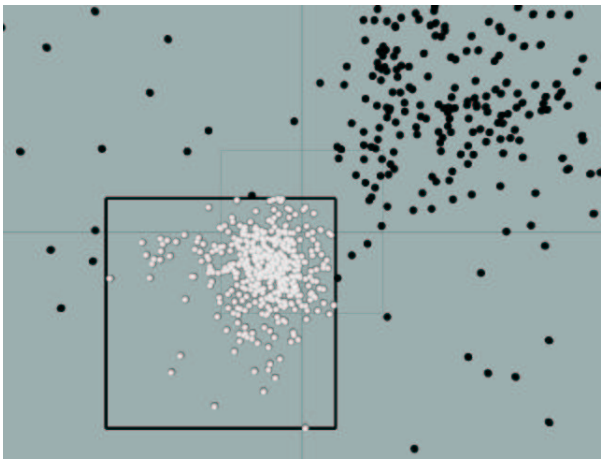


Figure 9: The image of a data sample from two superposed von Mises distributions on the two-dimensional torus under the stereographic projection (defined in Section 4) to $\mathbb{R}^2$. The black square represents the projected classifier. Light dots are classified positively, dark dots negatively.

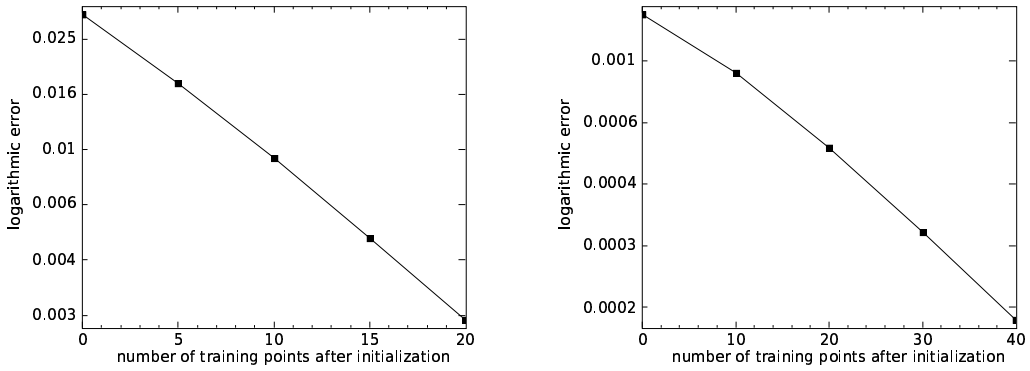


Figure 10: Learning curves on the 5-dimensional torus (left) and on the 10-dimensional torus (right). The figures show the average approximate generalization errors for the modified von Mises density as a function of the number of selected training examples. Since the variances are almost zero, they are not included in the diagram.

simulations. For every simulation, a true classifier was drawn from the uniform distribution on the torus. The resulting learning curves for dimensions $n = 5$ and $n = 10$ are shown in Figure 10.

Due to the product structure of the torus, volumina of rectangular subsets are given by the products of their side lengths. For higher dimensions, the volume of the initial version space becomes very small. Therefore, the initialization of the extended simplex algorithm yields a classifier whose error is by far smaller than the average generalization error of its spherical counterpart. This effect reflects the *statistical independence* of the data which makes the learning task a lot easier. For dimensions $n > 15$, the initialization of the algorithm alone provides a classifier with almost vanishing average generalization error. As the curves depicted in Figure 10 show the error decreases exponentially.

9. Conclusion

In this contribution we provided exact upper bounds for the generalization performace of binary classifiers. In order to do so, we used an active learning scheme for model selection, and we designed a constructive method which reduces such a bound by successive subdivisions of a version space.

The algorithm was first formulated for the generic case of a binary classification problem, where data lies on a n-dimensional hypersphere and where both classes are separable using $(n-1)$-dimensional great spheres as classifiers. We derived tight upper bounds for the case that the density of data is constant (cf. Proposition 3) as well as for cases, where at least an upper bound of the deviation from the constant density is known (cf. Proposition 9).

We then showed, using the concept of isometries, that abovementioned results are not restricted to hyperspherical data spaces. We showed that if a data space can be mapped onto (a subset of) a hypersphere using an isometry, the constructive active learning method can be applied and Propositions 3 and 9 remain valid and can be used to calculate the bound. In particular, the constructive algorithm can be applied to linear classification in the widely used Euclidean data space $\mathbb{R}^n$, and the corresponding bounds hold. A further extension to binary classification on products of spheres

is straightforward. As a simple example, we considered binary classification on products of circles and proved the exponential decrease of the upper bound for arbitrary densities. Using isometries we showed that this problem can be mapped, for example, onto a binary classification problem in $\mathbb{R}^n$ with axis-parallel hypercube classifiers for which the same exponential decrease holds.

The theoretical results were illustrated using a number of classification tasks using flat as well as non-constant densities, and the derived bounds were compared with the classification error on a test set as a standard method for assessing prediction quality. Since our focus lies on a theoretical analysis of active learning methods (the constructive methods being a vehicle of this analysis), an empirical evaluation and applications of the proposed algorithm to real world problems are of second importance here. Still, a few comments can be made. The computational complexity of the method is $O((n+1)^3)$ where n is the dimension of the hypersphere, hence the method works in practice. Empirically, it also provides good results for non-constant densities. The main current limitation, however, is the restriction of the method to separable classification problems.

Acknowledgments

This work was funded in part by the Deutsche Forschungsgemeinschaft (DFG grant Ob 102/10–2).

Appendix A.

The purpose of this appendix is to give a more detailed analysis of the complexity of the simplex algorithm (see Definition 2) as well as a proof of Proposition 4.

Complexity analysis. We first consider step two. The edge lengths

$$d(v_i, v_j) := \arccos(\langle v_i, v_j \rangle),$$

between vertices v_i, v_j of the simplex must be computed in order to determine which edge is to be cut next. To reduce the number of scalar products that actually need to be evaluated we keep a record of all edge lengths of the current simplex. After step 2, the current simplex is cut by a plane through the midpoint m of some edge (a,b). Assume b gets thrown out. Then all $\frac{n(n-1)}{2}$ edges of the facet opposite to b stay untouched. Further, the length of the new edge (a,m) is one half of the length of (a,b). It is left to compute the lengths of all other edges that contain m. Therefore, we need to compute

$$\frac{n(n+1)}{2} - \frac{n(n-1)}{2} - 1 = n-1$$

scalar products of vectors in $\mathbb{R}^{n+1}$ which gives us an additive term of order $O(n-1)$. In step three, one has to apply an orthonormalization procedure. The modified Gram-Schmidt algorithm (see Meyer, 2000) gives us another summand $O((n+1)^3)$. Since the computational complexity of the other steps is negligible we obtain $O((n+1)^3)$ as a rough complexity estimate for one iteration of the simplex algorithm (see Definition 2). Steps one and seven are performed only once. The initialization by choosing a random orthogonal matrix can be implemented by using an algorithm of Stewart (1980). The complexity of this algorithm is $O(n^2)$ plus the time needed for generating n pseudo-random vectors according to the standard normal distribution. The computation of the center of mass in step seven amounts to adding up all vertex vectors of the current simplex and normalizing their sum to length one. Hence, $O((n+1)^3)$ is a complexity estimate for the final step.

We now restate and prove Proposition 4 from Section 3:

Proposition 12 *Let S be the initial equilateral simplex from the simplex algorithm (see Definition 2). Let $k \in \mathbb{N}$ be the number of steps needed until the maximum of the edge lengths drops. Then*

$$n \leq k \leq \frac{n(n+1)}{2},$$

and these bounds are tight and attainable.

Proof The proof consists of four steps:

1. n is a lower bound: Assume $k \leq n-1$ and do k iterations of the algorithm. Since the degree of each vertex is n, each vertex of the initial simplex is end point of an edge of full length. Thus, if one of these initial vertices is contained in the new subsimplex, the subsimplex contains the adjacent edge of full length, too. The $k \leq n-1$ subdivisions have created at most $n-1$ new vertices. Thus, the new subsimplex contains at least two vertices of the initial simplex. Hence, its maximal edge length is still $\frac{\pi}{2}$.

2. The lower bound is tight: Choose some vertex e. Subdivide all n edges adjacent to e and keep the subsimplex containing the vertex e. All edges starting from e now have length $\frac{\pi}{4}$. The angle enclosed by any two edges at e is $\frac{\pi}{2}$. Now the spherical law of cosines tells us that all edges *not* adjacent to e have length $\frac{\pi}{3}$. This implies that the constructed subsimplex realizes the lower bound.

3. $\frac{n(n+1)}{2}$ is an upper bound: This is clear since $\frac{n(n+1)}{2}$ is the number of edges of the simplex.

4. The upper bound is tight: This is clear for $n = 1$.

The induction step $(n-1) \to n$ goes as follows: Use $\frac{n(n-1)}{2}$ steps to subdivide a facet F of the simplex. Then all edges contained in F are shortened, while the n edges connecting F with the opposite vertex e still have full length. Now subdivide the connecting edges, and always choose the subsimplex which contains e. In this case, e is the only common vertex belonging to the newly subdivided edge and the rest of the edges of full length. Hence, in each of these last n steps, only one edge length is reduced. An illustration of this case is shown in Figure 11. ■

Appendix B.

The purpose of this appendix is to introduce some differential geometric notions used in the main text. For a comprehensive treatise of Riemannian manifolds we refer to Gallot et al. (1990).

A *manifold* M is a generalization of Euclidean space $\mathbb{R}^n$. It is covered by *coordinate charts*, that is, bijective maps $u : U \to \mathbb{R}^n$, where $U \subset M$ is an open subset. The inverse of u is called a *parametrization*. For our work, the most important example of a manifold is the n-sphere $S^n = \{p \in \mathbb{R}^{n+1} \mid \|p\| = 1\}$. It can be covered by two charts, stereographic projection from the north and south pole. Another system of charts is given by the gnomonic projections. Both are discussed in detail in Section 4.

At each point $p \in M$, the manifold is approximated by its *tangent space* T_pM, which generalizes the tangent of a smooth curve. In the case of S^n, the space T_pS^n can be identified with the linear subspace

$$T_pS^n = \{X \in \mathbb{R}^{n+1} \mid \langle p, X \rangle = 0\}.$$

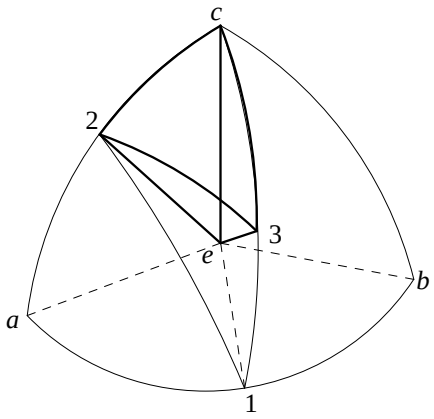


Figure 11: The figure shows a subdivision of a spherical simplex on $S^3 \subset \mathbb{R}^4$ under stereographic projection (see Section 4). The initial simplex, a tetrahedron, is (a,b,c,e). All of its edges have *spherical* length $\frac{\pi}{2}$. After three iterations, indicated by their midpoints $1,2,3$, the subsimplex with edges drawn in bold face still contains three edges (those starting from vertex e) of full length.

A *Riemannian metric* is a choice of a scalar product g_p for each tangent space T_pM. The pair (M,g) is called a *Riemannian manifold*. The canonical Riemannian metric of S^n is given by the restriction of the Euclidean scalar product on $\mathbb{R}^{n+1}$ to the subspace T_pS^n. Given some parametrization $f : \mathbb{R}^n \to U \subset S^n \subset \mathbb{R}^{n+1}$ of a subset U of the sphere, the matrix representation of g is computed by

$$g_{ij} = \langle \frac{\partial f}{\partial x_i}, \frac{\partial f}{\partial x_j} \rangle,$$

where $\langle , \rangle$ denotes the Euclidean scalar product on $\mathbb{R}^{n+1}$. We use this equality in Section 4 to compute the metric in stereographic and gnomonic coordinates.

Let M,N be manifolds of dimensions $\dim M = m$ and $\dim N = n$. Let $p \in M$ be some point in M. A map $f : M \to N$ is called *smooth at* p if there are charts $u : M \supset U \to \mathbb{R}^m$, $v : N \supset V \to \mathbb{R}^n$ with $p \in U$, $f(p) \in V$ such that the composition $\widetilde{f} = v \circ f \circ u^{-1} : \mathbb{R}^m \to \mathbb{R}^n$ is infinitely differentiable in the usual sense. We denote by $df : T_pM \to T_{f(p)}N$ the total differential of f at p. The map f is called *smooth* if it is smooth at all points of M.

A smooth bijective map $f : (M,g) \to (N,h)$ with smooth inverse is called a *diffeomorphism*. If f additionally preserves the metric,

$$g_p(X,Y) = h_{f(p)}(df(X), df(Y)),$$

we call f an *isometry*.

Given a metric g we can measure the length of a curve $\gamma : [a,b] \to M$ by integrating the norm of its tangent vector:

$$L(\gamma) = \int_a^b \|\dot{\gamma}(t)\| \, dt = \int_a^b \sqrt{g_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t))} dt.$$

The *geodesic distance* $d(p,q)$ of two points $p,q \in M$ is defined to be the infimum of lengths of all curves joining p with q. The minimizing curves are called *geodesics*. In the majority of cases, there

is no explicit formula for $d(p,q)$. Nevertheless, in the case of S^n with its canonical metric it is given by $d(p,q) = \arccos(\langle p,q\rangle)$. Here, the geodesic distance is realized by segments of great circles.

For each Riemannian metric g, there exists a corresponding *Riemannian volume form* ω given in local coordinates $u = (u_1,\ldots,u_m) : M \supset U \to \mathbb{R}^m$ by

$$\omega = \sqrt{\det(g)}du_1 \wedge \ldots \wedge du_n.$$

This can be viewed as a scaled version of the determinant that depends on the base point. Using a coordinate chart u, the volume of a subset A of M is given by

$$\mathrm{Vol}_g(A) = \int_A \omega = \int_{u(A)} \sqrt{\det(g)}du,$$

where the integration on the right hand side is performed in $\mathbb{R}^n$.

References

F.R. Bach. Active learning for misspecified generalized linear models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 65–72. MIT Press, Cambridge, MA, 2007.

M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 65–72, New York, NY, USA, 2006. ACM Press.

M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, 56(1–3):209–239, 2004.

D. J. Best and N. I. Fisher. Efficient simulation of the von Mises distribution. *Applied Statistics*, 28 (2):152–157, 1979.

L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

S. Fine, R. Gilad-Bachrach, and E. Shamir. Learning using query by committee, linear separation and random walks. *Theoretical Computer Science*, 284:25–51, 2002.

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.

S. Gallot, D. Hulin, and J. Lafontaine. *Riemannian Geometry*. Universitext. Springer, 1990.

M. Gromov. *Metric Structures for Riemannian and Non-Riemannian Spaces*. Progress in Mathematics. Birkhäuser, 1999.

R. Herbrich. *Learning Kernel Classifiers–Theory and Algorithms*. Adaptive Computation and Machine Learning. MIT Press, 2002.

G. Lebanon and J. Lafferty. Hyperplane margin classifiers on the multinomial manifold. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*. ACM Press, 2004.

C. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, 2000.

J. Milnor. The Schläfli differential inequality. In *Collected Papers (Volume 1)*. Publish or Perish, Houston, 1994.

H. Q. Minh, P. Niyogi, and Y. Yao. Mercer's theorem, feature maps, and smoothing. In *COLT*, pages 154–168, 2006.

T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

M. Opper, H. S. Seung, and H. Sompolinsky. Query by commitee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 287–294, Pittsburgh, Pennsylvania, United States, 1992.

G. W. Stewart. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*, 17:403–409, 1980.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

M. K. Warmuth, J. Liao, G. Rätsch, M. Mathieson, and C. Lemmen. Active learning in the drug discovery process. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1449–1456. MIT Press, 2002.

Journal of Machine Learning Research 9 (2008) 131-156 Submitted 10/05; Revised 7/07; Published 2/08

Evidence Contrary to the Statistical View of Boosting

David Mease MEASE_D@COB.SJSU.EDU
Department of Marketing and Decision Sciences
College of Business, San Jose State University
San Jose, CA 95192-0069, USA

Abraham Wyner AJW@WHARTON.UPENN.EDU
Department of Statistics
Wharton School, University of Pennsylvania
Philadelphia, PA, 19104-6340, USA

Editor: Yoav Freund

Abstract

The statistical perspective on boosting algorithms focuses on optimization, drawing parallels with maximum likelihood estimation for logistic regression. In this paper we present empirical evidence that raises questions about this view. Although the statistical perspective provides a theoretical framework within which it is possible to derive theorems and create new algorithms in general contexts, we show that there remain many unanswered important questions. Furthermore, we provide examples that reveal crucial flaws in the many practical suggestions and new methods that are derived from the statistical view. We perform carefully designed experiments using simple simulation models to illustrate some of these flaws and their practical consequences.

Keywords: boosting algorithms, LogitBoost, AdaBoost

1. Introduction

As the AdaBoost algorithm of Freund and Schapire (1996) gained popularity in the computer science community because of its surprising success with classification, the statistics community focused its efforts on understanding how and why the algorithm worked. Friedman, Hastie and Tibshirani in 2000 made great strides toward understanding the AdaBoost algorithm by establishing a statistical point of view. Among the many ideas in the Friedman, Hastie and Tibshirani *Annals of Statistics* paper, the authors identified a stagewise optimization in AdaBoost, and they related it to the maximization of the likelihood function in logistic regression. Much work has followed from this paper: extensions of the algorithm to the regression setting (e.g., Buhlmann and Yu, 2003), modification of the loss function (e.g., Hastie et al., 2001), and work on regularization methods for the original AdaBoost algorithm and variants (e.g., Lugosi and Vayatis, 2004). This broad statistical view of boosting is fairly mainstream in the statistics community. In fact, the statistics community has taken to attaching the *boosting* label to *any* classification or regression algorithm that incorporates a stagewise optimization.

Despite the enormous impact of the Friedman, Hastie and Tibshirani paper, there are still questions about the success of AdaBoost that are left unanswered by this statistical view of boosting. Chief among these is the apparent resistance to overfitting observed for the algorithm in countless examples from both simulated and real data sets. This disconnect was noted in some of the dis-

cussions published along with the original 2000 *Annals of Statistics* paper. For instance, Freund and Schapire (2000) note that, "one of the main properties of boosting that has made it interesting to statisticians and others is its relative (but not complete) immunity to overfitting," and write that the paper by Friedman, Hastie and Tibshirani "does not address this issue." Also Breiman (2000) writes, "a crucial property of AdaBoost is that it almost never overfits the data no matter how many iterations it is run," and states "unless I am missing something, there is no explanation in the paper."

Various arguments are given in response to the question of why boosting seems to not overfit. A view popular in computer science attributes the lack of overfitting to boosting's ability to achieve a large *margin* separating the two classes, as discussed by Schapire et al. (1998). A number of different opinions exist in the statistics community. Many statisticians simply argue that boosting does in fact overfit and construct examples to prove it (e.g., Ridgeway, 2000). While single examples certainly disprove claims that boosting *never* overfits, they do nothing to help us understand why boosting resists overfitting and performs very well for the large collection of examples that raised the question in the first place. Others argue that boosting will eventually overfit in most all cases if run for enough iterations, but that the number of iterations needed can be quite large since the overfitting is quite slow. Such a notion is difficult to disprove through real examples since any finite number of iterations may not be enough. Furthermore, it is difficult to prove limiting results for an infinite number of iterations without substantially over-simplifying the algorithm. Some evidence supporting the argument that boosting will eventually overfit can be found in Grove and Schuurmans (1998) which has examples for which boosting overfits when run for a very large number of iterations. Another argument often used is that boosting's success is judged with respect to 0/1 misclassification loss, which is a loss function that is not very sensitive to overfitting (e.g., Friedman et al., 2000b). More detailed explanations attribute the lack of overfitting to the stagewise nature of the algorithm (e.g., Buja, 2000). Along this same line, it has also been observed that the repeated iterations of the algorithm give rise to a self-averaging property (e.g., Breiman, 2000). This self-averaging works to reduce overfitting by reducing variance in ways similar to bagging (Breiman, 1996) and Random Forests (Breiman, 2001).

Whatever the explanation for boosting's resistance to overfitting in so many real and important examples, the statistical view of boosting as an optimization does little to account for this. In fact the statistical framework as proposed by Friedman, Hastie and Tibshirani does exactly the opposite; it suggests that *overfitting should be a major concern*. Still, in the final analysis, we do not imply that the statistical view is wrong. Indeed, we agree with Buja (2000) who writes, "There is no single *true* interpretation of anything; interpretation is a vehicle in the service of human comprehension. The value of an interpretation is in enabling others to fruitfully think about an idea." Certainly the paper of Friedman, Hastie and Tibshirani and other related work is quite valuable in this regard. However, any view or theoretical understanding generally gives rise to practical suggestions for implementation. Due to the disconnect between the statistical view and reality, many of these resulting practical suggestions are misguided and empirical performance suffers accordingly. In this paper we focus on illustrating this phenomenon through simulation experiments.

It is important to note that although this paper deals with "the statistical view of boosting", it is an overgeneralization to imply there is only one single view of boosting in the statistical community. All statisticians are not of a single mindset, and much literature has been produced subsequent to the Friedman, Hastie and Tibshirani *Annals of Statistics* paper. Much of what we categorize as the statistical view of boosting can be found in that original paper, but other ideas, especially those in Sections 3.9, 3.10, 4.9 and 4.10, are attributable to other researchers and subsequent publications in

the statistics community. For this reason, we are careful to provide references and direct quotations throughout this paper.

The following section describes the general setting for classification and the AdaBoost algorithm. Sections 3 and 4 consider a collection of practical suggestions, commonly held beliefs and modifications to the AdaBoost algorithm based on the statistical view. For each one, a simulation providing contradictory evidence is included. Section 5 mentions a slightly different set of simulations to consider, and finally Section 6 offers practical advice in light of the evidence presented in this paper as well as some concluding remarks.

2. The Classification Problem and Boosting

In this section we will begin by describing the general problem of classification in statistics and machine learning. Next we will describe the AdaBoost algorithm and give details of our implementation.

2.1 Classification

The problem of classification is an instance of what is known as *supervised learning* in machine learning. We are given training data $x_1,...,x_n$ and $y_1,...,y_n$ where each x_i is a $d-$dimensional vector of predictors $(x_i^{(1)},...,x_i^{(d)})$ and $y_i \in \{-1,+1\}$ is the associated observed class label. To justify generalization, it is usually assumed that the training data are *iid* samples of random variables (X,Y) having some unknown distribution. The goal is to learn a rule $\hat{C}(x)$ that assigns a class label in $\{-1,+1\}$ to any new observation x. The performance of this rule is usually measured with respect to misclassification error, or the rate at which new observations drawn from the same population are incorrectly labelled. Formally we can define the misclassification error for a classification rule $\hat{C}(x)$ as $\mathrm{P}(\hat{C}(X) \neq Y)$.

For any given data set misclassification error can be estimated by reserving a fraction of the available data for test data and then computing the percent of incorrect classifications resulting from the classifier trained on the remainder of the data. Various cross-validation techniques improve upon this scheme by averaging over different sets of test data. In this paper we will consider only examples of simulated data so that the joint distribution of X and Y is known. This will enable us to estimate misclassification error as accurately as desired by simply repeatedly simulating training and test data sets and averaging the misclassification errors from the test sets.

2.2 Boosting

AdaBoost (Freund and Schapire, 1996) is one of the first and the most popular boosting algorithms for classification. The algorithm is as follows. First let $F_0(x_i) = 0$ for all x_i and initialize weights $w_i = 1/n$ for $i = 1,...,n$. Then repeat the following for m from 1 to M:

- Fit the classifier g_m to the training data using weights w_i where g_m maps each x_i to -1 or 1.
- Compute the weighted error rate $\varepsilon_m \equiv \sum_{i=1}^{n} w_i \mathrm{I}[y_i \neq g_m(x_i)]$ and half its log-odds, $\alpha_m \equiv \frac{1}{2}\log\frac{1-\varepsilon_m}{\varepsilon_m}$.
- Let $F_m = F_{m-1} + \alpha_m g_m$.
- Replace the weights w_i with $w_i \equiv w_i e^{-\alpha_m g_m(x_i) y_i}$ and then renormalize by replacing each w_i by $w_i/(\sum w_i)$.

The final classifier is 1 if $F_M > 0$ and -1 otherwise. The popularity of this algorithm is due to a vast amount of empirical evidence demonstrating that the algorithm yields very small misclassification error relative to competing methods. Further, the performance is remarkably insensitive to the choice of the total number of iterations M. Usually any sufficiently large value of M works well. For the simulations in this paper we will take $M = 1000$, with the single exception of the simulation in Section 4.7 where it is instructive to consider $M = 5000$.

Many variations of the AdaBoost algorithm now exist. We will visit some of these in Sections 3 and 4 and compare their performance to the original AdaBoost algorithm. Further, these variations as well as AdaBoost itself are very flexible in the sense that the class of classifiers from which each g_m is selected can be quite general. However, the superior performance of AdaBoost is generally in the context of classification trees. For this reason we will use classification trees in our experiments. Specifically, the trees will be fit using the "rpart" function in the "R" statistical software package (http://www.r-project.org/). The R code for all the experiments run in this paper is available on the web page http://www.davemease.com/contraryevidence.

3. Experiments Which Contradict the Statistical View of Boosting

In this section we describe the results of several experiments based on simulations from the model introduced below. Each experiment is meant to illustrate particular inconsistencies between that which is suggested by the statistical view of boosting and what is actually observed in practice.

For the experiments we will consider in this section we will simulate data from the model

$$\mathrm{P}(Y = 1|x) = q + (1-2q)\ \mathrm{I}\left[\sum_{j=1}^{J} x^{(j)} > J/2\right].$$

We will take X to be distributed *iid* uniform on the d-dimensional unit cube $[0,1]^d$. The constants n, d, J and q will be set at different values depending on the experiment. Note that q is the Bayes error and $J \leq d$ is the number of effective dimensions. Recall n is the number of observations used to train the classifier. The unconditional probabilities for each of the two classes are always equal since $\mathrm{P}(Y = 1) = \mathrm{P}(Y = 0) = 1/2$. The only exceptions to this are experiments in which we take $J = 0$ for which the sum (and thus the indicator) is taken to be always zero. In these cases the model reduces to the "pure noise" model $\mathrm{P}(Y = 1|x) \equiv q$ for all x.

3.1 Should Stumps Be Used for Additive Bayes Decision Rules?

Additive models are very popular in many situations. Consider the case in which the Bayes decision rule is additive in the space of the predictors $x^{(1)},...,x^{(d)}$. By this we mean that the Bayes decision rule can be written as the sign of $\sum_{i=1}^{d} h_i(x^{(i)})$ for some functions $h_1,...,h_d$. This is true, for example, for our simulation model. The classification rule produced by AdaBoost is itself necessarily additive in the classifiers g_m. Thus when the g_m are functions of only single predictors the AdaBoost classification rule is additive in the predictor space. For this reason it has been suggested that one should use stumps (2-node trees) if one believes the optimal Bayes rule is approximately additive, since stumps are trees which only involve single predictors and thus yield an additive model in the predictor space for AdaBoost. It is believed that using trees of a larger size will lead to overfitting because it introduces higher-level interactions. This argument is made explicit in Hastie et al. (2001) on pages 323-324 and in Friedman et al. (2000a) on pages 360-361.

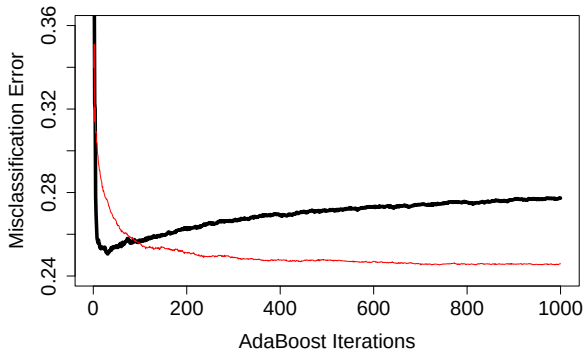


Figure 1: Comparison of AdaBoost with Stumps (Black, Thick) and 8-Node Trees (Red, Thin) for an Additive Bayes Rule

Despite the logic of this argument which is based on the idea that one should use an additive model when fitting an additive function, it can be observed that often, in fact, using larger trees is more effective than using stumps even when the Bayes rule is additive. The reason has to do with the fact that boosting with larger trees actually often overfits *less* than boosting with smaller trees in practice since the larger trees are more orthogonal and a self-averaging process prevents overfitting. We do not endeavor to make this argument rigorous here, but we will provide a compelling example.

For our example we will use our model with a Bayes error rate of $q = 0.1$, a training sample size of $n = 200$ and $d = 20$ dimensions of which $J = 5$ are active. Figure 1 displays the misclassification error of AdaBoost based on hold out samples of size 1000 (also drawn *iid* on $[0,1]^d$) as a function of the iterations. The results are averaged over 100 repetitions of the simulation. While AdaBoost with stumps (thick, black curve) leads to overfitting very early on, AdaBoost with 8-node trees (thin, red curve) does not suffer from overfitting and leads to smaller misclassification error. In fact, the misclassification error by 1000 iterations was smaller for the 8-node trees in 96 of the 100 simulations. The average (paired) difference in misclassification error was 0.031 with a standard error of $0.018/\sqrt{100} = 0.0018$. Also note that both algorithms here perform considerably worse than the Bayes error rate of $q = 0.1$.

The R code for this experiment as well as all others in this paper can be found at http://www.davemease.com/contraryevidence. We encourage the reader to appreciate the reproducibility of the qualitative result by running the code for various values of the parameters q, n, d and J.

It is worth further commenting on the fact that in this simulation AdaBoost with stumps leads to overfitting while AdaBoost with the larger 8-node trees does not, at least by 1000 iterations. This is of special interest since many of the examples other researchers provide to show AdaBoost can in fact overfit often use very small trees such as stumps as the base learner. Some such examples of overfitting can be found in Friedman et al. (2000a), Jiang (2000) and Ridgeway (2000) as well as Leo Breiman's 2002 Wald Lectures on Machine Learning.[1] The belief is that if stumps overfit then so will larger trees since the larger trees are more complex. (Clearly the example presented

1. Breiman's lecture notes can be found at http://www.stat.berkeley.edu/users/breiman/wald2002-1.pdf.

in this section shows that this is not the case.) To illustrate this viewpoint consider the quote from Jiang (2001) who writes, "all these base systems, even the ones as simple as the 'stumps', will unavoidably lead to suboptimal predictions when boosted forever." Additionally, such examples in which overfitting is observed also often deal with extremely low-dimensional cases such as $d = 2$ or even $d = 1$. By experimenting with the simulation code provided along with this paper one can confirm that in general AdaBoost is much more likely to suffer from overfitting in trivial low-dimensional examples as opposed to high-dimensional situations where it is more often used.

3.2 Should Smaller Trees Be Used When the Bayes Error is Larger?

Similar arguments to those in the previous section suggest that it is necessary to use smaller trees for AdaBoost when the Bayes error is larger. The reasoning is that when the Bayes error is larger, the larger trees lead to a more complex model which is more susceptible to overfitting noise. However, in practice we can often observe the opposite to be true. The higher Bayes error rate actually can favor the larger trees. This counterintuitive result may be explained by the self-averaging which occurs during the boosting iterations as discussed by Krieger et al. (2001). Conversely, the smaller trees often work well for lower Bayes error rates, provided they are rich enough to capture the complexity of the signal.

We illustrate this phenomenon by re-running the experiment in the previous section, this time using $q = 0$, which implies the Bayes error is zero. The average misclassification error over the 100 hold out samples is displayed in the top panel of Figure 2. It can now be observed that AdaBoost with stumps performs better than AdaBoost with 8-node trees. In fact, this was the case in 81 out of the 100 simulations (as opposed to only 4 of the 100 for $q = 0.1$ from before). The mean difference in misclassification error after 1000 iterations was 0.009 with a standard error of $0.011/\sqrt{100} = 0.0011$. The bottom panel of Figure 2 confirms that AdaBoost with stumps outperforms AdaBoost with 8-node tress only for very small values of q with this simulation model.

3.3 Should LogitBoost Be Used Instead of AdaBoost for Noisy Data?

The LogitBoost algorithm was introduced by Friedman et al. (2000a). The algorithm is similar to AdaBoost, with the main difference being that LogitBoost performs stagewise minimization of the negative binomial log likelihood while AdaBoost performs stagewise minimization of the exponential loss. By virtue of using the binomial log likelihood instead of the exponential loss, the LogitBoost algorithm was believed to be more "gentle" and consequently likely to perform better than AdaBoost for classification problems in which the Bayes error is substantially larger than zero. For instance, on page 309 Hastie et al. (2001) write, "it is therefore far more robust in noisy settings where the Bayes error rate is not close to zero, and especially in situations where there is misspecification of the class labels in the training data."

Despite such claims, we often observe the opposite behavior. That is, when the Bayes error is not zero, LogitBoost often overfits while AdaBoost does not. As an example, we consider the performance of AdaBoost and LogitBoost on the simulation from Section 3.1 in which the Bayes error was $q = 0.1$. The base learners used are 8-node trees. Figure 3 displays the performance averaged over 100 hold out samples. It is clear that LogitBoost (blue, thick) begins to overfit after about 200 iterations while AdaBoost (red, thin) continues to improve. After 1000 iterations the mean difference was 0.031 with a standard error of $0.017/\sqrt{100}$=0.0017. The misclassification error for LogitBoost at 1000 iterations was larger than that of AdaBoost in all but 4 of the 100 simulations.

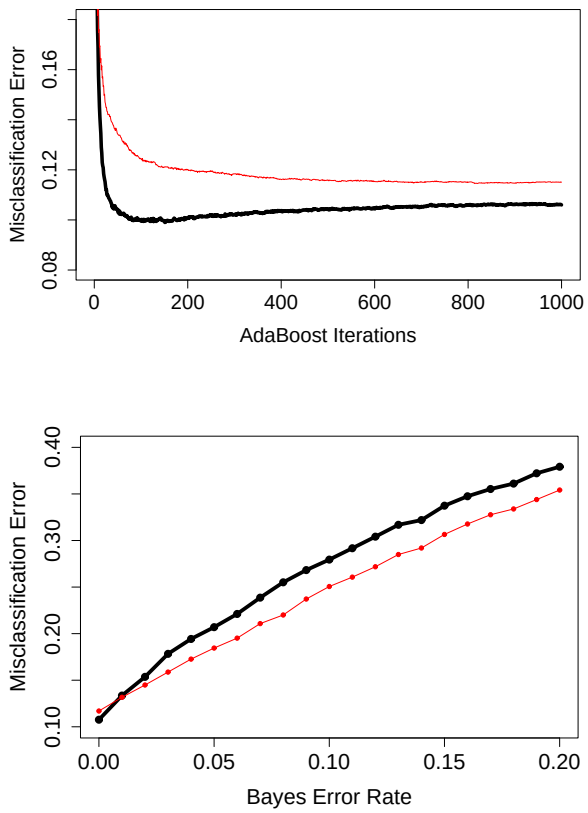


Figure 2: Comparison of AdaBoost with Stumps (Black, Thick) and 8-Node Trees (Red, Thin) for an Additive Bayes Rule. Top Panel: Misclassification Error for Zero Bayes Error as a Function of the Iterations. Bottom Panel: Misclassification Error at 1000 Iterations as a Function of the Bayes Error Rate q.

Other examples of this phenomenon of LogitBoost overfitting noisy data when AdaBoost does not can be found in Mease et al. (2007).

The R code used for LogitBoost was written by Marcel Dettling and Peter Buhlmann and can be found at http://stat.ethz.ch/∼dettling/boosting.html. Two small modifications were made to the code in order to fit 8-node trees, as the original code was written for stumps.

It should be noted that LogitBoost differs from AdaBoost not only in the loss function which it minimizes, but also in the Newton style minimization that it employs to carry out the minimization. For this reason it would be of interest to examine the performance of the algorithm in Collins et al. (2000) which minimizes the negative binomial log likelihood in a manner more analogous to AdaBoost. We do not consider that algorithm in this paper since our focus is mainly on the work of Friedman et al. (2000a) and the implications in the statistical community.

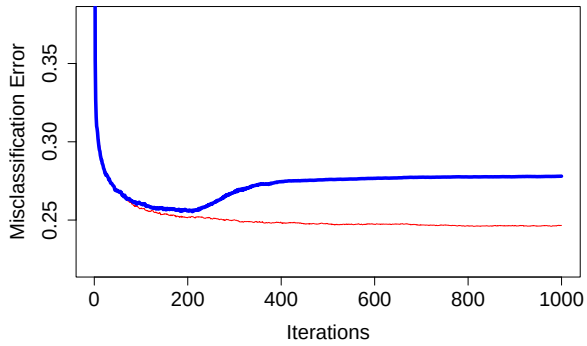


Figure 3: Comparison of AdaBoost (Red, Thin) and LogitBoost (Blue, Thick) with 8-Node Trees

3.4 Should Early Stopping Be Used to Prevent Overfitting?

In order to prevent overfitting, one popular regularization technique is to stop boosting algorithms after a very small number of iterations, such as 10 or 100. The statistics community has put a lot of emphasis on early stopping as evidenced by the large number of papers on this topic. For example, the paper "Boosting with Early Stopping: Convergence and Consistency" by Zhang and Yu (2005) tells readers that "boosting forever can overfit the data" and that "therefore in order to achieve consistency, it is necessary to stop the boosting procedure early." Standard implementations of boosting such as the popular gbm package for R by Ridgeway (2005) implement data-derived early stopping rules.

The reasoning behind early stopping is that after enough iterations have occurred so that the complexity of the algorithm is equal to the complexity of the underlying true signal, then any additional iterations will lead to overfitting and consequently larger misclassification error. However, in practice we can often observe that additional iterations beyond the number necessary to match the complexity of the underlying true signal actually reduce the overfitting that has already occurred rather than causing additional overfitting. This is likely due to the self-averaging property of AdaBoost to which we eluded earlier.

To illustrate this we use a somewhat absurd example. We take $J = 0$ in our simulation model, so that there is no signal at all, only noise. We have $\mathrm{P}(Y = 1|x) \equiv q$ so that Y does not depend on x in any way. We take a larger sample size of $n = 5000$ this time, and also use larger $2^8 = 256$-node trees. The experiment is again averaged over 100 repetitions, each time drawing the $n = 5000$ x values from $[0,1]^d$ with $d = 20$. The 100 hold out samples are also drawn from $[0,1]^{20}$ each time. The Bayes error rate is $q = 0.2$.

Since there is no signal to be learned, we can observe directly the effect of AdaBoost's iterations on the noise. We see in Figure 4 that early on there is some overfitting, but this quickly goes away and the misclassification error decreases and appears to asymptote very near the Bayes error rate of $q = 0.2$. In fact, the final average after 1000 iterations (to three decimals accuracy) is 0.200 with a standard error of $0.013/\sqrt{100}$=0.0013. Even more interesting, the misclassification error after 1000 iterations is actually less than that after the first iteration (i.e., the misclassification error for a single 2^8-node tree). The mean difference between the misclassification error after one iteration and that after 1000 iterations was 0.012 with a standard error of $0.005/\sqrt{100}$=0.0005. The difference was

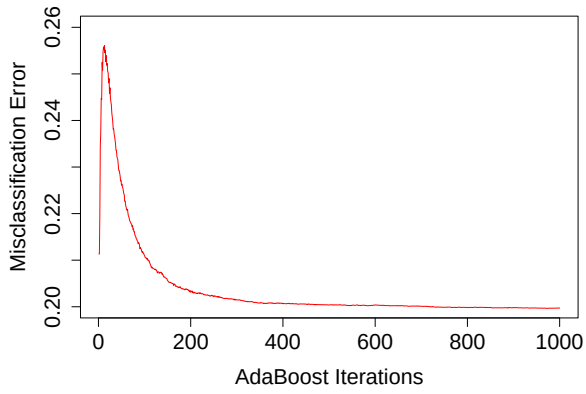


Figure 4: AdaBoost on 20% Pure Noise

positive in 99 of the 100 repetitions. Thus we see that not only does AdaBoost resist overfitting the noise, it actually fits a classification rule that is less overfit than its own 2^8-node tree base classifier.

3.5 Should Regularization Be Based on the Loss Function?

Since the statistical view of boosting centers on the stagewise minimization of a certain loss function on the training data, a common suggestion is that regularization should be based on the behavior of that loss function on a hold out or cross-validation sample. For example, the implementation of the AdaBoost algorithm in the gbm package (Ridgeway, 2005) uses the exponential loss $\sum_{i=1}^{n} e^{-y_i F_m(x_i)}$ to estimate the optimal stopping time. Indeed, if early stopping is to be used as regularization, the statistical view would suggest stopping when this exponential loss function begins to increase on a hold out sample. However, in practice the misclassification error often has little to do with the behavior of the exponential loss on a hold out sample. To illustrate this, we return to the experiment in Section 3.1. If we examine the exponential loss on hold out samples for AdaBoost with the 8-node trees, it can be seen that this loss function is exponentially increasing throughout the 1000 iterations. This is illustrated in Figure 5 which shows the linear behavior of the log of the exponential loss for a single repetition from this experiment on a hold out sample of size 1000. Thus, early stopping regularization based on the loss function would suggest stopping after just one iteration, when in fact Figure 1 shows we do best to run the 8-node trees for the full 1000 iterations. This behavior has also been noted for LogitBoost as well (with respect to the negative log likelihood loss) in Mease et al. (2007) and in Dettling and Buhlmann (2003). In the latter reference the authors estimated a stopping parameter for the number of iterations using cross-validation but observed that they "could not exploit significant advantages of estimated stopping parameters" over allowing the algorithm to run for the full number of iterations (100 in their case).

3.6 Should the Collection of Basis Functions Be Restricted to Prevent Overfitting?

Another popular misconception about boosting is that one needs to restrict the class of trees in order to prevent overfitting. The idea is that if AdaBoost is allowed to use *all* 8-node trees for instance, then the function class becomes too rich giving the algorithm too much flexibility which

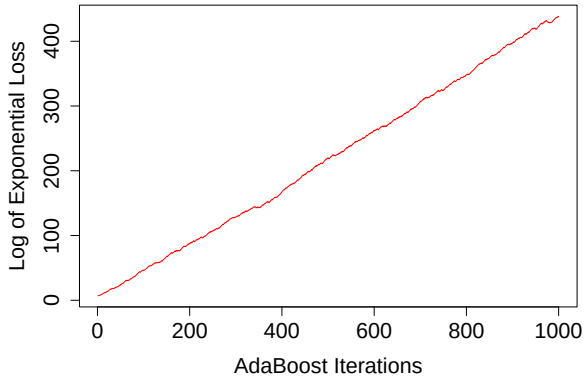


Figure 5: The Log of the Exponential Loss for AdaBoost on a Hold Out Sample

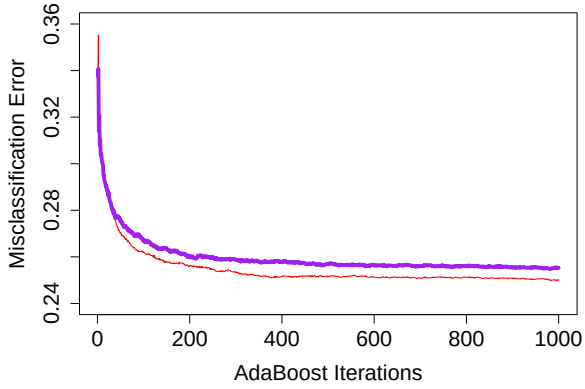


Figure 6: Comparison of AdaBoost with 8-Node Trees (Red, Thin) to AdaBoost with 8-Node Trees Restricted to Have at Least 15 Observations in Each Terminal Node (Purple, Thick)

leads to overfitting. This line of thinking gives rise to various methods for restricting or regularizing the individual trees themselves as a method of regularizing the AdaBoost algorithm. For instance, the implementation of AdaBoost in the gbm code (Ridgeway, 2005) has a parameter called "n.minobsinnode" which is literally the minimum number of observations in the terminal nodes of the trees. The default of this value is not 1, but 10.

In spite of this belief, it can be observed that the practice of limiting the number of observations in the terminal nodes will often degrade the performance of AdaBoost. It is unclear why this happens; however, we note that related tree ensemble algorithms such as PERT (Cutler and Zhao, 2001) have demonstrated success by growing the trees until only a single observation remains in each terminal node.

As an example of this performance degradation, we again revisit the simulation in Section 3.1 and compare the (unrestricted) 8-node trees used there to 8-node trees restricted to have at least 15

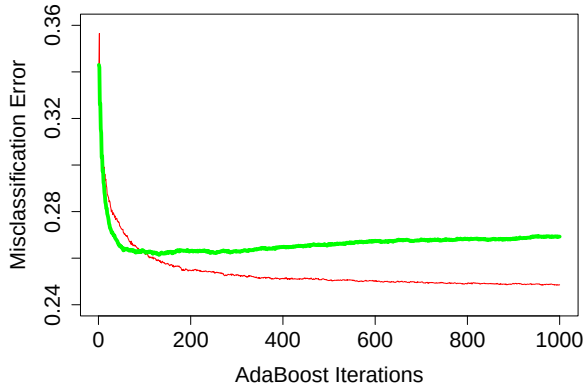


Figure 7: Comparison of AdaBoost (Red, Thin) and AdaBoost with Shrinkage (Green, Thick)

observations in each terminal node. (This is done in R by using the option "minbucket=15" in the "rpart.control" syntax.) Figure 6 shows the results with the unrestricted 8-node trees given by the red (thin) curve and the restricted 8-node trees given by the purple (thick) curve. The degradation in performance is evident, although not extremely large. The mean difference in misclassification error at 1000 iterations was 0.005 with a standard error of $0.010/\sqrt{100}$=0.001. AdaBoost with unrestricted 8-node trees gave a lower misclassification error in 67 of the 100 repetitions.

3.7 Should Shrinkage Be Used to Prevent Overfitting?

Shrinkage is yet another form of regularization that is often used for boosting algorithms. In the context of AdaBoost, shrinkage corresponds to replacing the α_m in the update formula $F_m = F_{m-1} + \alpha_m g_m$ by $\nu\alpha_m$ where ν is any positive constant less than one. The value $\nu = 0.1$ is popular. In the statistical view of boosting, shrinkage is thought to be extremely important. It is believed to not only reduce overfitting but also to increase the maximum accuracy (i.e., the minimum misclassification error) over the iterations. For instance, Friedman et al. (2000b) write, "the evidence so far indicates that the smaller the value of ν, the higher the overall accuracy, as long as there are enough iterations."

Despite such claims, it can be observed that shrinkage often does not improve performance and instead can actually cause AdaBoost to overfit in situations where it otherwise would not. To understand why this happens one needs to appreciate that it is the suboptimal nature of the stagewise fitting of AdaBoost that helps it to resist overfitting. Using shrinkage can destroy this resistance. For an example, we again revisit the simulation in Section 3.1 using the 8-node trees. In Figure 7 the red (thin) curve corresponds to the misclassification error for the 8-node trees just as in Section 3.1 and the green (thick) curve now shows the effect of using a shrinkage value of $\nu = 0.1$. It is clear that the shrinkage causes overfitting in this simulation. By 1000 iterations shrinkage gave a larger misclassification error in 95 of the 100 repetitions. The mean difference in misclassification error at 1000 iterations was 0.021 with a standard error of $0.012/\sqrt{100}$=0.0012.

3.8 Is Boosting Estimating Probabilities?

The idea that boosting produces probability estimates follows directly from the statistical view through the stagewise minimization of the loss function. Specifically, the exponential loss $\sum_{i=1}^{n} e^{-y_i F_m(x_i)}$, which is minimized at each stage by AdaBoost, achieves its minimum when the function $F_m(x)$ relates to the true conditional class probabilities $p(x) \equiv \mathrm{P}(Y=1|x)$ by the formula $F_m(x) = \frac{1}{2}\log\frac{p(x)}{1-p(x)}$. This leads to the estimator of $p(x)$ after m iterations given by

$$\hat{p}_m(x) = 1/(1+e^{-2F_m(x)}).$$

This relationship between the score function F_m in AdaBoost and conditional class probabilities is given explicitly in Friedman et al. (2000a). An analogous formula is also given for obtaining probability estimates from LogitBoost. Standard implementations of boosting such as Dettling and Buhlmann's LogitBoost code at http://stat.ethz.ch/∼dettling/boosting.html as well as the gbm LogitBoost code by Ridgeway (2005) output conditional class probabilities estimates directly.

Despite the belief that boosting is estimating probabilities, the estimator $\hat{p}_m(x)$ given above is often extremely overfit in many cases in which the classification rule from AdaBoost shows no signs of overfitting and performs quite well. An example is given by the experiment in Section 3.1. In Figure 1 we saw that the classification rule using 8-node trees performed well and did not overfit even by 1000 iterations. Conversely, the probability estimates are severely overfit early on. This is evidenced by the plot of the exponential loss in Figure 5. In this context the exponential loss can be thought of as an estimate of a *probability scoring rule* which quantifies the average disagreement between a true probability p and an estimate $\hat{p}$ using only binary data (Buja et al., 2006). For the exponential loss the scoring rule is $p\sqrt{(1-\hat{p})/\hat{p}} + (1-p)\sqrt{\hat{p}/(1-\hat{p})}$. The fact that the plot in Figure 5 is increasing shows that the probabilities become worse with each iteration as judged by this scoring rule. Similar behavior can be seen using other scoring rules such as the squared loss $(p-\hat{p})^2$ and the log loss $-p\log\hat{p} - (1-p)\log(1-\hat{p})$ as shown in Mease et al. (2007). This reference also shows the same behavior for the probability estimates from LogitBoost, despite the fact that efficient probability estimation is the main motivation for the LogitBoost algorithm.

The reason for the overfitting of these probability estimators is that as more and more iterations are added to achieve a good classification rule, the value of $|F_m|$ at any point is increasing quickly. The classification rule only depends on the sign of F_m and thus is not affected by this. However, this increasing tendency of $|F_m|$ impacts the probability estimates by causing them to quickly diverge to 0 and 1. Figure 8 shows the probability estimates $\hat{p}_m(x_i) = 1/(1+e^{-2F_m(x_i)})$ for AdaBoost from a single repetition of the experiment in Section 3.1 using 8-node trees on a hold out sample of size 1000. The top histogram corresponds to $m = 10$ iterations and the bottom histogram shows $m = 1000$ iterations. The histograms each have 100 equal width bins. It can be seen that after only 10 iterations almost all of the probability estimates are greater than 0.99 or less than 0.01, and even more so by 1000 iterations. This indicates a poor fit since we know all of the true probabilities are either 0.1 or 0.9.

Other researchers have also noted this type of overfitting with boosting and have used it as an argument in favor of regularization techniques. For instance, it is possible that using a regularization technique such as shrinkage or the restriction to stumps as the base learners in this situation could produce better probability estimates. However, from what we have seen of some regularization techniques in this paper, we know that regularization techniques often severely degenerate the classification performance of the algorithm. Furthermore, some are not effective at all in many

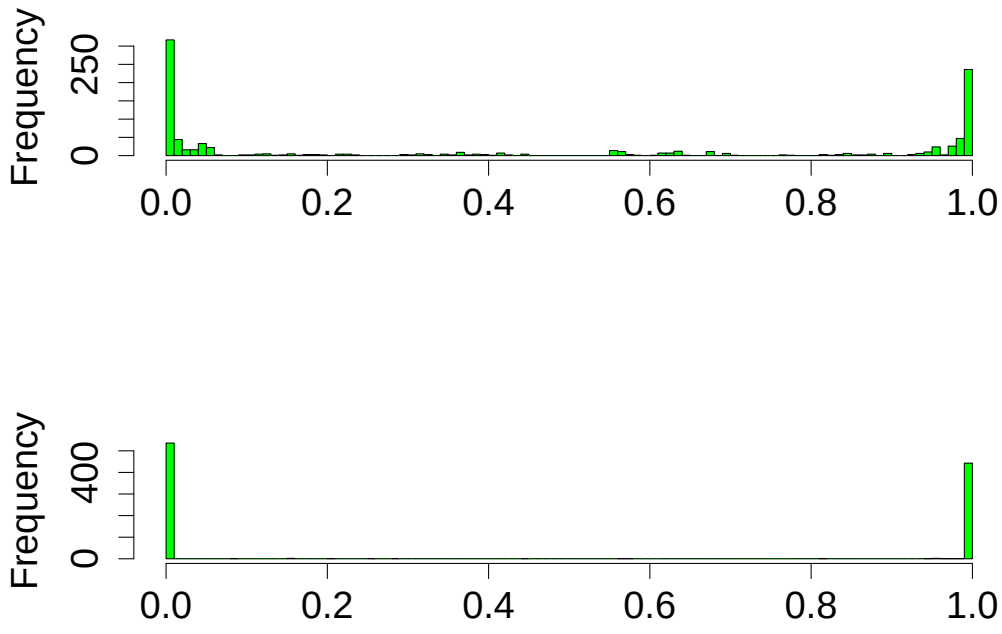


Figure 8: Probability Estimates From AdaBoost at $m = 10$ Iterations (Top) and $m = 1000$ Iterations (Bottom)

situations. For instance, early stopping, one of the most popular regularization techniques, is of little help when the probabilities overfit from the outset as in Figure 5. For a technique that achieves conditional probability estimation using AdaBoost without modification or regularization the reader should see Mease et al. (2007).

3.9 Is Boosting Similar to the One Nearest Neighbor Classifier?

In all the experiments considered in this paper, AdaBoost achieves zero misclassification error on the training data. This characteristic is quite typical of AdaBoost and has led some researchers to draw parallels to the (one) nearest neighbor classifier, a classifier which necessarily also yields zero misclassification error on the training data. This characteristic has also been suggested as a reason why AdaBoost will overfit when the Bayes error is not zero.

The belief in a similarity between boosting and the nearest neighbor classifier was not expressed in the original paper of Friedman et al. (2000a), but rather has been expressed more recently in the statistics literature on boosting by authors such as Wenxin Jiang in papers such as Jiang (2000), Jiang (2001) and Jiang (2002). In Jiang (2000), the equivalence between AdaBoost and the nearest neighbor classifier is established only for the case of $d = 1$ dimension. In the $d = 1$ case, the equivalence is merely a consequence of fitting the training data perfectly (and following Jiang's

convention of using midpoints of the training data for the classification tree splits). However, as we will see from the experiment in this section, the behavior of AdaBoost even in $d = 2$ dimensions is radically different from the nearest neighbor rule.

Despite this difference, Jiang goes on to suggest that the performance of AdaBoost in higher dimensions might be similar to the case of $d = 1$ dimension. For instance in "Is Regularization Unnecessary for Boosting?" Jiang (2001) writes, "it is, however, plausible to conjecture that even in the case of higher dimensional data running AdaBoost forever can still lead to a suboptimal prediction which does not perform much better than the nearest neighbor rule." Further, Jiang (2002) writes, "the fit will be perfect for almost all sample realizations and agree with the nearest neighbor rule at all the data points as well as in some of their neighborhoods" and that "the limiting prediction presumably cannot perform much better than the nearest neighbor rule."

To understand why equivalent behavior on the training data (or "data points" using Jiang's terminology above) does not imply similar performance for classification rules for $d > 1$, it is important to remember that in the case of continuous data the training data has measure zero. Thus the behavior on the training data says very little about the performance with respect to the population. This is well illustrated by the pure noise example from Section 3.4. For any point in the training data for which the observed class differs from the class given by the Bayes rule, both AdaBoost and nearest neighbor will classify this point as the observed class and thus disagree with the Bayes rule. However, the volume of the affected neighborhood surrounding that point can be arbitrarily small with AdaBoost, but will necessarily be close to $1/n$ of the total volume with nearest neighbor.

To help the reader visualize this, we consider a $d = 2$-dimensional version of the pure noise example from Section 3.4. We again use a Bayes error rate of $q = 0.2$ but now take only $n = 200$ points spread out evenly according to a Latin hypercube design. The left plot in Figure 9 shows the resulting classification of AdaBoost using 8-node trees after 1000 iterations and the right plot shows the rule for nearest neighbor. The training points with $Y = -1$ are colored black and those with $Y = +1$ are colored yellow. Regions classified as -1 are colored purple and those classified as $+1$ are colored light blue. Since the Bayes rule is to classify the entire area as -1, we can measure the overfitting of the rules by the fraction of the total area colored as light blue. The nearest neighbor classifier has 20% of the region colored as light blue (as expected), while AdaBoost has only 16%. The two classifiers agree "at all the [training] data points as well as in some of their neighborhoods" as stated by Jiang, but the "some" here is relatively small.

In higher dimensions this effect is even more pronounced. For the $d = 20$-dimensional example from Section 3.4 the area (volume) of the light blue region would be essentially zero for AdaBoost (as evidenced by its misclassification error rate matching almost exactly that of the Bayes error), while for nearest neighbor it remains at 20% as expected. Thus we see that the nearest neighbor classifier differs from the Bayes rule for 20% of the points in both the training data and the population while AdaBoost differs from the Bayes rule for 20% of the points in the training data but virtually none in the population.

The differences between the nearest neighbor classifier and AdaBoost are obvious in the other experiments in this paper as well. For instance, for the experiment in Section 3.1 the nearest neighbor classifier had an average misclassification error rate of 0.376 versus 0.246 for AdaBoost with the 8-node trees.

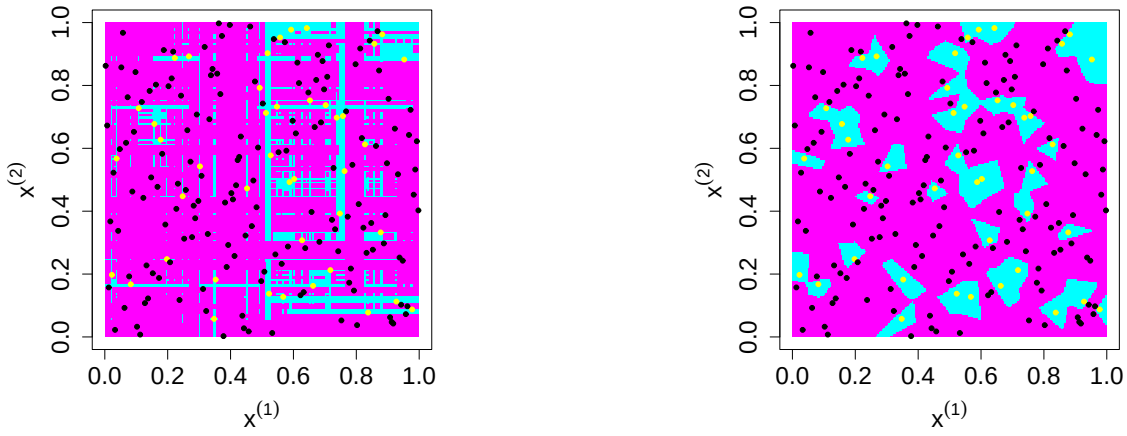


Figure 9: Comparison of AdaBoost (Left) and Nearest Neighbor (Right) on 20% Pure Noise

3.10 Is Boosting Consistent?

An important question to ask about any estimator is whether or not it is consistent. A consistent estimator is defined to be any estimator for which the estimated quantity converges in probability to the true quantity. In our context, to ask if AdaBoost is a consistent estimator is to ask if its classification rule converges in probability to the Bayes rule. If it is consistent, then with a sufficiently large training sample size n its misclassification error will come arbitrarily close to the Bayes error.

The belief in the statistics community is that AdaBoost is not consistent unless regularization is employed. The main argument given is that if AdaBoost is left unregularized it will eventually fit all the data thus making consistency impossible as with the nearest neighbor classifier. Consequently, all work on the consistency of boosting deals with regularized techniques. While we have noted in Section 3.9 that it is characteristic of AdaBoost to achieve zero misclassification error on the training data, we have also discussed the fact that this in no way determines its performance in general, as the training data has measure zero in the case of continuous data. In fact in Section 3.4 we observed that with a sample size of $n = 5000$ AdaBoost with 2^8-node trees achieved the Bayes error rate to three decimals on a 20% pure noise example despite fitting all the training data.

In this section we consider a simulation with this same sample size and again 2^8-node trees but we now include a signal in addition to the noise. We take $J = 1$ and use $d = 5$ dimensions and fix the Bayes error rate at $q = 0.1$. The resulting misclassification error rate averaged over 100 repetitions each with a hold out sample of size 1000 is shown in Figure 10. As before, AdaBoost fits all the training data early on, but the misclassification error after 1000 iterations averages only 0.105 with a standard error of $0.010/\sqrt{100}$=0.001. This is quite close to the Bayes error rate $q = 0.1$ and can be observed to come even closer by increasing the sample size. It should also be noted that this error rate is much below the limit of $2q(1-q) = 0.18$ that holds for the nearest neighbor classifier in this case.

The belief that unregularized AdaBoost can not be consistent is promoted by Wenxin Jiang's work mentioned in Section 3.9 connecting the performance of AdaBoost and the nearest neighbor classifier. His result for $d = 1$ rules out consistency in that case since the nearest neighbor rule is

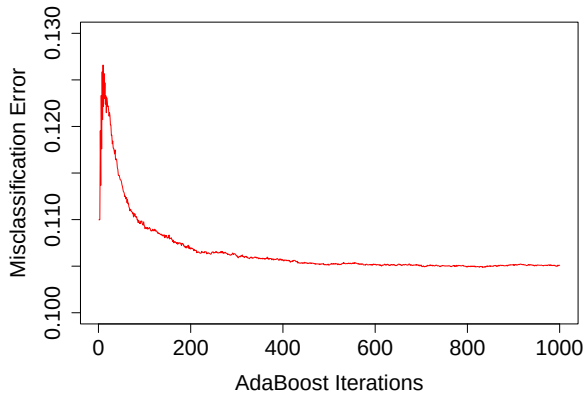


Figure 10: Performance of AdaBoost for a Simulation with a Bayes Error of 0.1

not consistent, but nothing is established for $d > 1$ with regard to AdaBoost. Jiang (2002) admits this when he writes, "what about boosting forever with a higher dimensional random continuous predictor x with $\dim(x) > 1$? We do not have theoretical results on this so far."

4. More Experiments Which Contradict the Statistical View of Boosting

In this section we revisit the experiments from Section 3 using a different simulation model. The purpose here is to show that the results are reproducible and do not depend on a particular simulation model. We also encourage readers to experiment with other simulation models by modifying the code provided on the web page.

The simulations in this section will use the model

$$P(Y=1|x) = \begin{cases} q & x^{(1)} \in [0,0.1) \cup [0.2,0.3) \cup [0.4,0.5) \cup [0.6,0.7) \cup [0.8,0.9) \\ 1-q & x^{(1)} \in [0.1,0.2) \cup [0.3,0.4) \cup [0.5,0.6) \cup [0.7,0.8) \cup [0.9,1]. \end{cases}$$

We will rerun each experiment from Section 3 using this model. Throughout this section we will use $d = 20$ dimensions and take X to be distributed *iid* uniform on the 20-dimensional unit cube $[0,1]^{20}$. For each experiment we will use twice the sample size of the analogous experiment in Section 3 and the same Bayes error q. The single exception will be the experiment in Section 4.9 in which we use a Bayes error of $q = 0.1$ and $d = 2$ dimensions for visualization purposes.

Note that while the experiments in Section 3 had $J \leq d$ effective dimensions, the experiments in this section will all have only one effective dimension as a result of this simulation model. The plots in Figure 19 are useful for visualizing this model in $d = 2$ dimensions.

4.1 Should Stumps Be Used for Additive Bayes Decision Rules?

As in Section 3.1 we use a Bayes error rate of $q = 0.1$ and $d = 20$ dimensions. We use the new simulation model with a training sample size of $n = 400$. Figure 11 displays the misclassification error of AdaBoost based on hold out samples of size 1000 (also drawn *iid* on $[0,1]^{20}$) as a function of the iterations. The results are again averaged over 100 repetitions of the simulation.

As in Section 3.1, Adaboost with 8-node trees (thin, red curve) does not show any signs of overfitting while AdaBoost with stumps (thick, black curve) leads to overfitting. The overfitting

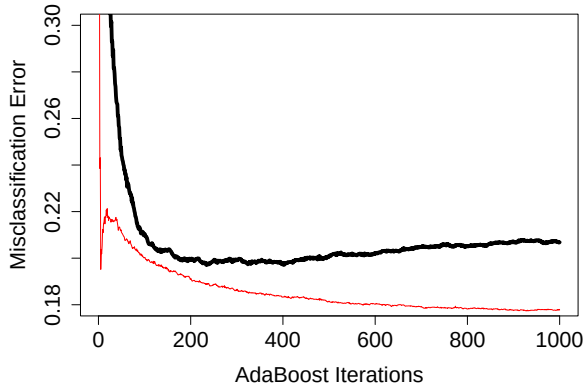


Figure 11: Comparison of AdaBoost with Stumps (Black, Thick) and 8-Node Trees (Red, Thin) for an Additive Bayes Rule

is evident in this experiment after about 400 iterations. Furthermore, AdaBoost with 8-node trees outperforms AdaBoost with stumps throughout the entire 1000 iterations. The misclassification error by 1000 iterations was smaller for the 8-node trees in 93 of the 100 simulations. The average (paired) difference in misclassification error after 1000 iterations was 0.029 with a standard error of $0.018/\sqrt{100} = 0.0018$. As before, since the simulation model used here has an additive Bayes decision rule, this evidence is directly at odds with the recommendation in Hastie et al. (2001) and Friedman et al. (2000a) that stumps are preferable for additive Bayes decision rules.

4.2 Should Smaller Trees Be Used When the Bayes Error is Larger?

As in Section 3.2, we observe that when we decrease the Bayes error rate from $q = 0.1$ to $q = 0$, the 8-node trees no longer have an advantage over the stumps. Figure 12 displays the results of the simulation in Section 4.1 using a Bayes error rate of $q = 0$. We see that the advantage of the 8-node trees has completely disappeared, and now the 8-node trees and stumps are indistinguishable. By 1000 iterations the misclassification errors for both are identical in all of the 100 repetitions.

Thus we see that the advantage of the larger trees in Section 4.1 is a result of the non-zero Bayes error, again suggesting that larger trees are in some way better at handling noisy data. This directly contradicts the conventional wisdom that boosting with larger trees is more likely to overfit on noisy data than boosting with smaller trees.

4.3 Should LogitBoost Be Used Instead of AdaBoost for Noisy Data?

We now rerun the experiment in Section 4.1 using AdaBoost and LogitBoost both with 8-node trees. Figure 13 displays the results with AdaBoost in red (thin) and LogitBoost in blue (thick). While LogitBoost performs better early on, it eventually suffers from overfitting near 400 iterations while AdaBoost shows no overfitting. Furthermore, the misclassification error for AdaBoost after 1000 iterations is (slightly) lower than the minimum misclassification error achieved by LogitBoost. After 1000 iterations the mean difference in misclassification error between LogitBoost and AdaBoost

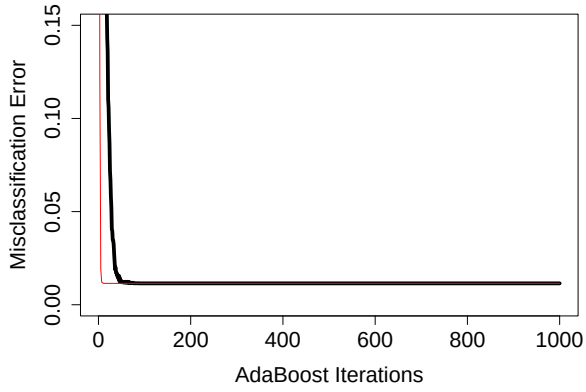


Figure 12: Comparison of AdaBoost with Stumps (Black, Thick) and 8-Node Trees (Red, Thin) for an Additive Bayes Rule with Zero Bayes Error

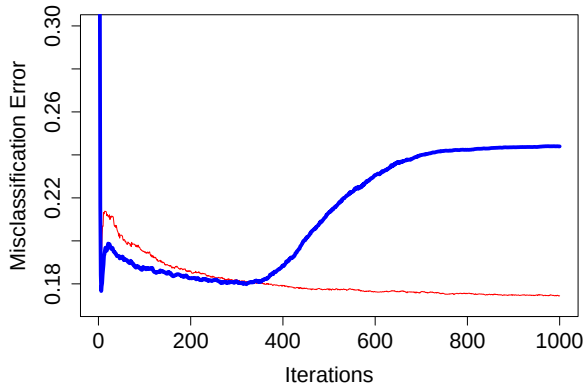


Figure 13: Comparison of AdaBoost (Red, Thin) and LogitBoost (Blue, Thick) with 8-Node Trees

was 0.069 with a standard error of $0.021/\sqrt{100}$=0.0021. The misclassification error for LogitBoost at 1000 iterations was larger than that of AdaBoost in all of the 100 repetitions.

Thus we again see that although LogitBoost was invented to perform better than AdaBoost for data with non-zero Bayes error, LogitBoost actually overfits the data while AdaBoost does not.

4.4 Should Early Stopping Be Used to Prevent Overfitting?

In this section we repeat the simulation from Section 3.4 using the new simulation model. Just as in Section 3.4 we use large $2^8 = 256$-node trees, a Bayes error rate of $q = 0.2$ and $d = 20$ dimensions. We now take twice the training sample size of Section 3.4 so that we have $n = 10,000$ points.

Figure 14 shows the resulting misclassification error averaged over 100 repetitions for hold out samples of size 1000. Although there is overfitting early on, the best performance is again achieved

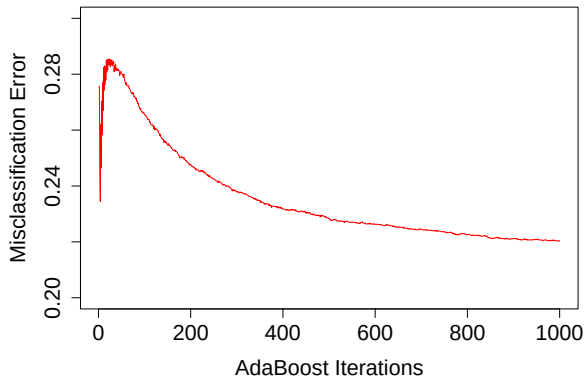


Figure 14: AdaBoost with 20% Bayes Error Using 256-Node Trees

by running the algorithm for the full 1000 iterations. We note that conventional early stopping rules here would be especially harmful since they would stop the algorithm after only a few iterations when the overfitting first takes place. Consequently any such early stopping rule would miss the optimal rule of running for the full 1000 iterations.

It should also be noted that the $2^8 = 256$-node trees used here are much richer than needed to fit the simple one-dimensional Bayes decision rule for this simulation model. Despite this, the misclassification error after 1000 iterations was lower than the misclassification error after the first iteration in all 100 of the reptitions. Thus it is again the self-averaging property of boosting that improves the performance as more and more iterations are run. Early stopping in this example would destroy the benefits of this property.

4.5 Should Regularization Be Based on the Loss Function?

As discussed in Section 3.5, regularization techniques for boosting such as early stopping are often based on minimizing a loss function such as the exponential loss in the case of AdaBoost. However, the performance of AdaBoost with regard to misclassification loss often has very little to do with the exponential loss function in practice.

In this section we examine the exponential loss for the experiment in Section 4.1 using 8-node trees. Figure 15 shows the increasing linear behavior for the log of the exponential loss for a single repetition of this experiment with a hold out sample of size 1000. Thus, just as in Section 3.5, the exponential loss increases exponentially as more iterations are run, while the misclassification error continues to decrease. Choosing regularization to minimize the exponential loss is again not useful for minimizing the misclassification error.

4.6 Should the Collection of Basis Functions Be Restricted to Prevent Overfitting?

In Section 3.6 we saw that restricting the number of observations in the terminal nodes of the trees to be at least 15 degraded the performance of AdaBoost, despite the common belief that such restrictions should be beneficial. In this section we rerun the experiment in Section 4.1 but again consider this same restriction.

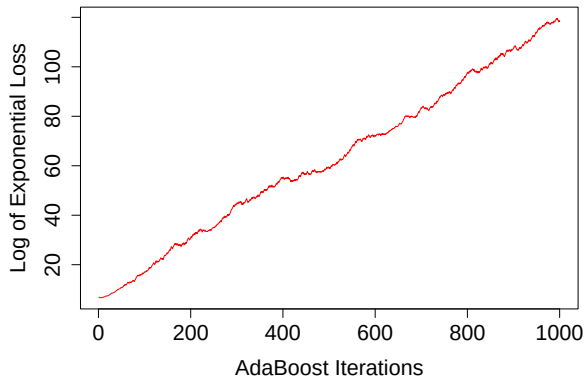


Figure 15: The Log of the Exponential Loss for AdaBoost on a Hold Out Sample

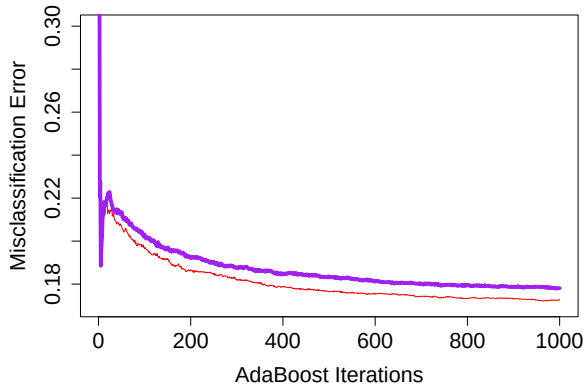


Figure 16: Comparison of AdaBoost with 8-Node Trees (Red, Thin) to AdaBoost with 8-Node Trees Restricted to Have at Least 15 Observations in the Terminal Nodes (Purple, Thick)

Figure 16 shows the results with the unrestricted 8-node trees given by the red (thin) curve and the 8-node trees restricted to have at least 15 observations in the terminal nodes given by the purple (thick) curve. As in Section 3.6, degradation in performance is evident. The mean difference in misclassification error at 1000 iterations was 0.005 with a standard error of $0.010/\sqrt{100}$=0.001. AdaBoost with unrestricted 8-node trees gave a lower misclassification error at 1000 iterations in 65 of the 100 repetitions for this simulation model.

4.7 Should Shrinkage Be Used to Prevent Overfitting?

In Section 3.7 we saw that shrinkage actually caused AdaBoost to overfit in a situation where it otherwise would not have, in spite of the popular belief that shrinkage prevents overfitting. In this section we rerun the experiment in Section 4.1 with 8-node trees again using a shrinkage value of

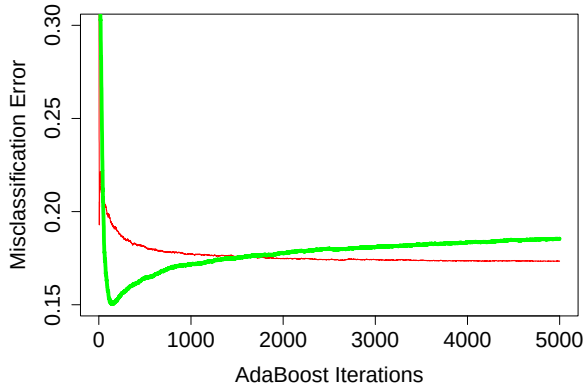


Figure 17: Comparison of AdaBoost (Red, Thin) and AdaBoost with Shrinkage (Green, Thick)

$\nu = 0.1$. Figure 17 shows the results with the red (thin) curve corresponding to no shrinkage and the green (thick) curve showing the results for shrinkage. The plot shows that again shrinkage causes overfitting.

It is interesting to note that in this simulation, unlike the simulation in Section 3.7, shrinkage has the beneficial effect of producing a lower misclassification error very early on in the process, despite causing the eventual overfitting. This suggests that a stopping rule which could accurately estimate the optimal number of iterations combined with shrinkage may prove very effective for this particular simulation. As a result of the good performance early on, the shrinkage actually gives a lower misclassification error at our chosen stopping point of 1000 iterations than without the shrinkage. However, if we run for enough iterations (the plot shows 5000 iterations) the overfitting caused by the shrinkage eventually overwhelms this advantage. By 5000 iterations the shrinkage leads to a larger misclassification error in 87 of the 100 repetitions. The mean difference in misclassification error at 5000 iterations was 0.012 with a standard error of $0.012/\sqrt{100}$=0.0012.

4.8 Is Boosting Estimating Probabilities?

In Section 3.8 we saw that the probability estimates suggested by Friedman et al. (2000a) for AdaBoost diverge quickly to 0 and 1 and consequently perform very poorly even for cases where the AdaBoost classification rule performs well. In this section we examine the probability estimates for a single repetition of the experiment in Section 4.1 on a hold out sample of size 1000.

The two histograms in Figure 18 show the resulting probability estimates for $m = 10$ iterations and $m = 1000$ iterations respectively using 8-node trees. Both histograms have 100 equal width bins. At 10 iterations the estimates have not yet diverged, but by 1000 iterations almost all of the probability estimates are greater than 0.99 or less than 0.01, just as we saw in Section 3.8. As before, this indicates a poor fit since with this simulation model all of the true probabilities are either 0.1 or 0.9.

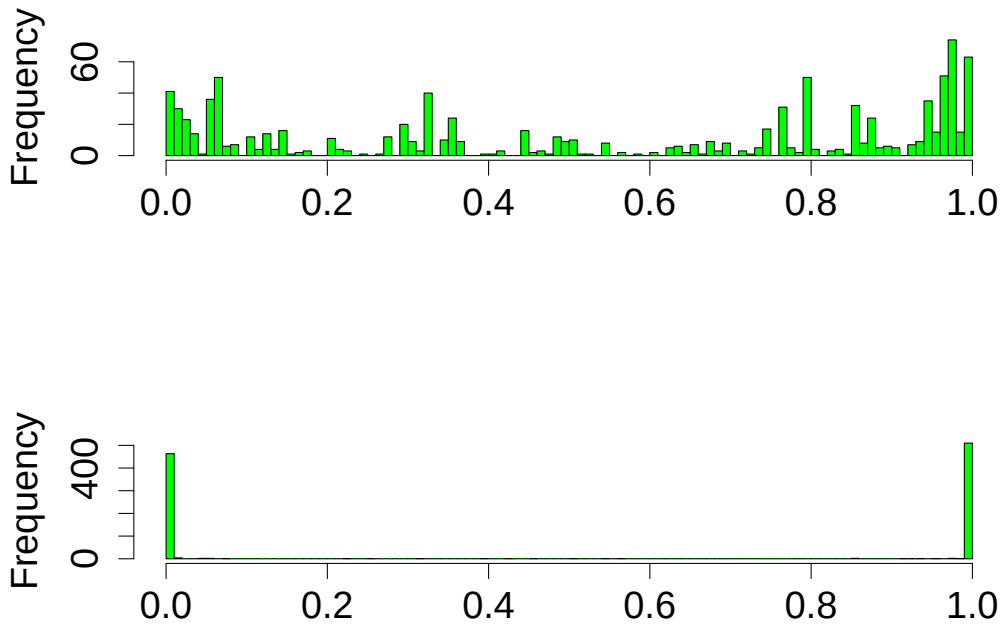


Figure 18: The Probability Estimates From AdaBoost at $m = 10$ Iterations (Top) and $m = 1000$ Iterations (Bottom)

4.9 Is Boosting Similar to the One Nearest Neighbor Classifier?

In Section 3.9 we saw that despite the fact that boosting agrees with the nearest neighbor classifier on all the training data, its performance elsewhere is quite different for $d > 1$ dimensions. For AdaBoost, areas surrounding points in the training data for which the observed class differs from that of the Bayes rule are classified according to the Bayes rule more often than they would be using the nearest neighbor rule.

We illustrate this again using $d = 2$ dimensions for visualization purposes. We use a Bayes error rate of $q = 0.1$ and take $n = 400$ points spread out evenly according to a Latin hypercube design. The plot on the left in Figure 19 shows the resulting classification rule of AdaBoost with 8-node trees at 1000 iterations for a single repetition using the new simulation model. The plot on the right shows the nearest neighbor rule. Both plots use the same color scheme as Figure 9. For the nearest neighbor rule, 21% of the points in the hold out sample disagree with the Bayes rule. This number is only 6% for AdaBoost, despite the fact that both classifiers classify every point in the training data according to the observed class label.

The difference between AdaBoost and the nearest neighbor rule is also well illustrated by other experiments in Section 4. For instance, in Section 4.1 the misclassification error for the nearest neighbor classifier was 0.499 but only 0.178 for AdaBoost with the 8-node trees.

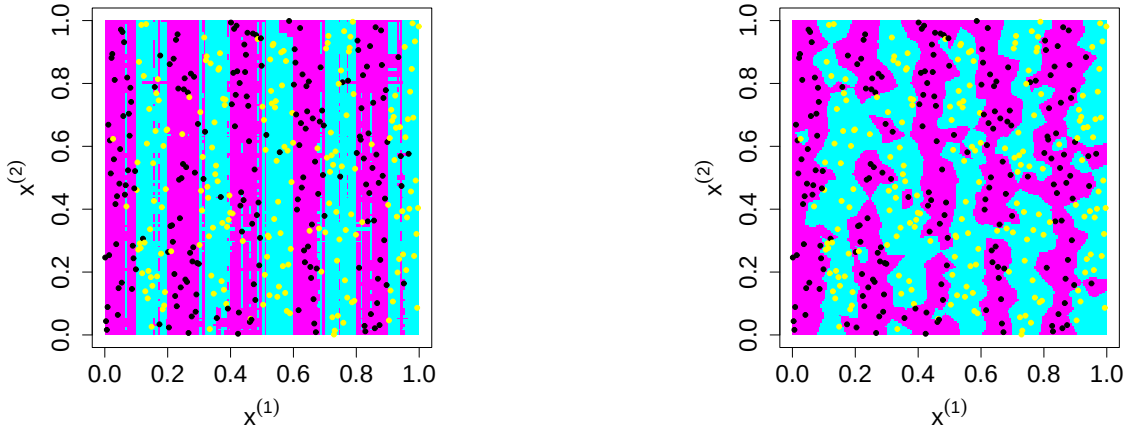


Figure 19: Comparison of AdaBoost (Left) and Nearest Neighbor (Right) with 10% Bayes Error

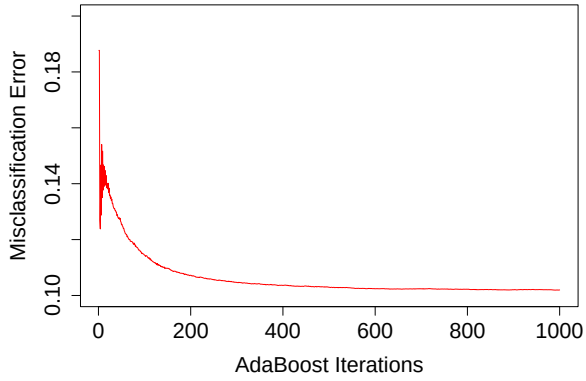


Figure 20: Performance of AdaBoost for a Simulation with a Bayes Error of 0.1

4.10 Is Boosting Consistent?

In Section 3.10 we illustrated that with a large sample size n, the misclassification error for AdaBoost can come quite close to the Bayes error rate, despite the fact that AdaBoost fits the training data perfectly. We illustrate this again in this section. As in Section 3.10, we use 2^8-node trees and a Bayes error rate of $q = 0.1$ but now take $n = 10,000$ and use the new simulation model.

Figure 20 shows the misclassification error averaged over 100 repetitions using hold out samples of size 1000. The mean misclassification error after 1000 iterations was 0.102 with a standard error of $0.009/\sqrt{100}$=0.0009. As we saw in Section 3.10, this is extremely close to the Bayes error rate and much less than the nearest neighbor bound of $2q(1-q) = 0.18$. We encourage readers to rerun the simulation with larger n to make the misclassification error even closer to the Bayes error.

5. Additional Experiments Which Contradict the Statistical View of Boosting

As mentioned at the beginning of Section 4, we encourage the reader to try simulation models other than those considered in this paper by using the R code provided on the web page http://www.davemease.com/contraryevidence. The simulation model can be specified by changing only three lines of this code in most cases. We have only considered two simulation models in this paper due to space constraints.

One criticism of the two simulation models considered in this paper is that both have a discontinuous (piecewise constant) conditional class probability function $p(x) \equiv \mathrm{P}(Y = 1|x)$. An argument can be made that both AdaBoost and LogitBoost can not provide a good fit to these models because of the discontinuities. To investigate this, we examined additional experiments from the simulation model specified by

$$p(x) = 1/(1 + e^{k(\sum_{j=1}^{J} x^{(j)} - J/2)})$$

where J is the number of effective dimensions as in Section 3 and k is a constant which determines the Bayes error rate. We note that this model has the same Bayes decision boundary as the model in Section 3 but now has a smooth conditional class probability function without any discontinuities. The results for this model are not included in the paper but are qualitatively extremely similar to those in Section 3. We encourage the reader to investigate this further.

6. Concluding Remarks and Practical Suggestions

By way of the simulations in Sections 3 and 4 we have seen that there are many problems with the statistical view of boosting and practical suggestions arising from that view. We do not endeavor to explain in this paper why these inconsistencies exist, nor do we offer a more complete view of boosting. Simply put, the goal of this paper has been to call into question this view of boosting that has come to dominate in the statistics community. The hope is that by doing so we have opened the door for future research toward a more thorough understanding of this powerful classification technique.

The statistical view of boosting focuses only on one aspect of the algorithm - the optimization. A more comprehensive view of boosting should also consider the stagewise nature of the algorithm as well as the empirical variance reduction that can be observed on hold out samples as with the experiments in this paper. Much insight on such ideas can be gained from reading work by the late Leo Breiman (e.g., Breiman, 2000, 2001) who subsequently abandoned interest in boosting and went on to work on his own classification technique known as Random Forests. The Random Forests algorithm achieves variance reduction directly through averaging as opposed to AdaBoost for which the variance reduction seems to happen accidently.

While we do not offer much in the way of an explanation for the behavior of AdaBoost in this paper, we will conclude with some practical advice in light of the evidence presented. First of all, AdaBoost remains one of, if not *the*, most successful boosting algorithms. One should not assume that newer, regularized and modified versions of boosting are necessarily better. We encourage readers to try standard AdaBoost along with these newer algorithms. If AdaBoost is not available as an option in your preferred software package, it is only a few lines of code to write yourself. Secondly, if classification is your goal, the best way to judge the effectiveness of boosting is by monitoring the misclassification error on hold out (or cross-validation) samples. We have seen that other loss functions are not necessarily indicative of the performance of boosting's classification

rule. Finally, much of the evidence we have presented is indeed counter-intuitive. For this reason, a practitioner needs to keep an open mind when experimenting with AdaBoost. For example, if stumps are causing overfitting, be willing to try larger trees. Intuition may suggest the larger trees will overfit even more, but we have seen that is not necessarily true.

Acknowledgments

D. Mease's research was supported by an NSF-DMS post-doctoral fellowship. The authors are grateful to Andreas Buja and Abba Krieger for their help and guidance.

References

L. Breiman. Discussion of additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:374–377, 2000.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

P. Buhlmann and B. Yu. Boosting with the L_2 loss: Regression and classification. *Journal of the American Statistical Association*, 98:324–339, 2003.

A. Buja. Discussion of additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:387–391, 2000.

A. Buja, W. Stuetzle, and Y. Shen. Loss functions for binary class probability estimation and classification: Structure and applications. 2006.

M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. In *Computational Learing Theory*, pages 158–169, 2000.

A. Cutler and G. Zhao. Pert: Perfect random tree ensembles. *Computing Science and Statistics*, 33: 490–497, 2001.

M. Dettling and P. Buhlmann. Boosting for tumor classification with gene expression data. *Bioinformatics*, 19:1061–1069, 2003.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

Y. Freund and R. E. Schapire. Discussion of additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:391–393, 2000.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:337–374, 2000a.

J. Friedman, T. Hastie, and R. Tibshirani. Rejoiner for additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:400–407, 2000b.

A. J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 692–699, 1998.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

W. Jiang. Does boosting overfit: Views from an exact solution. *Technical Report 00-03, Department of Statistics, Northwestern University*, 2000.

W. Jiang. Is regularization unnecessary for boosting? In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, pages 57–64, 2001.

W. Jiang. On weak base hypotheses and their implications for boosting regression and classification. *Annals of Statistics*, 30:51–73, 2002.

A. Krieger, C. Long, and A. J. Wyner. Boosting noisy data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 274–281, 2001.

G. Lugosi and N. Vayatis. On the bayes-risk consistency of regularized boosting methods. *Annals of Statistics*, 32:30–55, 2004.

D. Mease, A. Wyner, and A. Buja. Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research*, 8:409–439, 2007.

G. Ridgeway. Discussion of additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:393–400, 2000.

G. Ridgeway. Generalized boosted models: A guide to the gbm package. 2005.

R. E. Schapire, Y. Freund, P. L. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26:1651–1686, 1998.

T. Zhang and B. Yu. Boosting with early stopping: Convergence and consistency. *Annals of Statistics*, 33:1538–1579, 2005.

Journal of Machine Learning Research 9 (2008) 157-164 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008

Kristin P. Bennett BENNEK@RPI.EDU
Department of Mathematical Sciences
Rensselear Polytechnic Institute
Troy, NY 12180, USA

Editor: Yoav Freund

1. Introduction

Mease and Wyner (MW) argue experimentally that the statistical view of boosting does not account for its success and that following the now conventional wisdom arising from this view in Friedman et al. (2000) does not necessarily lead to choices in the boosting algorithm that improve generalization. The authors did an excellent job of defining a set of experiments in which small changes in the boosting algorithm (such as changing the hypothesis space, loss function, and shrinkage) produce significant changes in generalization that were unintuitive given the statistical view of AdaBoost (Freund and Schapire, 1996) expressed in Friedman et al. (2000).

The authors state "The statistical view focuses only on one aspect of the algorithm - the optimization." But one can argue just the opposite, that some of the problems and surprises come from not enough of the optimization perspective instead of too much. Analyzing AdaBoost's performance as an optimization algorithm in terms of convergence rates and optimality conditions (measured on the training data) can be quite revealing. First, we observe that the experiments in MW make dramatic changes in the convergence rates of AdaBoost and that these convergence rates are closely associated with the margin of the classifier. AdaBoost may avoid overfitting for two completely different reasons. Sometimes the algorithm is converging so slowly that stopping at a large number of iterations is still early stopping. At other times, AdaBoost converges relatively quickly and is in essence "overtrained" way past reasonable measures of the optimality conditions. In this case the classifier has converged and is no longer changing much, so the classifier does not overfit. Indeed, some overtraining appears to help improve the classifier slightly. Second, we observe that AdaBoost cannot be trained forever. For the separable case, overtraining AdaBoost and LogitBoost will eventually produce numeric problems that can produce artifacts in the generalization error. In Experiments 3.3 and 4.3 in MW, LogitBoost was overtrained to the point of failure. The so called overfitting observed for LogitBoost was really an algorithmic issue that is quite fixable. If LogitBoost is stopped appropriately or another stepsize strategy is used, the results for LogitBoost are as good as or better than those for AdaBoost. More discussion of these results can be found below.

2. A Mathematical Programmer's View of AdaBoost

AdaBoost optimizes a linear combination of weak hypotheses with respect to the exponential loss. AdaBoost is a coordinate descent (CD) algorithm, that iteratively optimizes the problem with respect to one hypothesis at a time using column generation (Bennett et al., 2000). The weak learner seeks the hypothesis that maximizes the inner product with the function gradient (Mason et al., 2000). The convergence properties of such coordinate descent algorithms have been extensively studied in the mathematical programming community and a full analysis of relevant CD results and their extension to the boosting case can be found in Rätsch (2001).

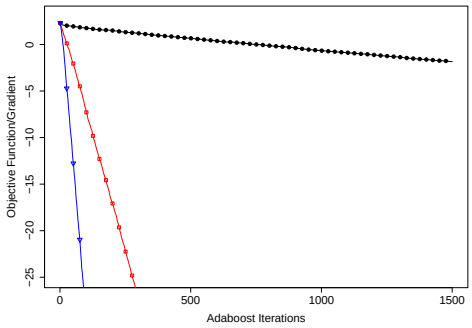
Thus from the mathematical programming perspective, we know AdaBoost inherits both the beneficial and potentially problematic properties of CD. We know from both the CD and original AdaBoost theoretical results that the AdaBoost *objective* converges linearly to the optimal objective. The simplicity of CD and its suitability for column generation make coordinate descent an attractive algorithm, but in practice coordinate descent is not widely used because it can be very slow and it has a tendency to cycle. CD guarantees that the objective function converges to the minimum but there is no guarantee that optimal hypothesis coefficients are attained, and cycling is possible. The AdaBoost loss function is particularly problematic since the exponential function is not strongly convex and the Hessian is rank deficient when the size of the hypothesis space exceeds the number of points. Overall, mathematical programming tells us that we can expect the AdaBoost objective value to converge linearly and the convergence rate to be slow, especially when cycling occurs. The paper on the dynamics of AdaBoost (Rudin et al., 2004) investigates this cycling behavior.

The MW experiments focus on the degenerate case in which the optimal objective value of the underlying exponential optimization problem is zero. LogitBoost and AdaBoost are functions of the form $min_{\alpha} J(f)\ s.t.\ f = Ha$ where H is the hypothesis space matrix containing all possible weak learners for that data set. In every case, there exists some linear combination of weak learners that classifies the points with no error, and therefore the objective can be driven to zero. The AdaBoost exponential loss function is $\sum \exp(-y_i f_i)$. The function space gradient is $\frac{\partial J(f)}{\partial f_i} = -y_i exp(-y_i f_i)$. Note the 1-norm of the function space gradient is the same as the objective, $\left\| \frac{\partial J(f)}{\partial f} \right\|_1 = \sum \exp(-y_i f(x_i))$ for two-class classification. The optimality condition is that the gradient with respect to α is zero, $\frac{\partial J(Ha)}{\partial \alpha} = H' \frac{\partial J(f)}{\partial f} = 0$. In theory, to check this gradient we need to know the weak learners for the full hypothesis space, H. But, for cases where the misclassification error is driven to 0, it is sufficient to monitor the gradient in function space. Fortunately, the norm of the function space gradient provides an upper bound on the norm of the true gradient since $\left\| H' \frac{\partial J(f)}{\partial f} \right\| \leq C \left\| \frac{\partial J(f)}{\partial f} \right\|$ for some fixed $C > 0$.

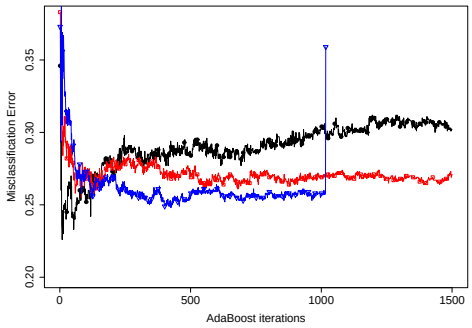
From a mathematical programming perspective, we are optimizing a degenerate, poorly-scaled problem for which the optimal objective value of 0 can only be achieved in the limit using a slower algorithm prone to cycling that may become numerically unstable. Clearly, convergence of the algorithm should be monitored closely. Yet, in most machine learning boosting papers, the focus is on generalization for a fixed number of iterations and rarely on optimization performance.

3. Convergence Rate of AdaBoost

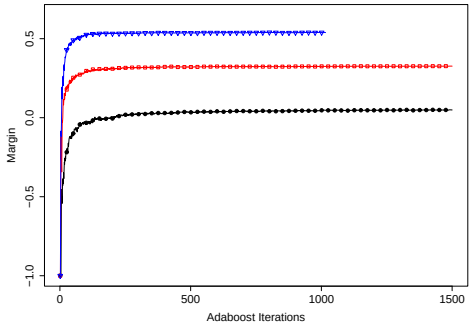
Let's examine the convergence rate and optimality conditions of AdaBoost in the MW experiments. Figure 1 contains three plots, one each for the log base 10 of the objective (or equivalently the 1-



(a) Objective/Gradient (log10)



(b) Test Error

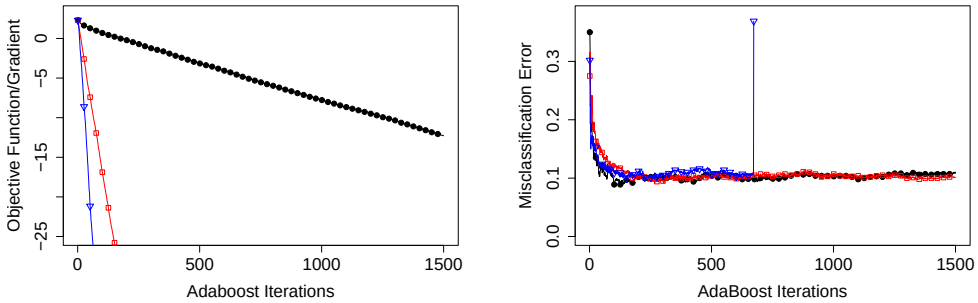


(c) Margin

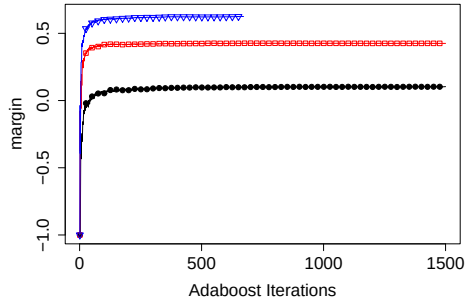
Figure 1: 1 trial of Experiment 3.1 (10% Bayes Error) for AdaBoost + stumps (black, dots), AdaBoost + 8-node trees (red, squares), and AdaBoost + 16-node trees (blue, triangles).

norm of the gradient), the testing error, and the training margin ($\frac{\min_i(y_i f_i)}{\sum_m \alpha_m}$) for 1500 iterations for the first trial of the experiment with 10% Bayes error in section 3.1 of MW. The graphs contain results for AdaBoost with stumps (black, dots), AdaBoost with 8-node trees (red, squares) and AdaBoost with 16-node trees (blue, triangles). Observe that the loss function and gradient are driven to zero for all three hypothesis spaces and that the remarkably different convergence rates are inversely proportional to the size of the trees being boosted. The results for AdaBoost with 16-node trees end at 1017 iterations because the objective becomes less than 10^{-322}, so a divide-by-zero error occurs. In general, AdaBoost with bigger trees achieves bigger margins and obtains better generalization. AdaBoost with stumps converges incredibly slowly and arguably should be run for more than 1500 iterations if early stopping is not desired.

Figure 2 contains the same three graphs for the first trial for the experiment with no Bayes Error in section 3.2. The objective/gradient and margin graphs are qualitatively similar for experiments 3.1 and 3.2. Note that the 16 node tree Adaboost algorithm underflows at 673 iterations. The testing error graph for experiment 3.2 is quite different. The performance for AdaBoost with stumps is much improved and now competitive or better than AdaBoost with 8 or 16 node trees. Here the margin results do not predict which type of boosted trees will generalize best. MW found at least one simple example in which margins don't work well.



a) Objective/Gradient (log10) (b) Test Error



(c) Margin

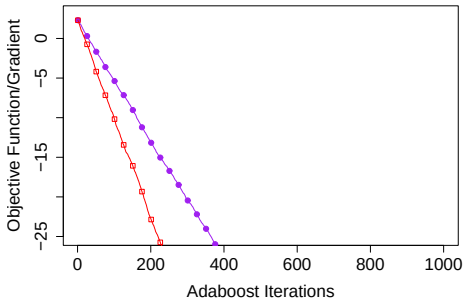
Figure 2: 1 trial of Experiment 3.2 (0 Bayes Error) for AdaBoost + stumps (black, dots), AdaBoost + 8-node trees (red, squares), and AdaBoost + 16-node trees (blue, triangles).

Figure 3 shows the results for AdaBoost with 8-node trees and AdaBoost with 8-node trees restricted to 15 nodes for the first trial of experiment 3.6. Here we see that restricting the trees slows convergence, decreases margins, and increases error.

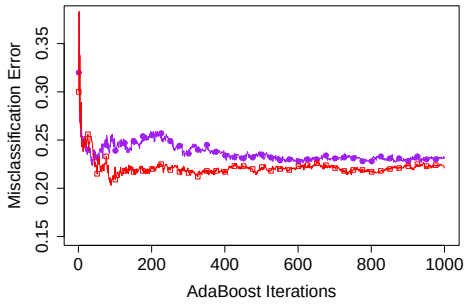
Figure 4 shows the results for AdaBoost with 8-node trees with no shrinkage (red, squares), .1 shrinkage (purple, dots), and .5 shrinkage (blue, triangles) for the first trial of experiment 3.3 (10 % Bayes error). Here we see that shrinkage can speed up or slow down the convergence rates. For .1 shrinkage compared to no shrinkage, the convergence rate was slower, the margin smaller, and the test error larger. For .5 shrinkage, the convergence rate was faster and the margin was larger than for the .1 shrinkage case. If the .5 shrinkage algorithm is terminated at using reasonable stopping criteria, the performance is quite comparable with the no shrinkage case, and improved over the .1 shrinkage case.

We present the following conjectures based on observations of this and other MW experiments for the separable case and leave fuller investigation to later work.

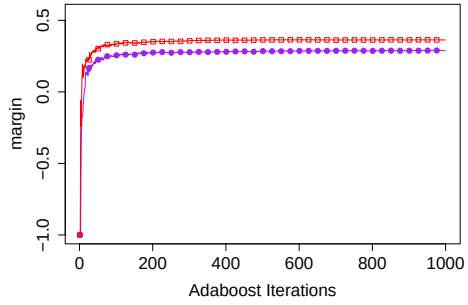
- *The convergence rate of AdaBoost is dependent on the space spanned by the weak learner and larger hypothesis spaces converge faster.* The weak learners produced by stumps are a subset of those from the 8-node decision tree which are in turn a subset of those produced by the 16-node decision tree. The bigger the decision tree, the better the weak learner can match



(a) Objective/Gradient (log10)



(b) Test Error

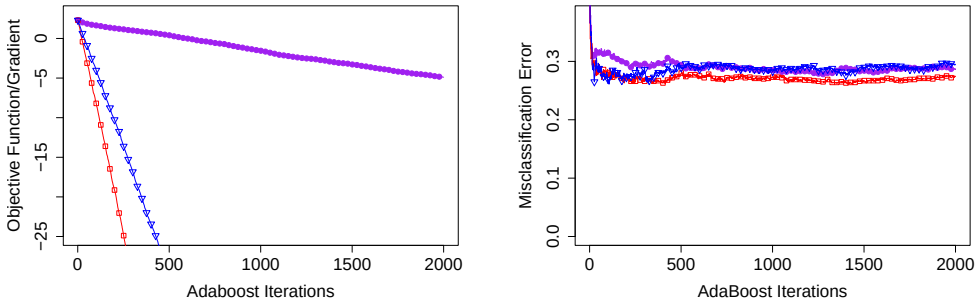


(c) Margin

Figure 3: 1 trial of Experiment 3.6 (10% Bayes Error) for AdaBoost + 8-node trees (red, squares), and AdaBoost + 8-node trees restricted to at least 15 observations in terminal nodes (purple, dots).

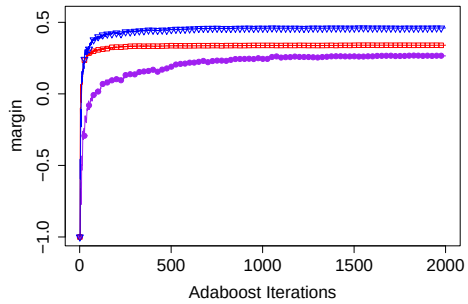
the gradient at each iteration (as reflected by weighted misclassification error). So AdaBoost can obtain a better decrease in the objective value. This conjecture is also supported by the fact that in Figure 3's run of experiment 3.6, decreasing the hypothesis space by restricting the terminal node size, also reduced the convergence rate.

- *For a fixed separable problem, faster convergence rates of AdaBoost can result in larger margins.* AdaBoost is known to approximately optimize the margin as measured by the 1-norm (Rosset et al., 2004; Schapire et al., 1998). The objective decreases the numerator of the margin and the iterations increase the denominator, so getting a smaller objective quicker creates a better margin. Bigger hypothesis spaces allow bigger steps resulting in larger margins. This finding is also supported by the fact that when shrinkage is used to change the convergence rate, the resulting margins changed as well (see Figure 4). For problems with no training error, we expect larger margins to translate to better generalization rates. But MW's experiments 3.2 and 4.2 show that this is not always the case. Figure 2 shows the margin for 1 run of Experiment 3.2. So MW are quite right in their conclusion that there is more to the



(a) Objective/Gradient (log10)

(b) Test Error



(c) Margin

Figure 4: 1 trial of Experiment 3.7 (10% Bayes Error) for AdaBoost + 8-node trees (red, squares), AdaBoost + 8-node trees with .5 shrinkage (blue, triangles) and AdaBoost + 8-node trees with .1 shrinkage (purple, dots).

generalization of AdaBoost then just optimizing the loss. Adding consideration of the margin is not enough either.

- *For slowly converging problems, AdaBoost will frequently be regularized by early stopping.* In experiments 3.1 and 3.2, AdaBoost with stumps is overfitting and the early stopping in MW at 1000 iterations helps the generalization error. For this specific experiment, the slow convergence is a result of cycling. For the first trial in experiments 3.1 and 3.2, AdaBoost with stumps only generated 158 and 156 distinct weak learners in 1000 iterations respectively. The weak learners generated by AdaBoost with 8-node trees and 16-node trees were distinct except for 2. By cycling through relatively few weak learners, AdaBoost with stumps strongly weights a few trees. This appears to be bad for generalization in experiment 3.1 and good for generalization for the no noise case in experiment 3.2.

- *For more rapidly converging problems, AdaBoost will converge and enter an overtraining phase.* For the larger tree cases, the objective and margins converge rapidly. Typically one would halt an optimization algorithm when the gradient became near 0. In the MW experiments, AdaBoost with 8-node trees is overtrained past the point where one would normally

halt an optimization algorithm based on gradient criteria (Gill et al.). AdaBoost doesn't overfit in this overtraining phase because it has converged and only very small changes are being made. Perhaps the overtraining phase contributes to the robustness of AdaBoost, since AdaBoost is performing the self-averaging discussed in MW and acting more like bagging. In the MW experiments, AdaBoost achieves better generalization when trained to an extraordinarily high degree of accuracy, a fact contrary to the usual loose convergence criteria used in support vector machines (Bennett and Parrado-Hernández, 2006). But care must be taken to halt the boosting algorithm before the overtraining produces numeric problems due to finite precision problems. As shown in Figure 1, AdaBoost with 16-node trees underflows at 1017 iterations for the 10% Bayes error case and at 673 iterations for the 0 error case. AdaBoost with 8-node trees also underflows eventually as well.

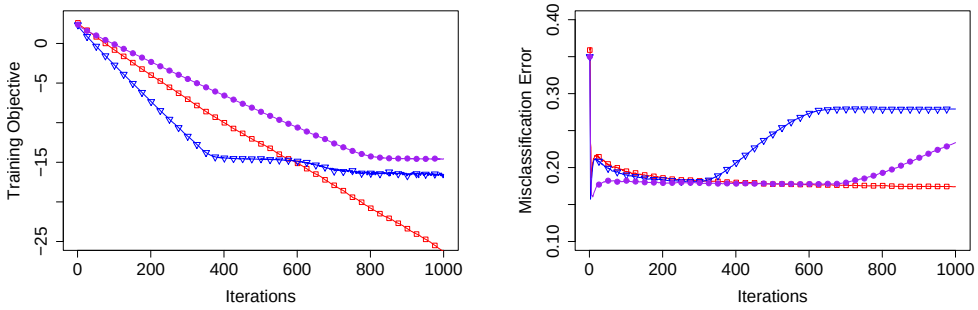
4. LogitBoost versus AdaBoost

Experiments 3.4 and 4.4 compare LogitBoost and AdaBoost and conclude LogitBoost overfits. Tracking the convergence of LogitBoost shows that this is not quite the case. We show our results repeating experiment 4.4 exactly as in the paper for AdaBoost and LogitBoost. Recall LogitBoost differs from AdaBoost in two ways. First, it uses the logistic loss instead of the exponential loss and second, it uses a Newton step instead of an exact step size. The Newton step for logistic loss works out to be 1/2 at each iteration. AdaBoost's stepsize is adaptive. The CD convergence results do not apply directly to LogitBoost as implemented in the paper because of the Newton step.

Figure 4 shows the average objective and misclassification results for 100 trials with 8-node trees. Note that at about 375 iterations, LogitBoost fails to obtain a decrease in the objective because the Newton step is too large when the objective is very small. From that point, the testing error declines. LogitBoost with shrinkage converges more slowly, so it can go more iterations before the step size fails. Once the objective becomes too small, the stepsize fails and the generalization performance of LogitBoost decreases remarkably. The LogitBoost objective is still small and continues to decrease slightly, but the self-averaging properties observed in AdaBoost in the overtraining phase are lost. Note that up until it missteps, LogitBoost is very competitive with AdaBoost. If LogitBoost and AdaBoost were halted at the same high degree of accuracy (e.g., 10^{-8}), there is no evidence of overfitting.

5. Conclusion

MW are correct is saying that optimization provides only part of the picture because optimization tells us nothing about generalization. Mathematical programming theory tells us that more well-posed boosting problems with well-conditioned loss functions (like the hinge loss) and explicit regularization in the objective should produce boosting algorithms with better behavior from an optimization perspective. But AdaBoost's ill-conditioning appears to be one of the secrets of its success. More investigation is needed comparing Adaboost with its regularized counterparts. Certainly machine learning researchers should mind their optimization theory and track the convergence of their algorithms. Optimality conditions should be used to halt and compare boosting algorithms instead of fixed iteration limits.



(a) Objective (log10) (b) Test Error

Figure 5: Average of 100 trials of Experiment 4.3 (10% Bayes Error) for AdaBoost (red, squares), LogitBoost (blue, triangles), and LogitBoost with .5 shrinkage (purple, circles) for 8-node trees.

References

K.P. Bennett and E. Parrado-Hernández. The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, 7:1265–1281, 2006.

K.P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 65–72, 2000.

Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, 148:156, 1996.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting (With discussion and a rejoinder by the authors). *Ann. Statist*, 28(2):337–407, 2000.

P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, London and New York.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. *Advances in Neural Information Processing Systems*, 12:512–518, 2000.

G. Rätsch. *Robust Boosting via Convex Optimization Theory and Applications*. PhD thesis, Universitat Potsdam, 2001.

S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.

C. Rudin, I. Daubechies, and R.E. Schapire. The dynamics of AdaBoost: Cyclic behavior and convergence of margins. *Journal of Machine Learning Research*, 5:1557–1595, 2004.

R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

Journal of Machine Learning Research 9 (2008) 165-170 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008

Andreas Buja BUJA.AT.WHARTON@GMAIL.COM
Statistics Department
The Wharton School, University of Pennsylvania
Philadelphia, PA 19104-6340

Werner Stuetzle WXS@STAT.WASHINGTON.EDU
Statistics Department
University of Washington
Seattle, WA 98195-4322

Editor: Yoav Freund

We thank the authors for writing a thought-provoking piece that may ruffle the feathers of recent orthodoxies in boosting. We also thank JMLR for publishing this article! Since the late 1990s, boosting has undergone the equivalent of a simultaneous X-ray, fMRI and PET exam, and the common view these days is that boosting is a kind of model fitting. As such, it is subjected to assumptions that are common in non-parametric statistics, such as: limiting the complexity of the base learner, building up complexity gradually by optimization, and preventing overfitting by early stopping or by regularizing the criterion with a complexity penalty. The theories backing this up use VC dimensions and other measures to show that, if the complexity of fits grows sufficiently slowly, asymptotic guarantees can be given. Into this orthodox scene Mease and Wyner throw one of the most original mind bogglers we have seen in a long time: "if stumps are causing overfitting, be willing to try larger trees." In other words, if boosting a low-complexity base learner leads to overfit, try a higher-complexity base learner; boosting it might just not overfit. Empirical evidence backs up the claim.

Is this counterintuitive wisdom so surprising? Yes, if seen from the point of view of orthodoxy, but less so when reviving some older memories. We may remind ourselves how boosting's fame arose in statistics when the late Leo Breiman stated in a discussed 1998 Annals of Statistics article (based on a 1996 report) that boosting algorithms are "the most accurate ... off-the-shelf classifiers on a wide variety of data sets." We should further remind ourselves what this praise was based on: boosting of the full CART algorithm by Breiman himself, and boosting of the full C4.5 algorithm by others. In other words, the base learners were anything but 'weak' in the sense of today's orthodoxy, where 'weak' means 'low complexity, low variance, and generally high bias.' (Few people today use PAC theory's untenable notion of weak learner, which was gently demolished by Breiman in the appendix of this same article.) Breiman's (1998b, p. 802) 02) major conclusion at the time was: "The main effect of both bagging and [boosting] is to reduce variance." It appears, therefore, that his notion of 'weak learner' was one of 'high complexity, high variance, and low bias'! This was before the low-variance orthodoxy set in and erased the memories of the early boosting experiences.

Unfortunately, soon thereafter Breiman saw his own assumptions thrown into question when he learned from Schapire et al.'s (1998) work that excellent results could also be achieved by boosting

stumps. This experience was later reinforced when Friedman et al. (2000) introduced the interpretation of boosting as model fitting: the base learner now had to be weak in the sense of low variance. Ever since, theoretical attempts at 'explaining boosting' have relied on low complexity of the base learner and controlling complexity of the final classifier to assure good generalization properties. These 'explanations,' however, have never been able to explain why boosting is relatively immune to overfitting, even when not stopped and not regularized and used with a high complexity base learner.

Mease and Wyner's achievement is to pull the messy truth out from under the rug of the low-variance orthodoxy. They do so with the equivalent of boy scout tools, some simple but telling simulations, which reinforce the idea that our reasonings about early stopping, regularization, low variance of the base learner, and the specifics of the surrogate loss function, are not or not the only essence of boosting. To explain why this is so, Mease and Wyner do not give us hard theory, but they point in a direction, essentially by recovering memories that predate the low-variance orthodoxy: "self-averaging" for variance reduction, which is the principle behind bagging and random forests.

While variance reduction is an aspect that has been ignored by the low-variance orthodoxy, the orthodoxy's implicit dogma, that boosting can reduce bias, is also true. As asserted and documented empirically a decade ago by Schapire et al. (1998, Section 5.3), boosting can do both. Depending on the data and the base learner, the effect that dominates may be bias reduction *or* variance reduction. In this regard Schapire et al.'s (1998) simulation results as summarized in their Table 1 (p. 1673) are illuminating, and had we taken them seriously sooner, we would be less surprised by Mease and Wyner's messages. Arguing against Breiman (1998b), Schapire et al. used the table to make the now orthodox point that boosting can reduce bias. An unprejudiced look shows, however, that the winner in all four scenarios is boosting C4.5, not boosting stumps, and when C4.5 is the base learner the overwhelming story is indeed variance reduction. With this information, the Mease-Wyner mind boggler is a touch less mind boggling indeed: From the combined evidence of Breiman (1998b) and Schapire et al. (1998), we should expect that boosting high-variance base learners generally outperforms boosting low-variance base learners. For the practitioner the recommendation should be to boost CART or C4.5. In theoretical terms, one should let most of the bias removal be done by the base learner and take advantage of boosting's variance removal; at the same time, boosting may further reduce the base learner's bias by another notch if that is possible.

Where does this leave us in terms of theory? The implications of Mease and Wyner's unorthodoxies stand: Complexity controlling theories of bias removal are off the mark; they are not incorrect but misleading, and they ignore a whole other dimension that matters hugely for the practice of boosting. What we need is a theory that explains bias and variance reduction in a single framework. We do not even know of a unified general theory of variance reduction, although some interesting work has been done in the area of bagging (Bühlmann and Yu, 2002) and random forests (Amit et al., 2001). The real jackpot, however, would be a theory that explains how and when boosting reduces bias *and* variance.

Meanwhile we are left with some tantalizing clues, above all Breiman's hunch (1999, p. 3): "AdaBoost has no random elements .... But just as a deterministic random number generator can give a good imitation of randomness, my belief is that in its later stages AdaBoost is emulating a random forest." If born out, this conjecture would have theoretical and practical implications. For one, it would mean that the initial stages of boosting may remove bias, whereas the later stages remove variance. According to Breiman (1998b, p. 803), boosting a high-variance base learner does not yield convergence but exhibits "back and forth rocking" of the weights, and "This variability

may be an essential ingredient of successful [boosting] algorithms." Breiman implies that at some point boosting iterations turn into a pseudo-random process whose behavior may resemble more the purely random iterations of bagging than those of a minimization process. This random process may be able to achieve the self-averaging effect of variance reduction that is so prominent when boosting high-variance base learners. If this view is correct, one may have to rethink the role of the surrogate loss function that is minimized by boosting. Its main role is to produce structured weights in the iterations, but with noisy errors, these weights may for practical purposes be as much random as they are systematic. This insight jibes with Wyner's (2002) malicious experiments in which he doubled the step size of discrete AdaBoost with C4.5, thereby assuring that the exponential loss never decreased and in fact provably remained at a constant level; his empirical results indicated that on average this SOR (successively over-relaxed) form of boosting performs as well as regular AdaBoost. These results may be taken as evidence that the minimization aspect is of little importance for a high-variance base learner; of greater importance may be a pseudo-random aspect of the reweighting scheme that achieves variance reduction similar to bagging, just more successfully due to a sort of adaptivity in the reweighting that improves over the purely random resampling of bagging.

If the pseudo-random aspect of boosting is critical for high-variance base learners, one may draw consequences and implement boosting with proper pseudo-random processes. So did Breiman. He didn't attempt a theory of boosting for high-variance base learners, and instead he put his intuitions to use in further proposals such as in his work on "half & half bagging" (Breiman, 1998a), apparently with success. Another example that benefited from Breiman's inspiration was Friedman's (2002) "stochastic gradient boosting" which inhibits convergence of boosting by computing gradient steps from random subsamples drawn without replacement. Friedman (ibid., p. 9) observes improvements over deterministic boosting in a majority of situations, above all for small samples and "high capacity" (high variance) base learners. He admits that "the reason why this randomization produces improvement is not clear," but suggests "that variance reduction is an important ingredient." Friedman goes on to suggest that stochastic AdaBoost with sampling from the weights rather than reweighting may have similar variance-reducing effects. In early boosting approaches such sampling (with replacement) was performed to match the given sample size, but Friedman suggests that further variance reduction could be gained by choosing smaller resamples.

An implication of Breiman's hunch is that the real difference between LogitBoost and AdaBoost is not so much due to the differences in loss functions as to the minimization method, at least when the base learner has relatively high variance, or generally in the late stages of boosting. AdaBoost can be interpreted as constrained gradient descent on the exponential loss, whereas LogitBoost is Newton descent on the logistic loss (Friedman et al., 2000). The two minimization schemes produce very different reweighting schemes, and they work off different working responses during the iterations. We are currently ignorant about whether LogitBoost develops pseudo-random behavior late in the iterations, similar to AdaBoost. If it does, the cause may be traced to the base learner, and the phenomenon may be robust to the specifics not only of loss functions but of algorithms as well.

Another implication of Breiman's hunch is that boosting does both, reduce bias and variance, in the same problem, but each primarily at different stages of the boosting iterations. If it is true that variance reduction occurs during later iterations, then this should go a long way to explain boosting's relative immunity to overfitting. By comparison, conventional fitting mechanisms only know how to do one thing: follow the data ever more closely, thereby continually reduce bias and continually

accumulate variance. According to orthodoxy, therefore, the art is to find the proper balance, and to this end auxiliary devices such as early stopping, regularization penalties and cross-validation come into play. Boosting seems to be different, but we do not have the theory yet to prove it.

All that we said so far is based on out-of-sample classification error. A peculiarity of classification error is that it is not the criterion being minimized in-sample because of its discontinuous nature. The role of minimizing a smooth surrogate loss function is to trace a path that leads to low classification error, but the surrogate loss is not of interest in itself. Yet, for the variance reducing properties of the resulting classifier, the surrogate loss is of interest. First of all, the surrogate loss should keep decreasing because for example discrete AdaBoost is constrained gradient descent with line search (Friedman et al., 2000). This explains why in terms of the surrogate loss, the fitted class probability estimates end up vastly overfitting the data, confirming the orthodox view in terms of the surrogate loss. Yet, two phenomena are also observed: in terms of out-of-sample classification error, no overfitting is taking place, and, according to Breiman, no convergence of the weights is taking place. The bouncing of the weights would indicate that, in spite of a well-behaved convex loss function, the descent directions chosen by the base learner become erratic. Such behavior would be plausible if the base learner is of the high-variance type, but the specifics of why the variance component of out-of-sample classification error is improved is not explained. It is quite clear, though, that explaining boosting's variance reduction would be a greater achievement than explaining its bias reduction. Bias can be largely taken care of by the base learner, variance can't.

If the peculiarities we observe in boosting are due to the use of two loss functions, one may ask whether any lessons learned carry over to other parts of statistics. The "statistical view" has indeed produced generalizations of boosting to other areas, such as regression: Bühlmann and Yu's L_2-boosting (2003), Friedman's gradient boosting in their deterministic and stochastic forms (2001; 2002), and boosting of exponential and survival models by Ridgeway (1999). In these single-loss function contexts, the paradoxical phenomena should no longer be visible, as they aren't for boosting if judged in terms of the surrogate loss function. Yet, Friedman's stochastic gradient boosting shows that adding an element of variance reduction with randomization may just be what the doctor ordered in most statistical model fitting contexts even with a single loss function. We should therefore aim for a variance reduction theory for all of statistics, reaching beyond classification.

Another question that may be raised for binary, or categorical response data in general, is whether classification error is as desirable a loss function as suggested by the attention it has received. Classification error is a bottom line number that may be appropriate in industrial contexts where real large scale engineering problems are solved, for example, in document retrieval. One might characterize these contexts as "the machine learner's black box problems." There do exist other contexts, though, and one might characterize them as "the problems of the interpreting statistician." When interpretation is the problem, attaining the last percent of classification accuracy is not the goal. Instead, one hopes to develop a functional form that reasonably fits the data but also "speaks," that is, lends itself to statements about what variables are associated with the categorical response. Fitting good conditional class probabilities takes on greater importance because associations and effects can then be measured in terms of differences in the logits of class 1 (for example) for a unit or percentage difference in the predictor variables. Interpretability is a problem for nonparametric model fits such as boosted trees. The decomposition of complex fits into interpretable components, for example with an ANOVA decomposition as suggested by Friedman et al. (2000), takes on considerable importance. In the end, one may want to produce a few telling plots explaining functional form and a few numbers summarizing the strengths of various associations. When

fitting models for conditional class probabilities, the surrogate loss becomes the primary loss function because it can be interpreted as a loss function for fitted class probabilities. It is one of the achievements of Friedman et al. (2000) to have shown that this is true for exponential loss as much as for logistic loss, even though there is a misperception, as pointed out by Mease and Wyner, that LogitBoost was specifically designed to recover class probabilities that AdaBoost couldn't. Exponential loss does similar things as logistic loss in Friedman et al.'s analysis, and they provide the appropriate link functions for both. All this is relevant only if minimization of the "surrogate/now-primary loss" is prevented from overfitting, with cross-validated early stopping, penalization, or variance-reducing randomization, and it comes at the cost of diminished classification performance, one of Mease and Wyner's points.

Diminished classification performance when estimating class probabilities is easily explained; it is due to a compromise that class probability estimation has to strike. It effectively attempts good classification simultaneously at *all* misclassification cost ratios. (Note that this ratio is assumed to be one in most of the boosting literature.) This statement can be made precise in a technical sense: unbiased loss functions for class probabilities, so-called "proper scoring rules," are weighted mixtures of cost-weighted misclassification losses Buja et al. (2005). After mapping exponential and logistic losses to probability scales with their associated inverse link functions, they turn into proper scoring rules and therefore exhibit the mixture structure just described. It follows that both loss functions attempt compromises across classification problems with non-equal misclassification costs. Both loss functions give inordinate attention to extreme cost ratios, but exponential loss even more so than logistic loss. At any rate, the nature of the compromise is such that no cost ratio, in particular not equal costs, is served optimally if the exponential or logistic losses are tuned to high out-of-sample performance. By comparison, overfitting these losses in-sample seems to provide benefits in terms of classification error. However, once we change our priorities from black-box performance to interpretation, and hence from classification to class probability estimation, we may prefer tuning surrogate loss and accept the increased classification error.

Anticipating an objection by Mease and Wyner, we should disclose that one of us (Buja) collaborated with them on an article that is relevant here (Mease et al., 2007). In this work we traveled the opposite of the usual direction by composing class probability estimates from layered classification regions, estimated at a grid of misclassification cost ratios. Presumably such class probability estimation inherits superior performance from boosting in classification. When interpretation is the goal, however, a simple functional form that "speaks" might be more desirable than the increased performance of the layered estimates we provide in our joint proposal. The problem is that our proposal inherits the interpretative disadvantages of boosted classification regions, which tend to be jagged around the edges and pockmarked with holes—not a credible feature when it comes to interpretation.

We started this discussion joining Mease and Wyner in their argument against today's boosting orthodoxy. We ended by questioning the single-minded reliance on classification error as the only yard stick of performance. Still, Mease and Wyner's call should be heard because the orthodoxy misattributes the causes of boosting's success and makes invalid recommendations.

References

Y. Amit, G. Blanchard, and K. Wilder. Multiple randomized classifiers: Mrcl. Technical report, University of Chicago, 2001.

L. Breiman. Half & half bagging and hard boundary points. Technical Report 534, Statistics Dept., Univ. of California, Berkeley, 1998a.

L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998b.

L. Breiman. Random forests—random features. Technical Report 567, Statistics Dept., Univ. of California, Berkeley, 1999.

P. Bühlmann and B. Yu. Analyzing bagging. *The Annals of Statistics*, 30(4):927–961, 2002.

Peter Bühlmann and Bin Yu. Boosting with the L_2 loss: Regression and classification. *Journal of the American Statistical Association*, 98:324–339, 2003.

A. Buja, W. Stuetzle, and Y. Shen. Loss functions for binary class probability estimation and classification: Structure and applications. Technical report, The Wharton School, University of Pennsylvania, 2005.

J. Friedman. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

J. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38:367–378, 2002.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

D. Mease, A. Wyner, and A. Buja. Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research*, 8:409–439, 2007.

G. Ridgeway. The state of boosting. *Computing Science and Statistics*, 31:172–181, 1999.

R. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.

A. Wyner. Boosting and the exponential loss. In *Proceedings of the Ninth Annual Conference on AI and Statistics*, 2002.

Journal of Machine Learning Research 9 (2008) 171-174 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008

Yoav Freund YFREUND@UCSD.EDU
Department of Computer Science and Engineering
University of California
San Diego, CA 92093

Robert E. Schapire SCHAPIRE@PRINCETON.EDU
Princeton University
Department of Computer Science
35 Olden Street
Princeton, NJ 08540

Editor: Yoav Freund

For such a simple algorithm, it is fascinating and remarkable what a rich diversity of interpretations, views, perspectives and explanations have emerged of AdaBoost. Originally, AdaBoost was proposed as a "boosting" algorithm in the technical sense of the word: given access to "weak" classifiers, just slightly better in performance than random guessing, and given sufficient data, a true boosting algorithm can provably produce a combined classifier with nearly perfect accuracy (Freund and Schapire, 1997). AdaBoost has this property, but it also has been shown to be deeply connected with a surprising range of other topics, such as game theory, on-line learning, linear programming, logistic regression and maximum entropy (Breiman, 1999; Collins et al., 2002; Demiriz et al., 2002; Freund and Schapire, 1996, 1997; Kivinen and Warmuth, 1999; Lebanon and Lafferty, 2002). As we discuss further below, AdaBoost can been seen as a method for maximizing the "margins" or confidences of the predictions made by its generated classifier (Schapire et al., 1998). The current paper by Mease and Wyner, of course, focuses on another perspective, the so-called statistical view of boosting. This interpretation, particularly as expounded by Friedman et al. (2000), focuses on the algorithm as a stagewise procedure for minimizing the exponential loss function, which is related to the loss minimized in logistic regression, and whose minimization can be viewed, in a certain sense, as providing estimates of the conditional probability of the label.

Taken together, these myriad interpretations of AdaBoost form a robust theory of the algorithm that provides understanding from an extraordinary range of points of view in which each perspective tells us something unique about the algorithm. The statistical view, for instance, has been of tremendous value, allowing for the practical conversion of AdaBoost's predictions into conditional probabilities, as well as the algorithm's generalization and extension to many other loss functions and learning problems.

Still, each perspective has its weaknesses, which are important to identify to keep our theory in touch with reality. The current paper is superb in exposing empirical phenomena that are apparently difficult to understand according to the statistical view. From a theoretical perspective, the statistical interpretation has other weaknesses. As discussed by Mease and Wyner, this interpretation does not explain AdaBoost's observed tendency not to overfit, particularly in the absence of regularization

or early stopping. It also says little about how well AdaBoost will generalize when provided with a finite data set, nor how its ability to generalize is dependent on the complexity or simplicity of the base classifiers, an issue that arises in the experiments comparing decision stumps and decision trees in this role.

Much of the difficulty arises from the fact that AdaBoost is a *classification* algorithm (at least as it is used and studied in the current paper). This means that AdaBoost's purpose is to find a rule h that, given X, predicts one of the labels $h(X)$, and that attempts to achieve minimal probability of an incorrect classification (in which $h(X)$ disagrees with the true label Y). This is quite different from the problem of estimating the conditional probability $P(Y|X)$. An accurate estimate of this conditional probability is a sufficient, but certainly not a necessary, condition for minimizing the classification error. A weaker requirement that is still sufficient is to estimate the set of inputs for which $P(Y = +1|X) > 1/2$. In most cases, this requirement is much weaker than the requirement of getting good estimates of conditional probabilities. For example, if $P(Y = +1|X) = 0.49$ then our estimate of the conditional probability need be accurate to within 1%, while if $P(Y = +1|X) = 0.2$ the accuracy we need is only 30%.

This simple observation demonstrates a crucial shortcoming in the statistical interpretation of Adaboost, and undermines many of its apparent consequences, including the following:

- *Adaboost can be interpreted as a method for maximizing conditional likelihood.* If the goal is not to estimate the conditional probability, there is no reason to maximize likelihood.

- *A question of central importance is whether Adaboost is asymptotically consistent.* When evaluating probability estimators, it is standard procedure to start by verifying that the estimator is unbiased. Once the estimator is confirmed to be unbiased, the next question is the rate at which its variance decreases with the size of the sample. Again, as the learning problem in the case of classification is a weaker one, it is not clear that this is the relevant sequence of questions that a theoretician should ask.

- *Decision stumps should be used as base classifiers when the input variables are independent* This argument is based on the assumption that the goal is to estimate probabilities.

The view of AdaBoost as a method for minimizing exponential loss, though in some ways quite useful, can also lead us very much astray, as pointed out to some degree by Mease and Wyner. Taken to an extreme, this view suggests that any method for minimizing exponential loss will be equally effective, and is likely to be much better if designed with speed and this explicit goal in mind. However, this is quite false. Indeed, any real-valued classifier F which classifies the training examples perfectly, so that $y_i F(x_i) > 0$ for each training example (x_i, y_i), can be modified to minimize the exponential loss $\sum_i e^{-y_i F(x_i)}$ simply by multiplying F by an arbitrarily large positive constant. This scaling of F of course has no impact on the classifications that it makes. Thus, in the common case in which an exponential loss of zero is possible, minimization of this loss means nothing more than that the computed classifier F has a classification error of zero on the training set. The minimization of this particular loss tells us nothing more, and leaves us as open to overfitting as any other method whose only purpose is minimization of the training error.

This means that, in order to understand AdaBoost, which does indeed minimize exponential loss, we need to go well beyond this narrow view. In particular, we need to consider the *dynamics* of AdaBoost—not just *what* it is minimizing, but *how* it goes about doing it.

Like other interpretations of AdaBoost, although the statistical view has its weaknesses, it also has its strengths, as noted above. Still, to fully understand AdaBoost, particularly in the face of such deficiencies, it seems unavoidable that we consider a range of explanations and modes of understanding. Where the statistical view may be lacking, the margins explanation in particular can often shed considerable light.

Briefly, the *margin* of a labeled example with respect to a classifier is a real number that intuitively measures the confidence of the classifier in its prediction on that example. More precisely, in the notation of Mease and Wyner, the margin on labeled example (x,y) is defined to be $yF_M(x)/\sum_m \alpha_m$. Equivalently, viewing the prediction of AdaBoost's combined classifier as a weighted majority vote of the base classifiers, the margin is the weighted fraction of base classifiers voting for the correct label minus the weighted fraction voting for the incorrect label.

The margins theory (Schapire et al., 1998) provides a complete analysis of AdaBoost in two parts: First, AdaBoost's generalization error can be bounded in terms of the distribution of margins of training examples, as well as the number of training examples and the complexity of the base classifiers. And second, it can be proved that AdaBoost's dynamics have a strong tendency to increase the margins of the training examples in a manner that depends on the accuracy of the base classifiers on the distributions on which they are trained.

This theory is quite useful for understanding AdaBoost in many ways (despite a few shortcomings of its own—see, for instance, Breiman (1999) as well as the recent work of Reyzin and Schapire (2006)). For starters, the theory, in which performance depends on margins rather than the number of rounds of boosting, predicts the same lack of overfitting commonly observed in practice. The theory provides non-asymptotic bounds which, although usually too loose for practical purposes, nevertheless illuminate qualitatively how the generalization error depends on the number of training examples, the margins, and the accuracy and complexity of the base classifiers. Finally, the theory is concerned directly with classification accuracy, rather than the algorithm's ability to estimate conditional probabilities, which is in fact entirely irrelevant to the theory.

Moreover, some of the phenomena observed by Mease and Wyner do not appear so mysterious when viewed in terms of the margins theory. For instance, the experiments in Section 3.1 show AdaBoost overfitting with stumps but not decision trees. In terms of margins, decision trees have higher complexity, which tends to hurt generalization, but also tend to produce much larger margins, which tend to improve generalization, an effect that can easily be strong enough to compensate for the increased complexity. Moreover, according to the theory, these larger margins tend to provide immunity against overfitting, and indeed, overfitting is expected exactly in the case that we are using base classifiers producing small margins, such as decision stumps. This is just what is observed in Figure 1.

In sum, the various theories of boosting, including the margins theory and the statistical view, are all imperfect but are largely complementary, each with its strengths and weaknesses, and each providing another piece of the AdaBoost puzzle. It is when they are taken together that we have the most complete picture of the algorithm, and the best chances of understanding, generalizing and improving it.

References

Leo Breiman. Prediction games and arcing classifiers. *Neural Computation*, 11(7):1493–1517, 1999.

Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48(1/2/3), 2002.

Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1/2/3):225–254, 2002.

Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 325–332, 1996.

Yoav Freund and Robert 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.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, April 2000.

Jyrki Kivinen and Manfred K. Warmuth. Boosting as entropy projection. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 134–144, 1999.

Guy Lebanon and John Lafferty. Boosting and maximum likelihood for exponential models. In *Advances in Neural Information Processing Systems 14*, 2002.

Lev Reyzin and Robert E. Schapire. How boosting the margin can also boost classifier complexity. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, October 1998.

Journal of Machine Learning Research 9 (2008) 175-180 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008

Jerome Friedman JHF@STANFORD.EDU
Trevor Hastie HASTIE@STANFORD.EDU
Robert Tibshirani TIBS@STANFORD.EDU
Department of Statistics
Stanford University
Stanford, CA 94305

Editor: Yoav Freund

1. Introduction

This is an interesting and thought-provoking paper. We especially appreciate the fact that the authors have supplied R code for their examples, as this allows the reader to understand and assess their ideas. The paper inspired us to re-visit many of these issues underlying boosting methods. However in the end we do not believe that the examples provided in the paper contradict our statistical view, although other views may well prove informative.

2. Our Statistical View of Boosting

Friedman et al. (2000) and our book (Hastie et al., 2001) argue that boosting methods have three important properties that contribute to their success:

1. they fit an additive model in a flexible set of basis functions
2. they use a suitable loss function for the fitting process
3. they regularize by forward stagewise fitting; with shrinkage this mimics an L_1 (lasso) penalty on the weights.

In many cases the paper ascribes consequences of this statistical view that are not the case. For example, it does not follow that smaller trees are necessarily better than larger ones for noisier problems (Sections 3.2 and 4.2), that the basis should necessarily be restricted as described in Sections 3.6 and 4.6, or that regularization should be based on the loss function used for fitting (Sections 3.5 and 4.5). To the extent possible model selection should be based on the ultimate loss associated with the application. Also, there is no requirement that test error have a unique minimum as a function of the number of included terms (Sections 3.4 and 4.4). However, to the extent that these are commonly held beliefs, the paper provides a valuable service by pointing out that they need not hold in all applications.

There is no direct relation between the application of shrinkage and overfitting (Sections 3.7 and 4.7). Heavy shrinkage emulates L_1 regularization, whereas its absence corresponds to stagewise

fitting approximating L_0 regularization. There is nothing in the statistical view that requires L_1 to be superior to L_0 in every application, although this is often the case. The best regularizer depends on the problem: namely the nature of the true target function, the particular basis used, signal-to-noise ratio, and sample size.

Finally, there is nothing in our statistical interpretation suggesting that boosting is similar to one nearest neighbor classification (Sections 3.9 and 4.9).

None-the-less, the paper does provide some interesting examples that appear to contradict the statistical interpretation. However these examples may have been carefully chosen, and the effects seems to vanish under various perturbations of the problem.

3. Can the "Wrong" Basis Work Better than the Right One?

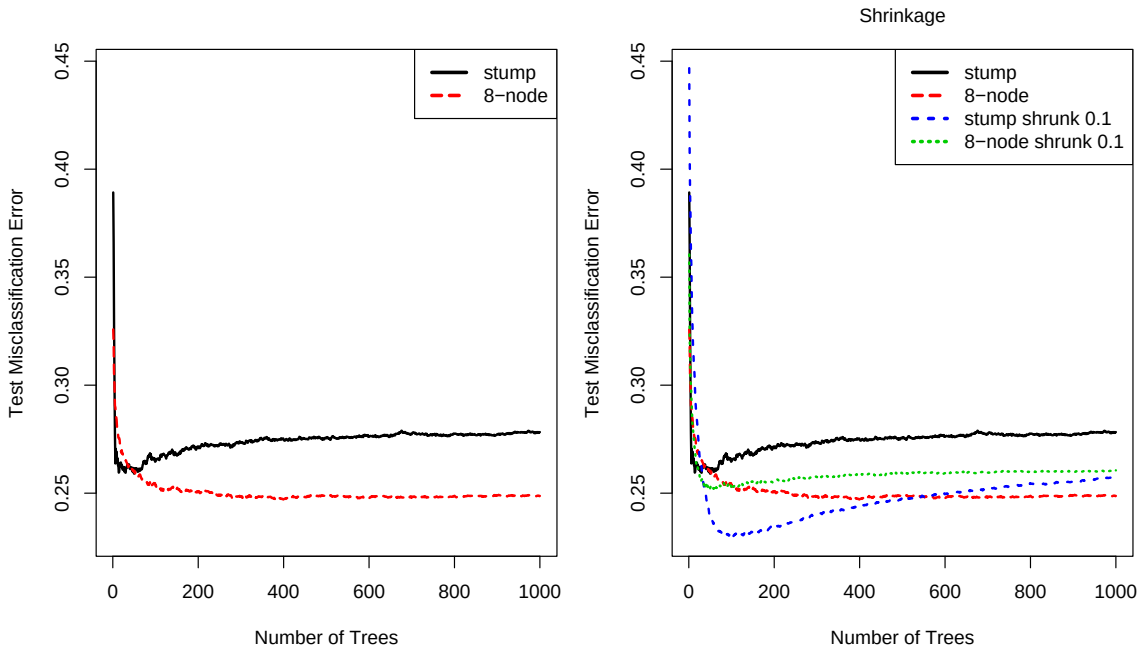


Figure 1: Average test misclassification error for 20 replications of Mease and Wyner's example used in their Figure 1. We used the package GBM in R, with the "adaboost" option. The left panel shows that 8-node trees outperform stumps. The right panel shows that stumps with shrinkage win handily.

The left panel of Figure 1 shows a version of the paper's Figure 1. We see that boosting with 8 node trees seems to outperform stumps, despite the fact that the generative model is additive in the predictor variables. However the right panel shows what happens to both stumps and 8 nodes trees when shrinkage is applied. Here shrinkage helps in both cases, and we see that stumps with shrinkage work the best of all.

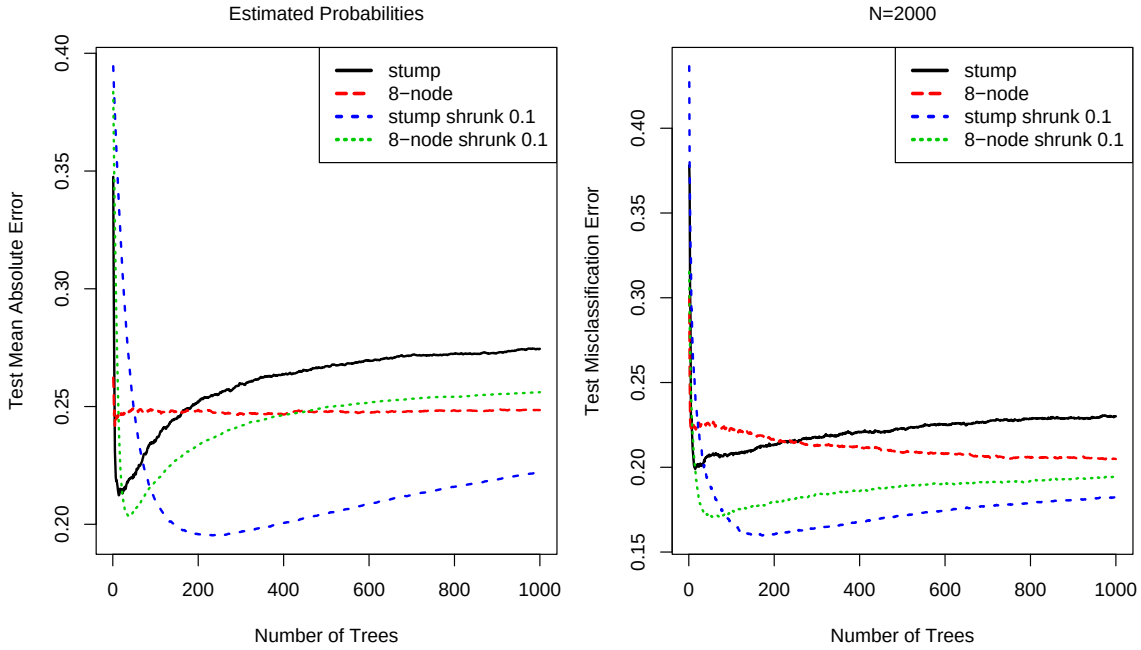


Figure 2: Left panel: average absolute deviations of the fitted probabilities from the true probabilities for the same simulations as in Figure 1. Right panel: average test misclassification error for the same simulations as in Figure 1, except using 2000 rather than 200 training examples.

We are not sure why unshrunken 8 node trees outperform unshrunken stumps in this example. As in the paper, we speculate that the extra splits in the 8 node tree might act as a type of regularizer, and hence they help avoid the overfitting displayed by unshrunken stumps in this example. All but the first split will tend to be noisy attempts at the other variables, which when averaged will have a "bagging" effect.

However this explanation becomes less convincing and indeed the effect itself seems to fade when we look more deeply. Figure 2 [left panel] shows the average absolute error in the estimated probabilities, while Figure 2[right panel] shows what happens when we increase the sample size to 2000. In Figure 3[left panel] we use the Bernoulli loss rather than exponential of Adaboost, and Figure 3[right panel] shows results for the regression version of this problem. In every case, the effect noted by the authors goes away and both the correct bases and shrinkage help performance. We repeated these runs on the second simulation example of Section 4, and the results were similar. Thus the effect illustrated by the authors is hard to explain, and seems to hold only for misclassification error. It depends on a very carefully chosen set of circumstances. Most importantly, we have to remember the big picture. Looking at the right panel of Figure 1, which method would anyone choose? Clearly, shrunken stumps work best here, just as might be expected from the statistical view.

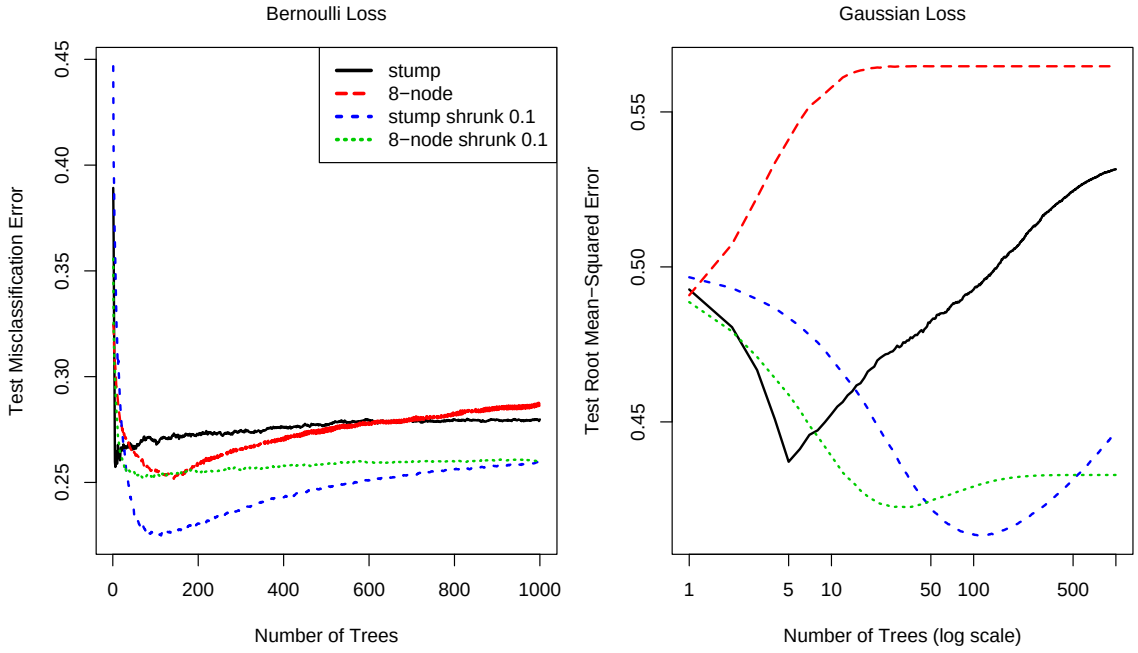


Figure 3: Left panel: test misclassification error when boosting with Bernoulli loss for the same simulations as in Figure 1. Right panel: root mean-squared test error when boosting with squared-error loss for the same simulations as in Figure 1 (legend as in left panel).

Figure 4 shows the fitted probabilities over the 20 runs, separately for each class, when using 250 shrunken stumps. Here 250 was chosen since it corresponds to the minimum in Figure 2[left panel]. This is an appropriate tradeoff curve if we are interested in probabilities; test deviance would also be fine. We see that the estimates are biased toward 0.5, which is expected when regularization is used. Hence they are underfit, rather than overfit.

A similar argument can be made concerning the paper's Figure 3. Yes, AdaBoost works better than Logitboost in this example. But using the statistical view of boosting, we have moved on and developed better methods like gradient boosting (Friedman, 2001) that typically outperform both of these methods.

Hastie et al. (2007) add further support to (3) of the statistical interpretation of boosting: they show that the incremental forward stagewise procedure used in boosting (with shrinkage) optimizes a criterion similar to but smoother than the L_1 penalized loss.

4. Conclusion

No theory, at least initially, can fully explain every observed phenomenon. Everything about regularized regression is not yet fully understood. There is still considerable ongoing research in the literature concerning the interplay between the target function, basis used, and regularization method. Hopefully, some of the apparent anomalies illustrated in this paper will eventually be explained with

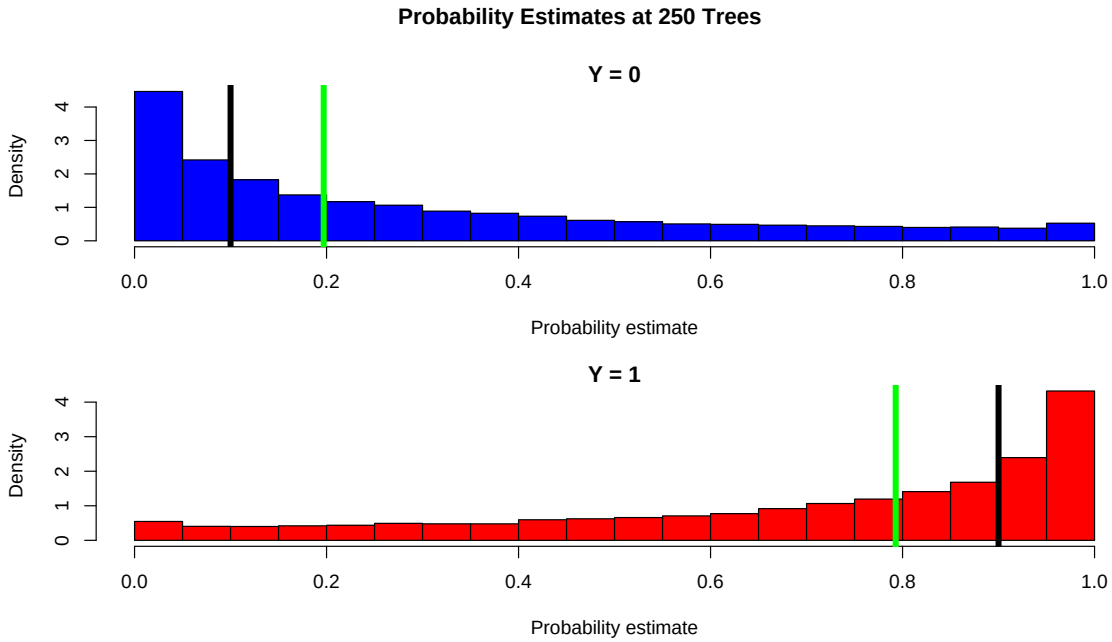


Figure 4: Fitted probabilities shown for the two classes, at 250 shrunken stumps. The vertical black bars are the target probabilities for this problem, and the green bars are the median of the estimates in each class.

a more thorough understanding of these issues. The paper provides a service in reminding us that there is still work remaining.

Although we would not begin to suggest that our statistical view of boosting has anywhere near the substance or importance of the Darwin's theory of evolution, the latter provides a useful analogy. The proponents of Intelligent Design point out that the theory of evolution does not seem to explain certain observed biological phenomena. And therefore they argue that evolution must be wrong despite the fact that it *does* explain an overwhelming majority of observed phenomena, and without offering an alternative *testable* theory.

We are sure that the authors will mount counter-arguments to our remarks, and due to the (standard format) of this discussion, they will have the last word. We look forward to *constructive* counter-arguments and alternative explanations for the success of boosting methods that can be used to extend their application and produce methods that perform better in practice (as in the right panel of Figure 1).

References

J. Friedman. Greedy function approximation: The gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting (with discussion). *Annals of Statistics*, 28:337–407, 2000.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning; Data Mining, Inference and Prediction*. Springer Verlag, New York, 2001.

T. Hastie, J. Taylor, R. Tibshirani, and G. Walther. Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, 1:1–29 (electronic), 2007. DOI: 10.1214/07-EJS004.

Journal of Machine Learning Research 9 (2008) 181-186 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008**: And Yet It Overfits**

Peter J. Bickel BICKEL@STAT.BERKELEY.EDU
Department of Statistics
University of California
Berkeley, CA 94720-3860, USA

Ya'acov Ritov YAACOV.RITOV@GMAIL.COM
Department of Statistics
The Hebrew University of Jerusalem
91905 Jerusalem, Israel

Editor: Yoav Freund

Galileo: God help us, I'm not half as sharp as those gentlemen in the philosophy department. I'm stupid, I understand absolutely nothing, so I'm compelled to fill the gaps in my knowledge... Sir, my branch of knowledge is still avid to know. The greatest problems still find us with nothing but hypotheses to go on. Yet we keep asking ourselves for proofs. Brecht (1980)

This is a "sock it to them" paper—though much less so than the previous versions one of us have seen.

The authors argue in the paper that AdaBoost (without early stopping) is one of, if not the, most successful boosting algorithms, and they present this paper as a disproof of what the , rather amorphous community of statistical practitioners, represented by Friedman, Hastie and Tibshirani have:

(i) Pointed out as remediable flaws of the original Freund-Schapire boosting algorithm;

(ii) Given as remedies.

Evidently, that community should be able to respond on its own. We in fact, agree with some of the hypotheses Mease and Wyner's limited simulations lead them to, whether these are or are not embraced by statistical practitioners. But others we find dubious and unproven. Let us stress the positive first.

1. They argue that boosting does not behave like nearest neighbor for $d > 1$. Not only do we agree with this but would conjecture even further without any proof:

2. That, for reasonable sequences of d dimensional distributions, the random classification rules induced by the stationary measures corresponding to boosting forever, should in a suitable sense as $n, d \to \infty$ concentrate near the Bayes rule. However, an example below shows that the improvement from $d = 1$ to $d = 2$ can be slight.

3. We don't believe that boosting is consistent, in the sense of section 3.10, for any d, but indeed there is no disproof for $d > 1$.

4. We agree that a sharp explanation why, for classification, boosting *may* not overfit—that is, continues to reduce the probability of misclassification long past the point where all training sample observations are correctly classified has not been provided in the statistical (or the machine learning) literature.

5. We agree that using more complex basis functions may actually improve performance. This was analyzed theoretically for L_2 boosting by Bühlmann and Yu (2003).

6. We agree that there is a need for a convincing argument for basing an early-stopping algorithm of a classifier on a loss function that is not the classification loss. *A-priori* we do not expect that stopping on any criterion, other than minimum classification error will work in general, even if the classifier itself is based on minimizing this indirectly relevant criterion. However, it certainly can be that a good early stopping algorithm will be based on estimate of the loss with respect to something other than from the classification error. It was proven to be so, for example, with L_2-boost.

Since we have never been persuaded on theoretical grounds of the superiority of other "boosts", logit or L_2, over AdaBoost , we leave this battle to others.

Where we really part company with the authors of this "against the heretics" paper is on the issue of the desirability of early stopping.

Galileo tries to explain his young student, Andrea, the structure of the Copernican system, to make it so simple that Andrea will be able to explain it to his mother. He rotates him on a chair, and tells him that an apple in the center is the earth. However, Andrea is smart enough to understand that so far and not more can be deduced from examples.
Andrea: Examples always work if you're clever. Only I can't lug my mother round in a chair like you did me. So you see it's a rotten example really. And suppose your apple is the earth like you say? Nothing follows. Brecht (1980)

Everybody can produce examples. The authors gave two examples. Other commentators will bring their own. Here are ours. We consider $X \in \mathbb{R}^2$ uniform on 5 concentric circles. The classes were randomly assigned according to $P(Y=1|x) = \text{logit}(4\text{sgn}\,(\xi)\sqrt{(}\xi))$ where $\xi = \|x\| - 2|x_1|$ and $\text{logit}(\zeta) = e^{\zeta}/(1+e^{\zeta})$. The training sample includes 500 i.i.d. observations. 200 more observations were used for early stopping. I.e., stopping when the mean empirical classification error on these 200 observations was minimal. Finally another set of 1500 observations was used to evaluate the mean classification error of the AdaBoost procedure as a function of the number of observations. See Figure 1 for a plot of typical set of Xs and $P(Y=1|X))$. The weak classifier we used was a standard classification tree with 8-terminal nodes. The simulations were run with slightly different set-ups a few tens of times. There was no case that contradicted the results of the single experiment we will present next (but see later). Our example is small. Since the goal of the discussed paper is to suggest policy on the basis of two examples, even one (we believe) reasonable counterexample should give some pause.

In Figure 2(a) we show the classification error as a function of the number of iterations. The horizontal lines are, from the top down: The risk of the closest neighbor, the risk of AdaBoost with early stopping, and the Bayes Risk. It is clear in this example that AdaBoost starts quite nicely. The primitive early stopping technique we employed is enough to give a decent performance. However, the performance of AdaBoost starts to degenerate after some tens of iterations. After 800 hundred

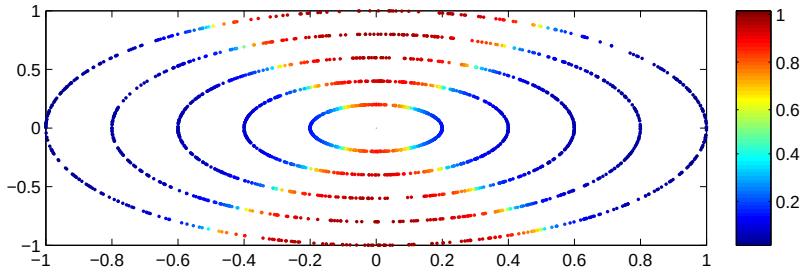


Figure 1: $P(Y=1|X)$ as function of location.

iterations it is only slightly better than the nearest neighbor classifier. Within the context of this example the "does not over-fit" property, can be understood at best as "it is simply a slow algorithm".

Figure 2(b) shows the root mean square error of AdaBoost implicit estimate of the probability as function of the number of iterations. It is clear that it degenerate much faster than AdaBoost performance. However, the implicit probability estimate is fair, as long as the classifier is in its prime. So, the authors' doubt whether boosting estimate probabilities cannot be based on this example.

However, we should mention that the apparent failure of the boosting algorithm to estimate probabilities is somewhat misleading. In Figure 2(c) we plot $F_m - F^0$, where F^0 is the ideal value, as a function of $P(Y=1|x)$ after $m=200$ iterations. $P(Y=1|x)$ is used here as proxy for the distance of the point from the $P(Y=1|x)=0.5$ boundary. What can be seen is that most of the error in the estimation of the $P(Y=1|x)$ comes from the easy to classify points. So, the boosting algorithm fails to estimate the probability where it does not really matter.

There are facts in life. One cannot invoke the church teaching or Aristotle's books in face of empirical facts. Galileo get annoyed by the insistence of the philosopher and the mathematician on using irrelevant arguments.

Galileo: My reasons! When a single glance at the stars themselves and my own notes make the phenomenon evident? Sir, your disputation becoming absurd. Brecht (1980)

The authors used a smart device to present the discrepancy between the one nearest neighbor classifier and AdaBoost. Namely, they compare their performance under a null hypothesis, where $P(Y=1|x)$ does not depend on x (this is true for Section 3.9 but not for 4.9, where for some unknown reasons something else was done). We did the same. The distribution of X and the sample sizes are as above, while $P(Y=1|x)\equiv 0.2$. The results are presented in Figure 3. The graphs are similar to Figure 2(a-b), in description and in essence. Boosting without early stopping is not like 1-NN, but it is not much better, at least for this tiny example.

We did observe a further interesting phenomenon when we added irrelevant explanatory variables. That is, 8 independent variables $X_3,\ldots,X_{10}$ were added, while the distribution of (X_1,X_2,Y) remains as above. The result was surprising. Adding these irrelevant variables improved the performance of "boosting-for-ever" to the level of the early stopping algorithm. This gives some credence to our conjecture 2.

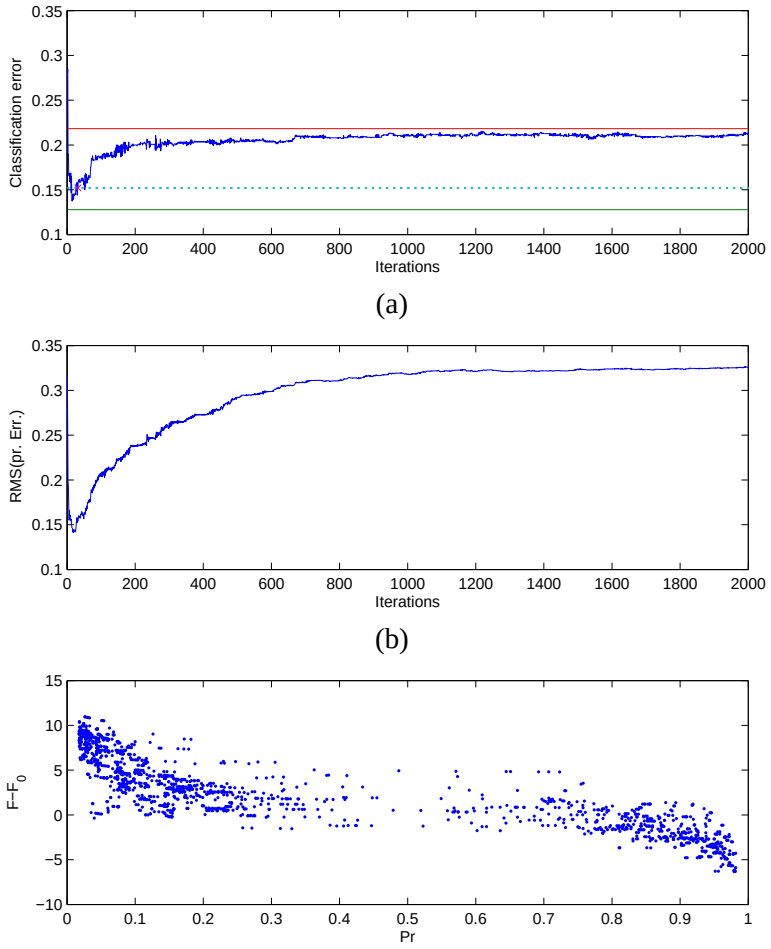
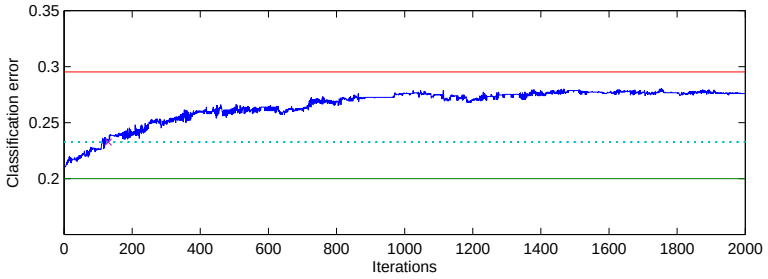
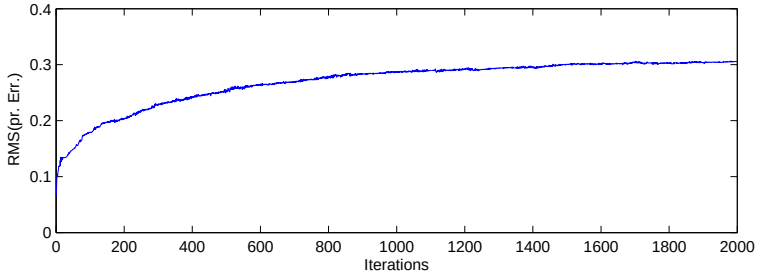


Figure 2: AdaBoost, with and without early stopping.



(a)



(b)

Figure 3: AdaBoost under the null hypothesis.

But, this is not the end of the story. We changed the the distribution a little. We left $P(Y|X_1,\ldots,X_{10}) = P(Y|X_1,X_2)$ as above. $X_3,\ldots,X_{10}$ are still independent of each other and independent of (X_1,X_2). However the distribution of (X_1,X_2) was changed to be discrete with 24 well isolated atoms. One can conceive the distribution as that of 24 columns with $500/24$ on the average observations. The Ys are i.i.d. given the column. AdaBoost with early stopping handled this situation very well and stopped after very few (e.g., 2) iterations. It essentially isolated the columns and left them intake. Boosting-for-ever stabilized nicely after very few iterations. However, as could be expected, it did break the columns. The result was that the misclassification error of boosting-for-ever was almost in the middle between the early stopping algorithm, and the strictly inferior 1-NN estimator.

A further point. The phenomenon of not overfitting for a long time is certainly interesting to investigate, but why this should be a virtue of a procedure is unclear, since it merely increases computation time at an often (perhaps usually) negligible improvement over stopping early.

AdaBoost is a mystery, but we, the weak, can solve only one toy problem at a time.
Galileo: Why try to be so clever now, that we at last have a chance of being less stupid? Brecht (1980)

AdaBoost was crowned by Leo Breiman as the best off-the-shelf classifier. It has some mysterious properties, particularly, sometimes continuing to improve off-sample performance even after completely collapsing on the data. It behaves better, sometimes, with 8-nodes trees, some other times with 256-node trees, and other times, mainly when examined by some statisticians, stumps are superior. It presents some mathematical challenges, which should be carefully investigated. However, many examples appearing in the literature are either very artificial, or the investigators

don't have a gold standard like the Bayes risk. We hope that this technique will grow out of its status of something like an art, to a scientifically justified method. But this is only a hope.

Acknowledgments

This work was supported in part by NSF grant DMS-0605236 and an ISF grant.

References

Bertolt Brecht. *Life of Galileo*. Arcade Publishing, New York, 1980. Translated by John Willet.

Peter Bühlmann and Bin Yu. Boosting with the l2 loss: Regression and classification. *Journal of the American Statistical Association*, 98:324–339, 2003.

Journal of Machine Learning Research 9 (2008) 187-194 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008

Peter Bühlmann BUHLMANN@STAT.MATH.ETHZ.CH
Seminar für Statistik, LEO C17
ETH Zurich
CH-8092 Zurich
Switzerland

Bin Yu BINYU@STAT.BERKELEY.EDU
Department of Statistics
367 Evans Hall #3860
Berkeley, CA 94720
USA

Editor: Yoav Freund

We would like to thank the authors for their provocative view on boosting. Their view is built upon some "contrary" evidence based on a particular simulation model. In our discussion, we argue that the structure of the simulation model explains many aspects of the "contrary" evidence. We touch upon the issue of shrinkage or small step-sizes, and we conclude that the "statistical view" provides constructive insights for applying boosting in a highly successful way.

The gradient and "statistical" point of view The gradient point of view of AdaBoost is, in our opinion, a great leap forward for understanding AdaBoost and deriving new variants of boosting now meaning much more than just AdaBoost. This view, which seems to be called the "statistical view" by Mease and Wyner (MW), has been pioneered by Breiman (1998, 1999), Friedman et al. (2000), Mason et al. (2000) Rätsch et al. (2001) and is not just a product of the statistics community. The gradient view of boosting allows transferring of the boosting methodology to many other contexts than just classification, see for example Meir and Rätsch (2003) or Bühlmann and Hothorn (2007) for an overview. We should also emphasize that the gradient view has never promised to explain everything about AdaBoost. Hence we are puzzled by the negative picture of this view painted in the paper under discussion: it differs greatly for most part from our experience and understanding of the statistical research on boosting. In particular, the MW paper seems to ignore simulation, real data and theoretical evidence about overfitting and early stopping (cf. Bartlett and Traskin 2007 regarding asymptotic theory for AdaBoost). We will discuss these issues in more details below.

The relevance of MW's counter-examples The evidences in MW are simulated "counter-examples". It is questionable that they are representative of situations encountered in practice. More importantly, with one exception, evidence of differences shown contradicting the so-called "statistical view" are 1 or 2 % in error rate. One wonders how important or meaningful these differences are in practice, even though they might be statistically significant. In any real world situation, the model used is for sure wrong and the approximation error of the model to the real situation could easily swallow these small differences in performance.

Furthermore, all the evaluation metrics in the MW paper are on statistical performance without any consideration of the computation involved or the meaning of the model derived. For large data problems, computation is an indispensable player and needs to be in the picture.

Additive decision boundary but non-additive logit-probabilities MW's model (in Section 3) is additive for the decision boundary. In terms of conditional probabilities $p(x) = \mathbb{P}[Y = 1|X = x]$ on the logit-scale, $\text{logit}(p(x))$ is *not* an additive function in the (feature) components of x.

Since the population minimizer of (gradient) AdaBoost or also of LogitBoost equals

$$F_{\text{pop}}(x) = 0.5 \cdot \text{logit}(p(x)) = 0.5 \cdot \log\left(\frac{p(x)}{1-p(x)}\right),$$

a (boosting) estimate will be good if it involves an effective parameterization. We believe that this is a central insight, which has been pioneered by Breiman (1998, 1999), Friedman et al. (2000) and which has been further developed by more recent asymptotic results on boosting. In the MW model, $F_{\text{pop}}(x)$ is non-additive in x while boosting with stumps yields an estimate $\hat{f}(x)$ which is additive in x. We think that this is the main reason why some of the figures in MW lead to "contrary" evidence: with our model, as illustrated below, the comparison of stumps versus larger trees for weak learners is always in favor of stumps, that is, stumps yield better performance and larger trees are more heavily overfitting which is the opposite finding to Figures 1, 2, and 11 in MW. MW's model in Section 4 involves only a single component of x and hence it is additive also on the logit-scale for the probabilities. But our own model described below does not confirm MW's statement that their findings "do not depend on a particular simulation model".

Other issues in MW concerning "contrary" evidence cannot be easily explained by the nature of the model.

Figure 3 intends to show that LogitBoost is worse than AdaBoost. The MW finding might seem relevant at 1000 iterations. But one doesn't need to go that far for both methods by early stopping. 100 or so iterations seems enough for stumps and 400 for 8-node trees. The performance difference is then less than 1%. Thus, having some computation savings in mind, early stopped LogitBoost is preferable.

Figure 4 tries to make the point that early stopping could hurt to lose about 1% performance when the total Bayes error is 20% and there is no structure to be learned. However, the 1000 iteration model undoubt-fully gives the wrong impression that something is there, while the early stopped model gives the correct impression that not much is to be learned. Hence we think early stopping is not hurting here. In addition, the starting value of boosting matters but this issue is ignored in standard AdaBoost. A (gradient) boosting algorithm should be started with $F_{init} = F_0 \equiv 0.5\log(\hat{p}/(1-\hat{p}))$ where $\hat{p}$ is the empirical frequency of $Y = 1$, cf. Bühlmann and Hothorn (2007). That is, boosting would (try to) improve upon the MLE from the "pure noise" model. Then, it is expected - and we checked this using gradient LogitBoost on the unbalanced example corresponding to Figure 4 in MW - that boosting will overfit from the beginning because the underlying structure is pure noise. The same idea could be applied to AdaBoost as well: in contrast, standard AdaBoost and MW start with the naive value $F_{init} = F_0 \equiv 0$.

Shrinkage and small step-sizes: another dimension for regularization MW makes some claims about additional shrinking using small step-sizes. As we understand Friedman (2001), he *never* intended to say that a shrinkage factor would avoid overfitting. Instead, he argued that introducing a shrinkage factor may improve the performance. Later, Efron et al. (2004) made the connection,

in the setting of linear models, that boosting with an infinitesimally small shrinkage step is equivalent to the Lasso under some conditions, and for general situations, Zhao and Yu (2007) showed that appropriate backward steps need to be added to boosting to get Lasso. This intriguing connection shows again that the shrinkage factor cannot eliminate overfitting. All what it achieves is a different, usually more powerful solution path (with a new regularization dimension through the step-size) than without shrinkage.

Our own findings with an additive model for logit-probabilities Now we devise our own simulation model to clarify some issues regarding overfitting, choice of weak learner and the estimation of probabilities via boosting. Arguably, as emphasized above, examples should not be overinterpreted. However, in view of many reported findings similar to what we show here, we feel that our examples are rather "representative" and we are reporting major instead of slight differences.

Our model is in the spirit of MW but on the logit-scale:

$$\begin{aligned}&\text{logit}(p(x)) = 8\sum_{j=1}^{5}(x_j - 0.5)\\ &Y \sim \text{Bernoulli}(p(x)),\end{aligned}$$

where $p(x) = \mathbb{P}[Y|X = x]$. This model has Bayes error rate approximately equal to 0.1 (as in the MW paper). We use this model as it is additive on the *logit-scale* for the probabilities since the population minimizer of (gradient) AdaBoost and (gradient) LogitBoost is $0.5\text{logit}(p(x))$. We use $n = 100$, $d = 20$ (i.e., 15 ineffective features), x as in MW and we show the results for one representative example with test set of size 2000. We skipped the repetition step over many realizations from the model: again, we think that one realization is representative and it mimics somewhat better the situation of analyzing one real data set.

We consider the misclassification test error, the surrogate loss test error (e.g., the test set average of $\exp(-y\hat{f})$ for AdaBoost) and the absolute error for probabilities

$$\frac{1}{2000}\sum_{i=1}^{2000}|\hat{p}(X_i) - p(X_i)|,$$

where averaging is over the test set.

All our computations have been done with MW's code for AdaBoost and the `R`-package `mboost` from Bühlmann and Hothorn (2007): we used stumps and larger trees as weak learners. By the way, MW's code is *not* implementing 8 node trees but trees which have on average about 6-8 terminal nodes (during the boosting iterations for this model). The results are displayed in Figures 1- 3. A comparison is also made to the naive estimator with $\hat{p}(x) \equiv 0.5$.

From this very limited experiment we find all facts that we view as important and typical for boosting:

1. Overfitting can be a severe issue when considering the test surrogate loss or for estimating conditional probabilities. In fact, overfitting is seen clearly for all three methods, that is gradient AdaBoost, LogitBoost and AdaBoost. In addition, the misclassification loss is much more insensitive with respect to overfitting. This has been pointed out very clearly in Bühlmann and Yu (2000) and in the rejoinder of Friedman et al. (2000).

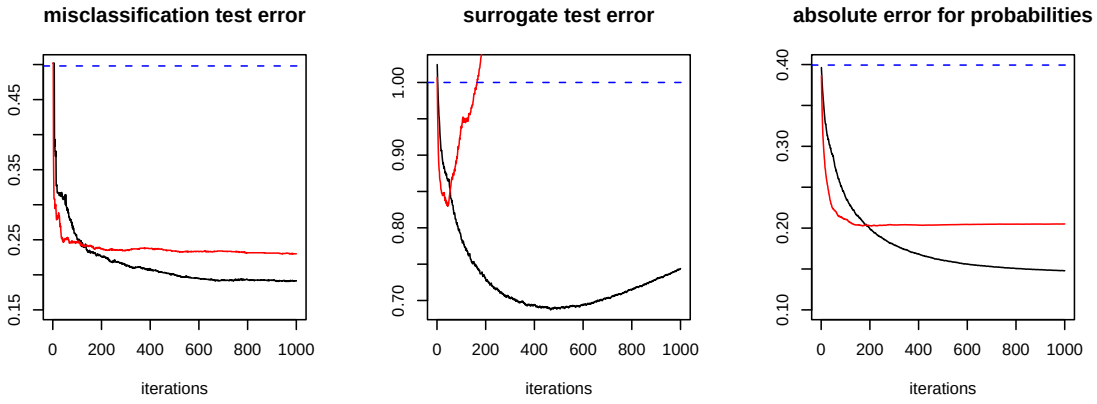


Figure 1: Gradient boosting with exponential loss (gradient AdaBoost). Left panel: Test set misclassification error; Middle panel: test set surrogate loss; Right panel: test set absolute error for probabilities. Black: stumps; Red: larger tree; Blue dashed line: naive estimator.

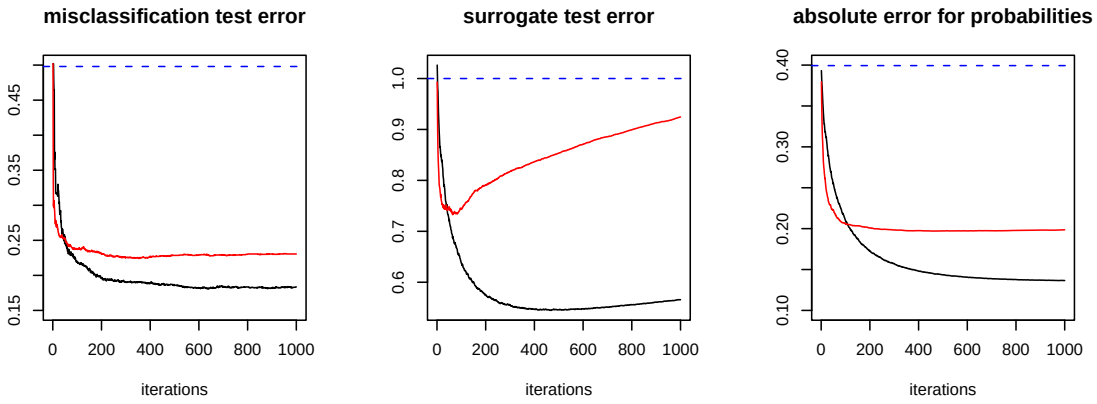


Figure 2: Gradient boosting with Binomial log-likelihood (gradient LogitBoost). Left panel: Test set misclassification error; Middle panel: test set surrogate loss; Right panel: test set absolute error for probabilities. Black: stumps; Red: larger tree; Blue dashed line: naive estimator.

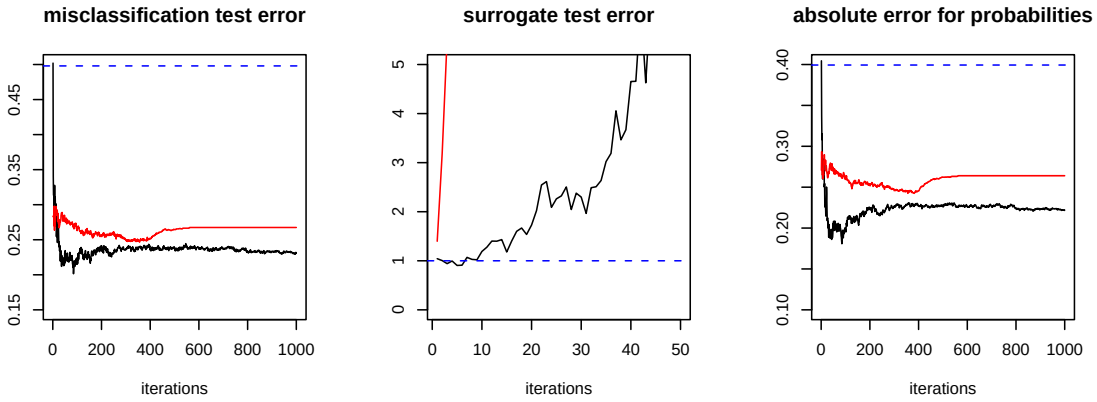


Figure 3: AdaBoost (as in MW). Left panel: Test set misclassification error; Middle panel: test set surrogate loss; Right panel: test set absolute error for probabilities. Black: stumps; Red: larger tree; Blue dashed line: naive estimator. More details are described in point 4 of our summary of findings.

2. Estimating conditional probabilities is quite reasonable when stopping early: as in point 1 above, we see very clearly that early stopping is absolutely crucial for all three methods. And LogitBoost with early stopping gives the best misclassification error and best probability estimate among the three.

3. Regarding the weak learner, larger trees are worse than stumps for our model where the conditional probability function is additive on the logit scale. The "statistical view" reveals the model behind AdaBoost and LogitBoost: we have to consider the logit-scale (the MW model is *not* additive in terms of the logit of conditional probabilities; note that for the decision boundary the scale doesn't matter while it does play a role for conditional probabilities).

   Larger trees do overfit more heavily for probability estimation or with respect to surrogate test loss. For non-additive models (for probabilities on the logit-scale), the overfitting will kick in later for large trees as the the underlying model requires a more complex fit to balance approximation ("bias") and stochastic error ("variance").

4. Somewhat more in line with the MW paper, the original AdaBoost has less a tendency to overfit than the gradient boosting version. The reason why AdaBoost with the larger tree in Figure 3 is staying constant after a while is due to the fact that the algorithm gets "stuck": it alternates back and forth and hence, the amount of overfitting is limited. At this stage of alternating behavior the estimated conditional class probabilities are very much concentrated around either zero or one (not shown but similar to Fig. 18 in MW), that is, overfitting has kicked in severely. We are not convinced that this "getting stuck" property of the algorithm is desirable, despite the consequence that a bound on overfitting is then obviously in action. The surrogate loss function in AdaBoost explodes much earlier (w.r.t. boosting iterations) and one needs to implement an upper bound in the program in order to avoid NA values (MW's code needs some small modification here!).

Our general understanding about Boosting and it's success Instead of going through all issues in MW, we choose instead to repeat several general understandings about boosting which were incorrectly questioned by the paper under discussion:

A. Overfitting does matter, and it is a function of the both the "bias" and "variance". Large trees do not overfit heavily in terms of classification error because:
(i) the misclassification loss is very insensitive to overfitting (see Bühlmann and Yu 2000 and the rejoinder of Friedman et al. 2000);
(ii) larger trees are not as "complex" as the number of nodes in them indicates since they are fitted in a greedy fashion (e.g., 8-node trees fitted by boosting are not 4 times as complex as stumps with two nodes).

   Most probably, the difference between plain vanilla AdaBoost and a gradient version of AdaBoost (as in MW) will *not* play a crucial role in terms of overfitting behavior; but gradient-based boosting seems somewhat more exposed to overfitting while AdaBoost can get stuck which naturally limits the amount of overfitting (on a single data-set).

B. Early stopping, particularly for probability estimation, is very important (because of overfitting) and brings computational savings. The supporting theory is given in, for example, Zhang and Yu (2005), Bühlmann (2006), Bartlett and Traskin (2007) and Bissantz et al. (2007).

C. Estimating probability via boosting is often quite reasonable. It is essential though to tune the boosting algorithm appropriately: a good choice is to do early stopping with respect to the log-likelihood test score (see next point regarding surrogate and evaluating loss).

D. It is important to distinguish between surrogate loss (implementing loss) and loss (evaluating loss) function. For example, there is no surprise that it can happen with AdaBoost that the training misclassification error is zero while the test set misclassification still decreases.

The usage of boosting as we have advocated in our works, and this is very much in line with Friedman et al. (2000) and their subsequent works, has proven to be very competitive and successful in applications. Gao et al. (2006) describe a successful application of boosting to a language transliteration problem. Lutz (2006) has won the performance prediction challenge of the world congress in computational intelligence in 2006 (WCCI 2006): he was using early-stopped LogitBoost with stumps. Part of his success is probably due to careful choice of choosing the stopping iteration: according to personal communication (he has been a former PhD student of the first author of this discussion), he stopped before reaching the minimal value of a cross-validation scheme. In summary, he did not take any of the findings from MW into account (he didn't know the paper at that time, of course). Maybe his success is more convincing evidence that LogitBoost with (i) its "natural" loss function for a binary classification problem, and using (ii) early stopping, (iii) simple weak learners and (iv) a small step size (i.e., shrinkage factor) often works surprisingly well. Other references about successful applications of gradient-based boosting can be found in Bühlmann and Hothorn (2007) which includes the `R` package `mboost` (standing for model-based boosting) for numerous application areas ranging from classification, regression, generalized regression to survival analysis.

References

P. L. Bartlett and M. Traskin. AdaBoost is consistent. *Journal of Machine Learning Research*, 8: 2347–2368, 2007.

N. Bissantz, T. Hohage, A. Munk, and F. Ruymgaart. Convergence rates of general regularization methods for statistical inverse problems and applications. *SIAM Journal on Numerical Analysis*, 45:2610–2636, 2007.

L. Breiman. Arcing classifiers (with discussion). *The Annals of Statistics*, 26:801–849, 1998.

L. Breiman. Prediction games & arcing algorithms. *Neural Computation*, 11:1493–1517, 1999.

P. Bühlmann. Boosting for high-dimensional linear models. *The Annals of Statistics*, 34:559–583, 2006.

P. Bühlmann and T. Hothorn. Boosting algorithms: regularization, prediction and model fitting (with discussion). *Statistical Science*, 22, 2007.

P. Bühlmann and B. Yu. Discussion of "Additive logistic regression: a statistical view" (J. Friedman, T. Hastie and R. Tibshirani, auths.). *The Annals of Statistics*, 28:377–386, 2000.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression (with discussion). *The Annals of Statistics*, 32:407–451, 2004.

J.H. Friedman. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 29:1189–1232, 2001.

J.H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion). *The Annals of Statistics*, 28:337–407, 2000.

J. Gao, H. Suzuki, and B. Yu. Approximation Lasso methods for language modeling. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 225–232, 2006.

R.W. Lutz. LogitBoost with trees applied to the WCCI 2006 performance prediction challenge datasets. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2006.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, Cambridge, 2000.

R. Meir and G. Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Computer Science, pages 119–184. Springer, 2003.

G. Rätsch, T. Onoda, and K.R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42:287–320, 2001.

T. Zhang and B. Yu. Boosting with early stopping: convergence and consistency. *The Annals of Statistics*, 33:1538–1579, 2005.

P. Zhao and B. Yu. Stagewise Lasso. *Journal of Machine Learning Research*, 8:2701–2726, 2007.

Journal of Machine Learning Research 9 (2008) 195-201 Published 2/08

Evidence Contrary to the Statistical View of Boosting: A Rejoinder to Responses

David Mease MEASE_D@COB.SJSU.EDU
Department of Marketing and Decision Sciences
College of Business, San Jose State University
San Jose, CA 95192-0069, USA

Abraham Wyner AJW@WHARTON.UPENN.EDU
Department of Statistics
Wharton School, University of Pennsylvania
Philadelphia, PA, 19104-6340, USA

Editor: Yoav Freund

1. Introduction

We thank the discussants for their comments on our paper. We also thank the editors for arranging the discussions. Many interesting points have been raised by the discussants. We can not respond to everything, but we do include a section addressing the main points of each discussant. Following these, we provide a final section in which we give some general concluding remarks.

Many of the discussants comment on the overfitting of boosting. Different authors will have different ideas of what the term overfitting means in the context of boosting, but for clarification throughout this rejoinder we will define overfitting as a positive slope for a specified loss metric as a function of the iterations. Specifically, the loss metric we focus on is misclassification error, although we understand that some of the discussants are concerned about other loss functions which quantify probability estimation accuracy rather than classification accuracy. The importance of focusing on misclassification error is underscored in the discussion by Freund and Schapire who remind us that AdaBoost is an algorithm for carrying out classification, not probability estimation.

2. Rejoinder for Kristin P. Bennett

Bennett provides a useful perspective on the situation by studying the convergence of boosting algorithms from the optimization point of view. We agree that this aspect of the problem is too often overlooked by researchers.

In studying the convergence of the algorithms, Bennett touches on a number of important considerations. For example, she mentions cycling in the context of stumps and notes that the cycling results in boosting using only a small number of unique trees. The number of unique trees is rarely noted by researchers in empirical investigations. Bennett's studies also lead her to concur that boosting algorithms sometimes benefit from a bagging type of self-averaging during what she calls the "overtraining" stage. ("Overtraining" as defined by Bennett should not be confused with "over*fitting*" as we have defined it). Another point on which we strongly agree with Bennett is that

boosting's resistance to overfitting can occur for different reasons in different contexts. We believe that this is one reason why researchers have difficulty in coming to agreement regarding various explanations for boosting.

In addition to studying convergence, Bennett also looks at the margin of the classification rules. We mentioned margin theory in our paper only briefly since our focus was on the statistical view of boosting, and margin theory is generally separate from this view. It is our hope, however, that by finding holes in the accepted statistical view we are encouraging researchers to approach the problem from different perspectives to help explain the phenomena left unexplained by the statistical view. Unfortunately, in this case Bennett points out that examination of the margin still fails to explain the results of the experiment in Section 3.2.

We believe that studying boosting as an algorithm (rather than as a statistical model) in the way Bennett has done can be quite helpful in understanding some of its remaining mysteries. We are glad that our examples have inspired this type of investigation.

3. Rejoinder for Andreas Buja and Werner Stuetzle

One of the main points argued by Buja and Stuetzle is that most of the current literature on boosting does not explain its variance reduction. They argue that a complete view of boosting should explain both its ability to reduce bias and variance. We certainly agree with this point. In fact, some of the examples in our paper such as those in Sections 3.4 and 4.4 illustrate this quite well. It is interesting that Buja and Stuetzle site references from the early research on boosting which argue in this same direction. It is a shame that more attention has not been given to boosting's ability to reduce variance in addition to bias in more recent research. We hope that our paper helps to rejuvenate research on this aspect of boosting.

Buja and Stuetzle go on to argue that there is often a need for more than a black box classifier which produces small misclassification error. Some applications call for interpretable and diagnostic models and/or conditional class probability estimates. This is certainly true, and we agree that research that extends boosting in these directions is quite welcome. However, in carrying out this research it is important to be honest about situations in which the theory does not explain the performance of traditional boosting algorithms for classification. One of our purposes in writing this paper was to promote this honesty.

4. Rejoinder for Yoav Freund and Robert E. Schapire

Freund and Schapire focus their discussion on margin theory. It is quite interesting that margin theory has not been embraced much at all by the statistical community. In fact, Freund and Schapire's paper "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods" appeared less than two years before the Friedman, Hastie and Tibshirani paper "Additive Logistic Regression: A Statistical View Of Boosting" in the same journal (*Annals of Statistics*). Despite this, the statistics community has largely ignored it in favor of the more familiar theory in the latter paper.

Freund and Schapire make a case for margin theory by arguing that this theory explains the results of the experiment in Section 3.1 while obviously the statistical theory does not. However, it should be noted that Bennett believes margin theory still does not explain the results for Section 3.2. Despite this, we still believe that margin theory is worth pursuing, especially given the large number of inconsistencies between the statistical view and the reality of the simulations presented

in our paper. As we have mentioned, we wrote the paper with the goal of promoting alternative explanations for boosting other than the statistical view, which leaves much unexplained. Margin theory is one such alternative explanation.

Additionally, Freund and Schapire make a number of other points on which we agree and wish to underscore. They argue that in the statistics research on boosting too much importance is placed on class probability estimation over class estimation, the use of stumps over larger trees and theoretical questions of consistency and asymptotic variance over more relevant theoretical questions.

5. Rejoinder for Jerome Friedman, Trevor Hastie and Robert Tibshirani

Friedman, Hastie and Tibshirani argue that many of the ten statements made in our paper should not be ascribed to the statistical view as laid out in Friedman et al. (2000). This is understandable, and in the case of the similarity with nearest neighbor algorithms, for example, we have even noted this in our original paper. However, other statements could be argued to follow fairly directly, whether that is the intent of the authors or the fault of the reader. At the very least, we believe it is fair to say the statistical view offers very little to help explain the non-intuitive nature of the results in our paper.

Of all ten statements, the most direct relationship to the work of Friedman, Hastie and Tibshirani is the idea in Sections 3.1 and 4.1 that stumps should be used for additive Bayes rules. We believe the authors would agree that this is the strongest connection to their work, which is why they focused the majority of their discussion on the experiment in Section 3.1. We also think it is refreshing that they admit that they "are not sure why" the results of this experiment are such, but they do produce some graphs to try to understand this better. Some of their graphs show the performance of the probability estimates. It is argued in the discussion by Freund and Schapire that probability estimation for boosting has very little to do with boosting's classification performance. We agree and for this reason we will not comment on the graphs for probability estimation. However, the graphs showing misclassification error are of interest and we discuss these below.

Friedman, Hastie and Tibshirani's Figure 1 shows that using shrinkage in our original experiment causes stumps to "win handily". We note, however, that the shrinkage causes overfitting, so it also becomes necessary to stop the boosting process before 600 iterations to realize any advantage over the 8-node trees. In practice the optimal stopping time is not known, and a fair comparison would require incorporating the uncertainty in the estimation of this value.

The right panel of their Figure 2 shows that with a larger training sample size of $n = 2000$ the overfitting caused by shrinkage is not as severe and the stumps maintain their advantage over the full 1000 iterations. However, by 1000 iterations the overfitting for the stumps has caused the gap with the 8-node trees to close considerably and extrapolation would suggest that additional iterations would result in this gap becoming even smaller or disappearing altogether. Meanwhile, the 8-node trees again show no signs of overfitting.

We believe the more interesting research question with regard to Friedman, Hastie and Tibshirani's Figures 1 and 2 is not which algorithm performs best for a certain stopping time and sample size, but rather why all algorithms display overfitting with regard to misclassification error except the 8-node trees without shrinkage. The authors state that the larger trees can have a bagging effect, and we certainly agree with this. We feel that understanding this effect better (as well as understanding why shrinkage can destroy this effect) is essential to gaining a better understanding of boosting.

The left panel of Friedman, Hastie and Tibshirani's Figure 3 shows the effect of using Bernoulli loss rather than exponential loss. With Bernoulli loss all algorithms show overfitting. This is another curiosity not explained by the statistical view of boosting. In fact, the paper by Friedman et al. (2000) seems to suggest the opposite should be expected.

In their final paragraph Friedman, Hastie and Tibshirani welcome "*constructive* counter-arguments and alternative explanations for the success of boosting methods." We will make a couple of comments here in response to this. First, we believe that explanations such as the variance reducing bagging effect of boosting fall into this category. Our paper is certainly not the first to suggest this notion, nor do we offer a theoretical explanation for the phenomenon, but our paper does stress the importance of not overlooking this effect and thus we hope promotes constructive research on this topic. Secondly, we feel that researchers currently embrace the statistical view too strongly, and for this reason it is difficult for researchers to offer any alternative explanations without first tearing down the current beliefs to some degree. We base this statement on our own experience. For instance, an early version of Mease et al. (2007) was rejected for publication by a different outlet. That paper offers a method for estimating probabilities using AdaBoost. Two of the three referees rejected the paper arguing that in order to estimate probabilities using boosting it would be sufficient to use LogitBoost in place of AdaBoost.

6. Rejoinder for P. J. Bickel and Ya'acov Ritov

Bickel and Ritov begin by identifying some points we made in our paper with which they agree. The amount of agreement is substantial. The disagreement is focused largely on what is generally regarded to be the most mysterious property of the AdaBoost algorithm: its ability to reduce the (test) error rate long after the training data has been fit without error. Other classifiers such as CART, neural nets, LDA and logistic regression perform optimally only with some sort of appropriate early stopping to prevent over parameterization and overfitting of the data. It is understandable that Bickel and Ritov's negative remarks focus on this particular issue, as it is the feature of AdaBoost most at odds with the statistical view.

The example provided by Bickel and Ritov is a model for which early stopping is essential to achieve optimal classification performance. If AdaBoost is not stopped after 10 or 20 iterations in their example, it will overfit the data and the generalization error will increase steadily. Their example is not the first of its kind, but rather is typical of the simulations used to provide empirical support for the statistical view. We addressed this point directly in our paper:

> *Such examples in which overtting is observed often deal with extremely low-dimensional cases such as $d = 2$ or even $d = 1$. By experimenting with the simulation code provided along with this paper, one can confirm that in general AdaBoost is much more likely to suffer from overtting in trivial low-dimensional examples as opposed to high-dimensional situations where it is more often used.*

The example provided by Bickel and Ritov is of this same spirit. It is a $d = 2$ dimensional model that lives on $d = 1$ dimensional circular manifolds. Since it is well known that AdaBoost will converge to the nearest neighbor classifier in one dimension, the results for this simulation are not unexpected. Along these same lines, Bickel and Ritov provide further evidence for our claim that overfitting is largely a symptom of trivial low dimensional examples by observing that when

they add 8 additional dimensions with no signal the overfitting disappears. The authors refer to this phenomenon as "surprising", but we would argue that this should also be expected.

Bickel and Ritov note interesting changes when they discretize some of the predictors. We did not discuss this in our paper, but the difference in behavior for discrete data and continuous data is tremendous. We learned first hand about this in Mease et al. (2007) when we artificially introduced discreteness by adding replicates. Because of this difference, we focused our current paper on the case of continuous data. The case of discrete data relates more to what Friedman et al. (2000) called the *population version* of boosting. The statistical view of boosting is largely based on this population version of boosting, and thus the statistical view becomes more relevant for discrete data.

Bickel and Ritov also bring up probability estimation in their discussion. Probability estimation is a popular topic among statisticians with regard to boosting, but again we will not go into any detail here because we agree with Freund and Schapire that probability estimation is not the central topic, but rather classification. It is curious, however, that Bickel and Ritov state that "most of the error in the estimation" of the probabilities "comes from the easy to classify points". The truth of this statement depends on how one measures the error in probability estimates. For example, if one computes the RMSE between the true probabilities and the estimated probabilities for the data in Bickel and Ritov's Figure 2(c), it can be observed that the RMSE is actually highest near the points with a true probability of 1/2.

In conclusion, Bickel and Ritov have presented a couple of low dimensional examples in which overfitting occurs unless early stopping is applied, while our paper presents higher dimensional examples where this is not the case. Since "everybody can produce examples" as Bickel and Ritov state, one may wonder which set of examples is more useful to study for the purpose of understanding boosting. We make the argument for our examples based on the fact that higher dimensional examples are more common of a use case for boosting, and that our examples are more similar to the many examples in which overfitting does not occur that first led practitioners to be attracted to boosting originally. We feel that understanding such examples is most useful for understanding boosting.

7. Rejoinder for Peter Buhlmann and Bin Yu

Of the six discussions, Buhlmann and Yu seem to be the least accepting of the empirical findings in our paper. They state that the "main reason" why we obtain contrary evidence in many cases is the functional form of the simulation model, and they propose considering a model that is additive on the logit scale. The model they propose turns out to be a special case of our model in Section 5 with $k = 8$. As stated in the paper, we had already considered the results for this specific model carefully using various values of k, and those results were consistent qualitatively with the other simulations in the paper. Thus, we were surprised to read that Buhlmann and Yu based any disagreement on results from this model. On closer inspection, we learned that the discrepancy is most likely due to sampling error. Buhlmann and Yu considered only a single repetition. The plot in the first panel of their Figure 3 shows that for that single repetition, the 8-node trees begin to overfit near 400 iterations and the stumps have a lower misclassification error throughout the 1000 iterations. However, if the results are averaged over many repetitions, one can learn that this behavior is not true in aggregate. The first plot in Figure 1 in this rejoinder shows the misclassification error for the Buhlmann and Yu model averaged over 100 repetitions. The stumps overfit while the 8-node trees do not, and the 8-node trees achieve a lower misclassification error than the stumps at 1000

iterations. Thus, the results for the Buhlmann and Yu model are consistent with the results for the other simulations in our paper.

It should be noted that with the small sample size of $n = 100$ chosen by Buhlmann and Yu, there is indeed often a problem with floating point overflow errors in R as they mention. They dealt with this issue by making a modification to our code (which is an alternative explanation for the discrepancies in the results). We avoided this issue when making the first plot in Figure 1 in this rejoinder by considering 100 randomly chosen repetitions for which the floating point overflow error did not occur. For readers who are uncomfortable with this, there are a couple of other ways of handling the problem. First, if a larger sample size is chosen then the problem goes away completely. For example, the second plot in Figure 1 in this rejoinder considers our original sample size of $n = 200$. For this sample size there were no problems with overflow errors in 100 repetitions. Again, one can see that the 8-node trees lead to a lower misclassification error at 1000 iterations and do not overfit, while the stumps do overfit. A second way of dealing with the overflow errors for the sample size of $n = 100$ is to consider only the first 500 iterations. When we ran only 500 iterations we did not observe any overflow errors in 100 repetitions, and the results were consistent with those in the first plot in Figure 1 in this rejoinder.

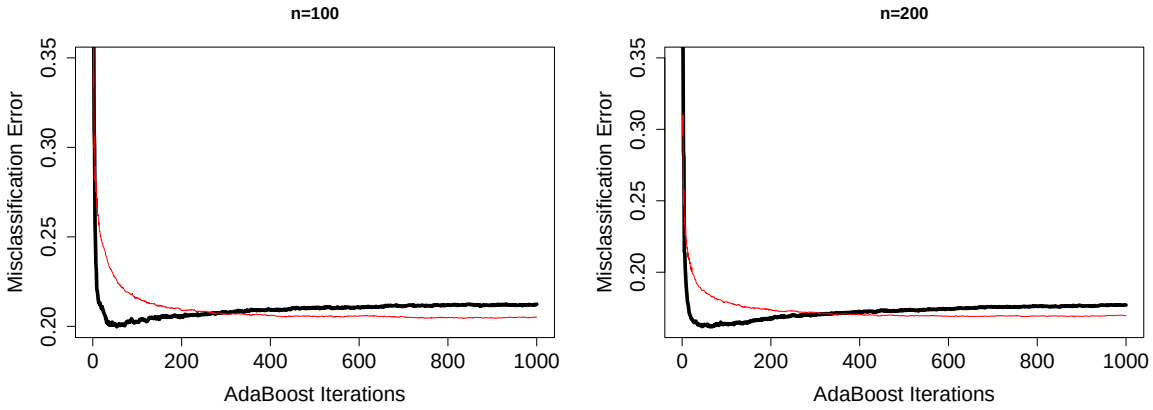


Figure 1: Comparison of AdaBoost with Stumps (Black, Thick) and 8-Node Trees (Red, Thin) for the Buhlmann and Yu Model Using 100 Repetitions and Sample Sizes of $n = 100$ and $n = 200$

Thus, concerning this simulation model proposed by Buhlmann and Yu, we conclude that the results are largely consistent with the other results in our paper. However, Buhlmann and Yu also discuss our Figures 3 and 4 independently of their simulation model. We address this below.

With regard to LogitBoost in our Figure 3, Buhlmann and Yu mention that with early stopping LogitBoost does almost as well as AdaBoost. This assumes one can estimate the optimal stopping time well, when in practice the stopping time estimation using LogitBoost can be difficult. In fact, we mentioned in our paper that the authors of the particular LogitBoost code we used reported that the stopping estimation was not effective for their purposes. Conversely AdaBoost performs fine in our Figure 3 without early stopping. Also, considering that the creators of LogitBoost (Friedman, Hastie and Tibshirani) mentioned in their discussion that they "have moved on" from LogitBoost, there seems to be little left to be said in its defense.

With regard to our Figure 4, Buhlmann and Yu seem unimpressed by this example which shows how AdaBoost can reduce variance. In fact, despite its excellent performance they lament that the classifier "gives the wrong impression." This is an interesting statement and illustrates the desire on the part of the statistical community to view AdaBoost as an interpretable model rather than a black box classifier. Certainly a model can give one the wrong impression but a classifier arguably only classifies well or does not classify well.

In conclusion, Buhlmann and Yu are considerably opposed to our most important claims: 1) that large trees generally work better than stumps and 2) that overfitting is usually not a problem and 3) that early stopping initiatives are often not only unnecessary but also counterproductive. With regard to classification we have shown that their simulation does not provide strong evidence to support their case. However, when the goal is the more difficult problem of conditional probability estimation, we will not offer any disagreement. We have not focused our analysis on this problem, and we do not feel that boosting algorithms should be regarded as "state-of-the-art" probability estimators, since they are usually outperformed by competitors like Random Forests or even logistic regression. Rather, it is the statistical view that asserts that AdaBoost works, erroneously in our opinion, because it estimates probabilities. Indeed, the efforts of Buhlmann and Yu to understand and improve the performance of boosting for probability estimation is productive and worthwhile. Where we part company is in name only. We question whether it is logical to continue to call the algorithms boosting algorithms since there is a considerable disconnect from the original AdaBoost algorithm for classification.

8. Conclusion

We believe the discussions provided by the six sets of authors have been extremely valuable. We are encouraged by the amount of discussion of two main ideas which we feel will lead to a better understanding of boosting.

First, the discussions have helped to clarify that (out of sample) minimization of the surrogate loss function (and equivalently probability estimation) is often a very different problem from (out of sample) minimization of misclassification error. The original AdaBoost algorithm was intended for the latter but much research in the statistical community has focused on the former.

Secondly, with regard to minimization of misclassification error, the idea that boosting is reducing variance has been acknowledged in the discussions. The extent to which this phenomenon is essential for boosting's success and of interest as a research topic can be debated, but we are encouraged to see it acknowledged by such a prominent group of researchers.

References

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:337–374, 2000.

D. Mease, A. Wyner, and A. Buja. Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research*, 8:409–439, 2007.

Journal of Machine Learning Research 9 (2008) 203-233 Submitted 5/07; Published 2/08

Optimization Techniques for Semi-Supervised Support Vector Machines

Olivier Chapelle[*] CHAP@YAHOO-INC.COM
Yahoo! Research
2821 Mission College Blvd
Santa Clara, CA 95054

Vikas Sindhwani VIKASS@CS.UCHICAGO.EDU
University of Chicago, Department of Computer Science
1100 E 58th Street
Chicago, IL 60637

Sathiya S. Keerthi SELVARAK@YAHOO-INC.COM
Yahoo! Research
2821 Mission College Blvd
Santa Clara, CA 95054

Editor: Nello Cristianini

Abstract

Due to its wide applicability, the problem of semi-supervised classification is attracting increasing attention in machine learning. Semi-Supervised Support Vector Machines (S^3VMs) are based on applying the margin maximization principle to both labeled and unlabeled examples. Unlike SVMs, their formulation leads to a non-convex optimization problem. A suite of algorithms have recently been proposed for solving S^3VMs. This paper reviews key ideas in this literature. The performance and behavior of various S^3VM algorithms is studied together, under a common experimental setting.

Keywords: semi-supervised learning, support vector machines, non-convex optimization, transductive learning

1. Introduction

In many applications of machine learning, abundant amounts of data can be cheaply and automatically collected. However, manual labeling for the purposes of training learning algorithms is often a slow, expensive, and error-prone process. The goal of semi-supervised learning is to employ the large collection of unlabeled data jointly with a few labeled examples for improving generalization performance.

The design of Support Vector Machines (SVMs) that can handle partially labeled data sets has naturally been a vigorously active subject. A major body of work is based on the following idea: solve the standard SVM problem while treating the unknown labels as additional optimization variables. By maximizing the margin in the presence of unlabeled data, one learns a decision boundary that traverses through low data-density regions while respecting labels in the input space. In other words, this approach implements the *cluster assumption* for semi-supervised learning—that is,

∗. Most of the work was done while at MPI for Biological Cybernetics,Tübingen, Germany.

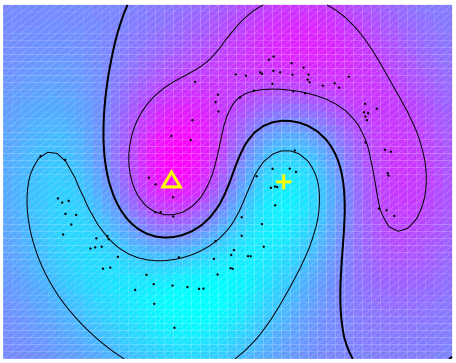


Figure 1: Two moons. There are 2 labeled points (the triangle and the cross) and 100 unlabeled points. The global optimum of S^3VM correctly identifies the decision boundary (black line).

points in a data cluster have similar labels (Seeger, 2006; Chapelle and Zien, 2005). Figure 1 illustrates a low-density decision surface implementing the cluster assumption on a toy two-dimensional data set. This idea was first introduced by Vapnik and Sterin (1977) under the name *Transductive SVM*, but since it learns an inductive rule defined over the entire input space, we refer to this approach as *Semi-Supervised SVM* (S^3VM).

Since its first implementation by Joachims (1999), a wide spectrum of techniques have been applied to solve the non-convex optimization problem associated with S^3VMs, for example, local combinatorial search (Joachims, 1999), gradient descent (Chapelle and Zien, 2005), continuation techniques (Chapelle et al., 2006a), convex-concave procedures (Fung and Mangasarian, 2001; Collobert et al., 2006), semi-definite programming (Bie and Cristianini, 2006; Xu et al., 2004), non-differentiable methods (Astorino and Fuduli, 2007), deterministic annealing (Sindhwani et al., 2006), and branch-and-bound algorithms (Bennett and Demiriz, 1998; Chapelle et al., 2006c).

While non-convexity is partly responsible for this diversity of methods, it is also a departure from one of the nicest aspects of SVMs. Table 1 benchmarks the empirical performance of various S^3VM implementations against the globally optimal solution obtained by a Branch-and-Bound algorithm. These empirical observations strengthen the conjecture that the performance variability of S^3VM implementations is closely tied to their susceptibility to sub-optimal local minima. Together with several subtle implementation differences, this makes it challenging to cross-compare different S^3VM algorithms.

The aim of this paper is to provide a review of optimization techniques for semi-supervised SVMs and to bring different implementations, and various aspects of their empirical performance, under a common experimental setting.

In Section 2 we discuss the general formulation of S^3VMs. In Sections 3 and 4 we provide an overview of various methods. We present a detailed empirical study in Section 5 and present a discussion on complexity in Section 6.

	∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton	BB
Coil3	61.6	61	56.6	56.7	61.6	61.5	0
2moons	61	37.7	63.1	68.8	22.5	11	0

Table 1: Generalization performance (error rates) of different S^{3}VM algorithms on two (small) data sets Coil3 and 2moons. Branch and Bound (BB) yields the globally optimal solution which gives perfect separation. BB can only be applied to small data sets due to its high computational costs. See Section 5 for experimental details.

2. Semi-Supervised Support Vector Machines

We consider the problem of binary classification. The training set consists of l labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $y_i = \pm 1$, and u unlabeled examples $\{\mathbf{x}_i\}_{i=l+1}^{n}$, with $n = l+u$. In the linear S^{3}VM classification setting, the following minimization problem is solved over *both* the hyperplane parameters $(\mathbf{w}, b)$ and the label vector $\mathbf{y}_u := [y_{l+1} \dots y_n]^\top$,

$$\min_{(\mathbf{w},b),\ \mathbf{y}_u} I(\mathbf{w}, b, \mathbf{y}_u) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} V(y_i, o_i) + C^\star \sum_{i=l+1}^{n} V(y_i, o_i) \tag{1}$$

where $o_i = \mathbf{w}^\top \mathbf{x}_i + b$ and V is a loss function. The Hinge loss is a popular choice for V,

$$V(y_i, o_i) = \max(0, 1 - y_i o_i)^p . \tag{2}$$

It is common to penalize the Hinge loss either linearly ($p = 1$) or quadratically ($p = 2$). In the rest of the paper, we will consider $p = 2$. Non-linear decision boundaries can be constructed using the *kernel trick* (Vapnik, 1998).

The first two terms in the objective function I in (1) define a standard SVM. The third term incorporates unlabeled data. The loss over labeled and unlabeled examples is weighted by two hyperparameters, C and $C^\star$, which reflect confidence in labels and in the cluster assumption respectively. In general, C and $C^\star$ need to be set at different values for optimal generalization performance.

The minimization problem (1) is solved under the following class balancing constraint,

$$\frac{1}{u}\sum_{i=l+1}^{n} \max(y_i, 0) = r \quad \text{or equivalently} \quad \frac{1}{u}\sum_{i=l+1}^{n} y_i = 2r - 1. \tag{3}$$

This constraint helps in avoiding unbalanced solutions by enforcing that a certain user-specified fraction, r, of the unlabeled data should be assigned to the positive class. It was introduced with the first S^{3}VM implementation (Joachims, 1999). Since the true class ratio is unknown for the unlabeled data, r is estimated from the class ratio on the labeled set, or from prior knowledge about the classification problem.

There are two broad strategies for minimizing I:

1. **Combinatorial Optimization:** For a given fixed $\mathbf{y}_u$, the optimization over $(\mathbf{w}, b)$ is a standard SVM training.[1] Let us define:

$$\mathcal{J}(\mathbf{y}_u) = \min_{\mathbf{w},b} \ I(\mathbf{w}, b, \mathbf{y}_u). \tag{4}$$

1. The SVM training is slightly modified to take into account different values for C and $C^\star$.

The goal now is to minimize $\mathcal{J}$ over a set of binary variables. This combinatorial view of the optimization problem is adopted by Joachims (1999), Bie and Cristianini (2006), Xu et al. (2004), Sindhwani et al. (2006), Bennett and Demiriz (1998), and Chapelle et al. (2006c). There is no known algorithm that finds the global optimum efficiently. In Section 3 we review this class of techniques.

2. **Continuous Optimization:** For a fixed $(\mathbf{w}, b)$, $\arg\min_y V(y, o) = \text{sign}(o)$. Therefore, the optimal $\mathbf{y}_u$ is simply given by the signs of $o_i = \mathbf{w}^\top \mathbf{x}_i + b$. Eliminating $\mathbf{y}_u$ in this manner gives a continuous objective function over $(\mathbf{w}, b)$:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} \max(0, 1 - y_i o_i)^2 + C^\star \sum_{i=l+1}^{n} \max(0, 1 - |o_i|)^2 . \tag{5}$$

This form of the optimization problem illustrates how S^{3}VMs implement the cluster assumption. The first two terms in (5) correspond to a standard SVM. The last term (see Figure 2) drives the decision boundary, that is, the zero output contour, away from unlabeled points. From Figure 2, it is clear that the objective function is non-convex.

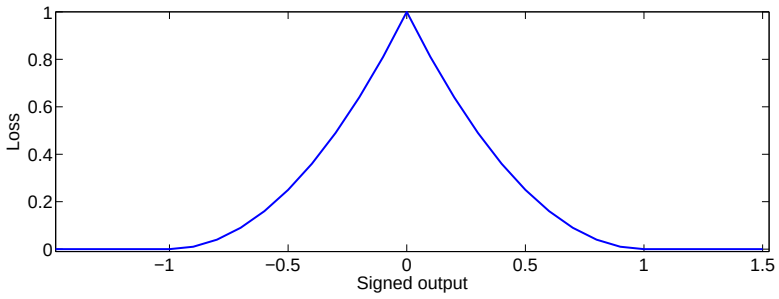


Figure 2: The effective loss $\max(0, 1 - |o|)^2$ is shown above as a function of $o = (\mathbf{w}^\top \mathbf{x} + b)$, the real-valued output at an unlabeled point $\mathbf{x}$.

Note that in this form, the balance constraint becomes $\frac{1}{u}\sum_{i=l+1}^{n} \text{sign}(\mathbf{w}^\top \mathbf{x}_i + b) = 2r - 1$ which is non-linear in $(\mathbf{w}, b)$ and not straightforward to enforce. In Section 4 we review this class of methods (Chapelle and Zien, 2005; Chapelle et al., 2006a; Fung and Mangasarian, 2001; Collobert et al., 2006).

3. Combinatorial Optimization

We now discuss combinatorial techniques in which the labels $\mathbf{y}_u$ of the unlabeled points are explicit optimization variables. Many of the techniques discussed in this section call a standard (or slightly modified) supervised SVM as a subroutine to perform the minimization over $(\mathbf{w}, b)$ for a fixed $\mathbf{y}_u$ (see 4).

3.1 Branch-and-Bound (BB) for Global Optimization

The objective function (4) can be globally optimized using Branch-and-Bound techniques. This was noted in the context of S^3VM by Wapnik and Tscherwonenkis (1979) but no details were presented there. In general, global optimization can be computationally very demanding. The technique described in this section is impractical for large data sets. However, with effective heuristics it can produce globally optimal solutions for small-sized problems. This is useful for benchmarking practical S^3VM implementations. Indeed, as Table 1 suggests, the exact solution can return excellent generalization performance in situations where other implementations fail completely. Branch-and-Bound was first applied by Bennett and Demiriz (1998) in association with integer programming for solving linear S^3VMs. More recently Chapelle et al. (2006c) presented a Branch-and-Bound (BB) algorithm which we outline in this section. The main ideas are illustrated in Figure 3.

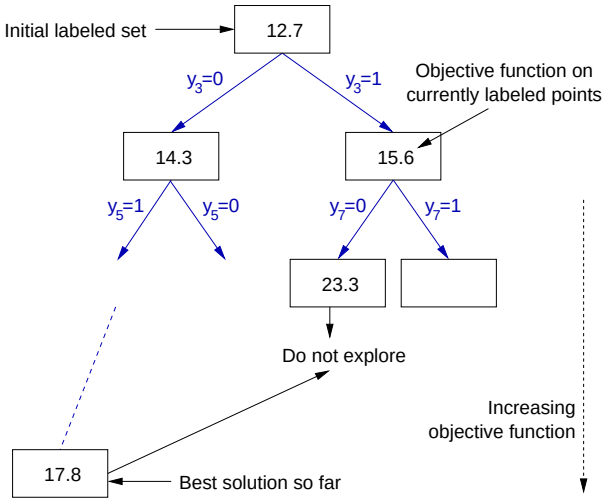


Figure 3: Branch-and-Bound Tree

Branch-and-Bound effectively performs an exhaustive search over $\mathbf{y}_u$, pruning large parts of the solution space based on the following simple observation: suppose that a lower bound on $\min_{\mathbf{y}_u \in A} \mathcal{J}(\mathbf{y}_u)$, for some subset A of candidate solutions, is greater than $\mathcal{J}(\tilde{\mathbf{y}}_u)$ for some $\tilde{\mathbf{y}}_u$, then A can be safely discarded from exhaustive search. BB organizes subsets of solutions into a binary tree (Figure 3) where nodes are associated with a fixed partial labeling of the unlabeled data set and the two children correspond to the labeling of some new unlabeled point. Thus, the root corresponds to the initial set of labeled examples and the leaves correspond to a complete labeling of the data. Any node is then associated with the subset of candidate solutions that appear at the leaves of the subtree rooted at that node (all possible ways of completing the labeling, given the partial labeling at that node). This subset can potentially be pruned from the search by the Branch-and-Bound procedure if a lower bound over corresponding objective values turns out to be worse than an available solution.

The effectiveness of BB depends on the following design issues: (1) the lower *bound* at a node and (2) the sequence of unlabeled examples to *branch* on. For the lower bound, Chapelle et al. (2006c) use the objective value of a standard SVM trained on the associated (extended) labeled

set.[2] As one goes down the tree, this objective value increases as additional loss terms are added, and eventually equals $\mathcal{J}$ at the leaves. Note that once a node violates the balance constraint, it can be immediately pruned by resetting its lower bound to ∞. Chapelle et al. (2006c) use a labeling-confidence criterion to choose an unlabeled example and a label on which to branch. The tree is explored on the fly by depth-first search. This confidence-based tree exploration is also intuitively linked to Label Propagation methods (Zhu and Ghahramani, 2002) for graph-transduction. On many small data sets (e.g., Table 1 data sets have up to 200 examples) BB is able to return the globally optimal solution in reasonable amount of time. We point the reader to Chapelle et al. (2006c) for pseudocode.

3.2 S^{3}VM^{light}

S^{3}VM^{light} (Joachims, 1999) refers to the first S^{3}VM algorithm implemented in the popular SVM^{light} software.[3] It is based on local combinatorial search guided by a label switching procedure. The vector $\mathbf{y}_u$ is initialized as the labeling given by an SVM trained on the labeled set, thresholding outputs so that $u \times r$ unlabeled examples are positive. Subsequent steps in the algorithm comprise of switching labels of two examples in opposite classes, thus always maintaining the balance constraint. Consider an iteration of the algorithm where $\mathbf{y}_u$ is the temporary labeling of the unlabeled data and let $(\tilde{\mathbf{w}}, \tilde{b}) = \text{argmin}_{\mathbf{w},b} \;\; I(\mathbf{w}, b, \mathbf{y}_u)$ and $\mathcal{J}(\mathbf{y}_u) = I(\tilde{\mathbf{w}}, \tilde{b}, \mathbf{y}_u)$. Suppose a pair of unlabeled examples indexed by (i, j) satisfies the following condition,[4]

$$y_i = 1, y_j = -1, V(1, o_i) + V(-1, o_j) > V(-1, o_i) + V(1, o_j) \tag{6}$$

where o_i, o_j are outputs of $(\tilde{\mathbf{w}}, \tilde{b})$ on the examples $\mathbf{x}_i, \mathbf{x}_j$. Then after switching labels for this pair of examples and retraining, the objective function $\mathcal{J}$ can be easily shown to strictly decrease. S^{3}VM^{light} alternates between label-switching and retraining. Since the number of possible $\mathbf{y}_u$ is finite, the procedure is guaranteed to terminate in a finite number of steps at a local minima of (4), that is, no further improvements are possible by interchanging two labels.

In an outer loop, S^{3}VM^{light} gradually increases the value of $C^\star$ from a small value to the final value. Since $C^\star$ controls the non-convex part of the objective function (4), this annealing loop can be interpreted as implementing a "smoothing" heuristic as a means to protect the algorithm from sub-optimal local minima. The pseudocode is provided in Algorithm 1.

3.3 Deterministic Annealing S^{3}VM

Deterministic annealing (DA) is a global optimization heuristic that has been used to approach hard combinatorial or non-convex problems. In the context of S^{3}VMs (Sindhwani et al., 2006), it consists of relaxing the discrete label variables $\mathbf{y}_u$ to real-valued variables $\mathbf{p}_u = (p_{l+1}, \ldots, p_{l+u})$ where p_i is interpreted as the probability that $y_i = 1$. The following objective function is now considered:

$$\begin{aligned} I'(\mathbf{w}, b, \mathbf{p}_u) &= E\left[I(\mathbf{w}, b, \mathbf{y}_u)\right] \\ &= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} V(y_i, o_i) + C^\star \sum_{i=l+1}^{n} p_i V(1, o_i) + (1 - p_i) V(-1, o_i) \end{aligned} \tag{7}$$

2. Note that in this SVM training, the loss terms associated with (originally) labeled and (currently labeled) unlabeled examples are weighted by C and $C^\star$ respectively.
3. Note that in the S^{3}VM literature, this particular implementation is often referred as "TSVM" or "Transductive SVM".
4. This switching condition is slightly weaker than that proposed by Joachims (1999).

Algorithm 1 S^3VM^{light}

Train an SVM with the labeled points. $o_i \leftarrow \mathbf{w} \cdot \mathbf{x}_i + b$.
Assign $y_i \leftarrow 1$ to the ur largest o_i, -1 to the others.
$\tilde{C} \leftarrow 10^{-5} C^\star$
while $\tilde{C} < C^\star$ **do**
  repeat
    Minimize (1) with $\{y_i\}$ fixed and $C^\star$ replaced by $\tilde{C}$.
    if $\exists (i, j)$ satisfying (6) **then**
      Swap the labels y_i and y_j
    end if
  until No labels have been swapped
  $\tilde{C} \leftarrow \min(1.5C, C^\star)$
end while

where E denotes expectation under the probabilities $\mathbf{p}_u$. Note that at optimality with respect to $\mathbf{p}_u$, p_i must concentrate all its mass on $y_i = \text{sign}(\mathbf{w}^\top \mathbf{x}_i + b)$ which leads to the smaller of the two losses $V(1, o_i)$ and $V(-1, o_i)$. Hence, this relaxation step does not lead to loss of optimality and is simply a reformulation of the original objective in terms of continuous variables. In DA, an additional entropy term $-H(\mathbf{p}_u)$ is added to the objective,

$$I''(\mathbf{w}, b, \mathbf{p}_u; T) = I'(\mathbf{w}, b, \mathbf{p}_u) - TH(\mathbf{p}_u)$$
$$\text{where } H(\mathbf{p}_u) = -\sum_i p_i \log\ p_i + (1 - p_i)\ \log\ (1 - p_i),$$

and $T \geq 0$ is usually referred to as 'temperature'. Instead of (3), the following class balance constraint is used,

$$\frac{1}{u} \sum_{i=l+1}^{n} p_i = r.$$

Note that when $T = 0$, I'' reduces to (7) and the optimal $\mathbf{p}_u$ identifies the optimal $\mathbf{y}_u$. When $T = \infty$, I'' is dominated by the entropy term resulting in the maximum entropy solution ($p_i = r$ for all i). T parameterizes a family of objective functions with increasing degrees of non-convexity (see Figure 4 and further discussion below).

At any T, let $(\mathbf{w}_T, b_T, \mathbf{p}_{uT}) = \text{argmin}_{(\mathbf{w},b),\mathbf{p}_u} I''(\mathbf{w}, b, \mathbf{p}_u; T)$. This minimization can be performed in different ways:

1. *Alternating Minimization*: We sketch here the procedure proposed by Sindhwani et al. (2006). Keeping $\mathbf{p}_u$ fixed, the minimization over $(\mathbf{w}, b)$ is standard SVM training—each unlabeled example contributes two loss terms weighted by $C^\star p_i$ and $C^\star(1 - p_i)$. Keeping $(\mathbf{w}, b)$ fixed, I'' is minimized subject to the balance constraint $\frac{1}{u}\sum_{i=l+1}^{n} p_i = r$ using standard Lagrangian techniques. This leads to:

$$p_i = \frac{1}{1 + e^{(g_i - \nu)/T}} \tag{8}$$

where $g_i = C^\star[V(1, o_i) - V(-1, o_i)]$ and ν, the Lagrange multiplier associated with the balance constraint, is obtained by solving the root finding problem that arises by plugging (8) back in the balance constraint. The alternating optimization proceeds until $\mathbf{p}_u$ stabilizes in a KL-divergence sense. This method will be referred to as DA in the rest of the paper.

2. *Gradient Methods*: An alternative possibility[5] is to substitute the optimal $\mathbf{p}_u$ (8) as a function of $(\mathbf{w}, b)$ and obtain an objective function over $(\mathbf{w}, b)$ for which gradient techniques can be used:

$$\mathcal{S}(\mathbf{w}, b) := \min_{\mathbf{p}_u} I''(\mathbf{w}, b, \mathbf{p}_u; T). \tag{9}$$

$\mathcal{S}(\mathbf{w}, b)$ can be minimized by conjugate gradient descent. The gradient of $\mathcal{S}$ is easy to compute. Indeed, let us denote by $\mathbf{p}_u^*(\mathbf{w}, b)$ the argmin of (9). Then,

$$\frac{\partial \mathcal{S}}{\partial w_i} = \frac{\partial I''}{\partial w_i} + \sum_{j=l+1}^{n} \underbrace{\frac{\partial I''}{\partial p_j}\bigg|_{\mathbf{p}_u = \mathbf{p}_u^*(\mathbf{w}, b)}}_{0} \frac{\partial p_j^*(\mathbf{w})}{\partial w_i} = \frac{\partial I'(\mathbf{w}, b, \mathbf{p}_u^\star(\mathbf{w}))}{\partial w_i}.$$

The partial derivative of I'' with respect to p_j is 0 by the definition of $\mathbf{p}_u^*(\mathbf{w}, b)$. The argument goes through even in the presence of the constraint $\frac{1}{u}\sum p_i = r$; see Chapelle et al. (2002, Lemma 2) for a formal proof. In other words, we can compute the gradient of (7) with respect to $\mathbf{w}$ and consider $\mathbf{p}_u$ fixed. The same holds for b. This method will be referred to as ∇DA in the rest of the paper.

Figure 4 shows the effective loss terms in $\mathcal{S}$ associated with an unlabeled example for various values of T. In an outer loop, starting from a high value, T is decreased geometrically by a constant factor. The vector $\mathbf{p}_u$ is then tightened back close to discrete values (its entropy falls below some threshold), thus identifying a solution to the original problem. The pseudocode is provided in Algorithm 2.

Table 2 compares the DA and ∇DA solutions as $T \to 0$ at two different hyperparameter settings.[6] Because DA does alternate minimization and ∇DA does direct minimization, the solutions returned by them can be quite different. Since ∇DA is faster than DA, we only report ∇DA results in the Section 5.

Algorithm 2 DA/∇DA

Initialize $p_i = r \quad i = l+1, \ldots, n$
Set $T = 10C^\star$, $R = 1.5$, $\varepsilon = 10^{-6}$.
while $H(\mathbf{p}_{uT}) > \varepsilon$ **do**
  Solve $(\mathbf{w}_T, b_T, \mathbf{p}_{uT}) = \text{argmin}_{(\mathbf{w},b),\mathbf{p}_u} I''(\mathbf{w}, b, \mathbf{p}_u; T)$ subject to: $\frac{1}{u}\sum_{i=l+1}^{n} p_i = r$
  (find local minima starting from previous solution—alternating optimization or gradient methods can be used.)
  $T = T/R$
end while
Return $\mathbf{w}_T, b_T$

3.4 Convex Relaxation

We follow Bie and Cristianini (2006) in this section, but outline the details for the squared Hinge loss (see also Xu et al., 2004, for a similar derivation). Rewriting (1) as the familiar constrained

5. Strictly speaking, this approach is more along the lines of methods discussed in Section 4.
6. In Sindhwani et al. (2006), the best solution in the optimization path is returned.

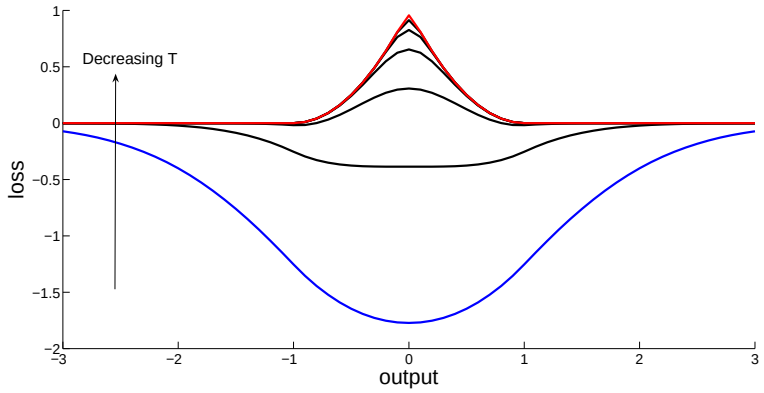


Figure 4: DA parameterizes a family of loss functions (over unlabeled examples) where the degree of non-convexity is controlled by T. As $T \to 0$, the original loss function (Figure 2) is recovered.

	DA	∇DA	DA	∇DA
g50c	6.5	7	8.3	6.7
Text	13.6	5.7	6.2	6.5
Uspst	22.5	27.2	11	11
Isolet	38.3	39.8	28.6	26.9
Coil20	3	12.3	19.2	18.9
Coil3	49.1	61.6	60.3	60.6
2moons	36.8	22.5	62.1	30

Table 2: Generalization performance (error rates) of the two DA algorithms. DA is the original algorithm (alternate optimization on $\mathbf{p}_u$ and $\mathbf{w}$). ∇DA is a direct gradient optimization on $(\mathbf{w}, b)$, where $\mathbf{p}_u$ should be considered as a function of $(\mathbf{w}, b)$. The first two columns report results when $C^\star = C$ and the last columns report results when $C^\star = C/100$. See Section 5 for more experimental details.

optimization problem of SVMs:

$$\min_{(\mathbf{w},b),\mathbf{y}_u} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i^2 + C^\star\sum_{i=l+1}^{n}\xi_i^2 \text{ subject to: } y_i o_i \geq 1-\xi_i \;\; i=1,\ldots,n.$$

Consider the associated dual problem:

$$\min_{\{y_i\}}\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K_{ij} \text{ subject to: } \sum_{i=1}^{n}\alpha_i y_i = 0, \;\; \alpha_i \geq 0$$

where $K_{ij} = \mathbf{x}_i^\top \mathbf{x}_j + D_{ij}$ and D is a diagonal matrix given by $D_{ii} = \frac{1}{2C}$, $i = 1,\ldots,l$ and $D_{ii} = \frac{1}{2C^\star}$, $i = l+1,\ldots,n$.

Introducing an $n \times n$ matrix Γ, the optimization problem can be reformulated as:

$$\min_{\Gamma} \max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j \Gamma_{ij} K_{ij} \tag{10}$$

$$\text{under constraints} \quad \sum \alpha_i y_i = 0, \;\; \alpha_i \geq 0, \;\; \Gamma = yy^\top. \tag{11}$$

The objective function (10) is now convex since it is the pointwise supremum of linear functions. However the constraint (11) is not. The idea of the relaxation is to replace the constraint $\Gamma = yy^\top$ by the following set of convex constraints:

$$\begin{aligned} &\Gamma \succeq 0, \\ &\Gamma_{ij} = y_i y_j, \;\; 1 \leq i, j \leq l, \\ &\Gamma_{ii} = 1, \;\; l+1 \leq i \leq n. \end{aligned}$$

Though the original method of Bie and Cristianini (2006) does not incorporate the class balancing constraint (3), one can additionally enforce it as $\frac{1}{u^2} \sum_{i,j=l+1}^{n} \Gamma_{ij} = (2r-1)^2$. Such a soft constraint is also used in the continuous S^{3}VM optimization methods of Section 4.

The convex problem above can be solved through Semi-Definite Programming. The labels of the unlabeled points are estimated from Γ (through one of its columns or its largest eigenvector).

This method is very expensive and scales as $O((l+u^2)^2(l+u)^{2.5})$. It is possible to try to optimize a low rank version of Γ, but the training remains slow even in that case. We therefore do not conduct empirical studies with this method.

4. Continuous Optimization

In this section we consider methods which do not include $\mathbf{y}_u$, the labels of unlabeled examples, as optimization variables, but instead solve suitably modified versions of (5) by continuous optimization techniques. We begin by discussing two issues that are common to these methods.

Balancing Constraint The balancing constraint (3) is relatively easy to enforce for all algorithms presented in Section 3. It is more difficult for algorithms covered in this section. The proposed workaround, first introduced by Chapelle and Zien (2005), is to instead enforce a linear constraint:

$$\frac{1}{u} \sum_{i=l+1}^{n} \mathbf{w}^\top \mathbf{x}_i + b = 2\tilde{r} - 1, \tag{12}$$

where $\tilde{r} = r$. The above constraint may be viewed as a "relaxation" of (3). For a given $\tilde{r}$, an easy way of enforcing (12) is to translate all the points such that the mean of the unlabeled points is the origin, that is, $\sum_{i=l+1}^{n} \mathbf{x}_i = 0$. Then, by fixing $b = 2\tilde{r} - 1$, we have an unconstrained optimization problem on $\mathbf{w}$. We will assume that the $\mathbf{x}_i$ are translated and b is fixed in this manner; so the discussion will focus on unconstrained optimization procedures for the rest of this section. In addition to being easy to implement, this linear constraint may also add some robustness against uncertainty about the true unknown class ratio in the unlabeled set (see also Chen et al., 2003, for related discussion).

However, since (12) relaxes (3),[7] the solutions found by algorithms in this section cannot strictly be compared with those in Section 3. In order to admit comparisons (this is particularly important for the empirical study in Section 5), we vary $\tilde{r}$ in an outer loop and do a dichotomic search on this value such that (3) is satisfied.

7. Note that simply setting $\tilde{r} = r$ in (12) will not enforce (3) exactly.

Primal Optimization For linear classification, the variables in $\mathbf{w}$ can be directly optimized. Nonlinear decision boundaries require the use of the "kernel trick" (Boser et al., 1992) using a kernel function $k(\mathbf{x},\mathbf{x}')$. While most of the methods of Section 3 can use a standard SVM solver as a subroutine, the methods of this section need to solve (5) with a non-convex loss function over unlabeled examples. Therefore, they cannot directly use off-the-shelf dual-based SVM software. We use one of the following primal methods to implement the techniques in this section.

Method 1 We find $\mathbf{z}_i$ such that $\mathbf{z}_i \cdot \mathbf{z}_j = k(\mathbf{x}_i,\mathbf{x}_j)$. If B is a matrix having columns $\mathbf{z}_i$, this can be written in matrix form as $B^\top B = K$. The Cholesky factor of K provides one such B. This decomposition was used for ∇S^{3}VM (Chapelle and Zien, 2005). Another possibility is to perform the eigendecomposition of K as $K = U\Lambda U^\top$ and set $B = \Lambda^{1/2}U^\top$. This latter case corresponds to the *kernel PCA map* introduced by Schölkopf and Smola (2002, Section 14.2). Once the $\mathbf{z}_i$ are found, we can simply replace $\mathbf{x}_i$ in (5) by $\mathbf{z}_i$ and solve a linear classification problem. For more details, see Chapelle et al. (2006a).

Method 2 We set $\mathbf{w} = \sum_{i=1}^n \beta_i \phi(\mathbf{x}_i)$ where ϕ denotes a higher dimensional feature map associated with the nonlinear kernel. By the Representer theorem (Schölkopf and Smola, 2002), we indeed know that the optimal solution has this form. Substituting this form in (5) and using the kernel function yields an optimization problem with β as the variables.

Note that the centering mentioned above to implement (12) corresponds to using the modified kernel Schölkopf and Smola (2002, page 431) defined by:

$$k(\mathbf{x},\mathbf{x}') := k(\mathbf{x},\mathbf{x}') - \frac{1}{u}\sum_{i=l+1}^{n} k(\mathbf{x},\mathbf{x}_i) - \frac{1}{u}\sum_{i=l+1}^{n} k(\mathbf{x}',\mathbf{x}_i) + \frac{1}{u^2}\sum_{i,j=l+1}^{n} k(\mathbf{x}_i,\mathbf{x}_j). \tag{13}$$

All the shifted kernel elements can be computed in $O(n^2)$ operations.

Finally, note that these methods are very general and can also be applied to algorithms of Section 3.

4.1 Concave Convex Procedure (CCCP)

The CCCP method (Yuille and Rangarajan, 2003) has been applied to S^{3}VMs by Fung and Mangasarian (2001), Collobert et al. (2006), and Wang et al. (2007). The description given here is close to that in Collobert et al. (2006).

CCCP essentially decomposes a non-convex function f into a convex component f_{vex} and a concave component f_{cave}. At each iteration, the concave part is replaced by a linear function (namely, the tangential approximation at the current point) and the sum of this linear function and the convex part is minimized to get the next iterate. The pseudocode is shown in Algorithm 3.

Algorithm 3 CCCP for minimizing $f = f_{vex} + f_{cave}$

Require: Starting point $\mathbf{x}_0$
  $t \leftarrow 0$
  while $\nabla f(\mathbf{x}_t) \neq 0$ **do**
    $\mathbf{x}_{t+1} \leftarrow \arg\min_{\mathbf{x}} f_{vex}(\mathbf{x}) + \nabla f_{cave}(\mathbf{x}_t) \cdot \mathbf{x}$
    $t \leftarrow t+1$
  end while

In the case of S^{3}VM, the first two terms in (5) are convex. Splitting the last non-convex term corresponding to the unlabeled part as the sum of a convex and a concave function, we have:

$$\max(0,1-|t|)^2 = \underbrace{\max(0,1-|t|)^2+2|t|}_{\text{convex}} \underbrace{-2|t|}_{\text{concave}}.$$

If an unlabeled point is currently classified positive, then at the next iteration, the effective (convex) loss on this point will be

$$\tilde{L}(t) = \begin{cases} 0 & \text{if } t \geq 1, \\ (1-t)^2 & \text{if } |t| < 1, \\ -4t & \text{if } t \leq -1. \end{cases}$$

A corresponding $\tilde{L}$ can be defined for the case of an unlabeled point being classified negative. The CCCP algorithm specialized to S^{3}VMs is given in Algorithm 4. For optimization variables we employ method 1 given at the beginning of this section.

Algorithm 4 CCCP for S^{3}VMs

Starting point: Use the $\mathbf{w}$ obtained from the supervised SVM solution.
repeat
$y_i \leftarrow \text{sign}(\mathbf{w}\cdot\mathbf{x}_i+b),\ \ l+1 \leq i \leq n.$
$(\mathbf{w},b) = \arg\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\max(0,1-y_i(\mathbf{w}\cdot\mathbf{x}_i+b))^2 + C^\star\sum_{i=l+1}^{n}\tilde{L}(y_i(\mathbf{w}\cdot\mathbf{x}_i+b)).$
until convergence of $y_i,\ l+1 \leq i \leq n.$

The CCCP method given in Collobert et al. (2006) does not use annealing, that is, increasing $C^\star$ slowly in steps as in S^{3}VM^{light} to help reduce local minima problems. We have however found performance improvements with annealing (see Table 12).

4.2 ∇S^{3}VM

This method is proposed by Chapelle and Zien (2005) to minimize directly the objective function (5) by gradient descent. For optimization variables, method 1 given at the beginning of this section is used. Since the function $t \mapsto \max(0,1-|t|)^2$ is not differentiable, it is replaced by $t \mapsto \exp(-st^2)$, with $s = 5$ (see Figure 5), to get the following smooth optimization problem:[8]

$$\min_{\mathbf{w},b}\ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\max(0,1-y_i(\mathbf{w}\cdot\mathbf{x}_i+b))^2 + C^\star\sum_{i=l+1}^{n}\exp(-s(\mathbf{w}\cdot\mathbf{x}_i+b)^2). \qquad (14)$$

As for S^{3}VM^{light}, ∇S^{3}VM performs annealing in an outer loop on $C^\star$. In the experiments we followed the same annealing schedule as in Chapelle and Zien (2005): $C^\star$ is increased in 10 steps to its final value. More precisely, at the ith step, $C^\star$ is set to $2^{i-10}C^\star_{final}$.

4.3 Continuation S^{3}VM (cS^{3}VM)

Closely related to ∇S^{3}VM, Chapelle et al. (2006a) proposes a *continuation* method for minimizing (14). Gradient descent is performed on the same objective function (with the same loss for the

8. Chapelle and Zien (2005) used $s = 3$ with hinge loss, $p = 1$ in (2), but $s = 5$ seems to be a better choice for quadratic hinge loss ($p = 2$).

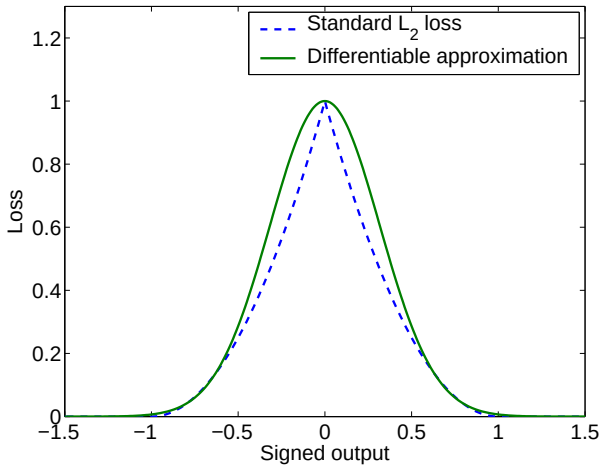


Figure 5: The loss function on the unlabeled points $t \mapsto \max(0, 1-|t|)^2$ is replaced by a differentiable approximation $t \mapsto \exp(-5t^2)$.

unlabeled points as shown in Figure 5), but the method used for annealing is different. Instead of slowly increasing $C^\star$, it is kept fixed, and a continuation technique is used to transform the objective function.

This kind of method belongs to the field of global optimization techniques (Wu, 1996). The idea is similar to deterministic annealing (see Figure 6). A smoothed version of the objective function is first minimized. With enough smoothing the global minimum can hopefully be easily found. Then the smoothing is decreased in steps and the minimum is tracked—the solution found in one step serves as the starting point for the next step. The method is continued until there is no smoothing and so we get back to the solution of (14). Algorithm 5 gives an instantiation of the method in which smoothing is achieved by convolution with a Gaussian, but other smoothing functions can also be used.

Algorithm 5 Continuation method for solving $\min_{\mathbf{x}} f(\mathbf{x})$

Require: Function $f : \mathbb{R}^d \mapsto \mathbb{R}$, initial point $\mathbf{x}_0 \in \mathbb{R}^d$
Require: Sequence $\gamma_0 > \gamma_1 > \ldots \gamma_{p-1} > \gamma_p = 0$.
  Let $f_\gamma(\mathbf{x}) = (\pi\gamma)^{-d/2} \int f(\mathbf{x}-\mathbf{t}) \exp(-\|\mathbf{t}\|^2/\gamma) d\mathbf{t}$.
  for $i = 0$ to p **do**
    Starting from $\mathbf{x}_i$, find local minimizer $\mathbf{x}_{i+1}$ of f_{γ_i}.
  end for

It is not clear if one should only smooth the last non-convex term of (14) (the first two terms are convex) or the whole objective as in Chapelle et al. (2006a). It is noteworthy that since the loss for the unlabeled points is bounded, its convolution with a Gaussian of infinite width tends to the zero function. In other words, with enough smoothing, the unlabeled part of the objective function vanishes and the optimization is identical to a standard SVM.

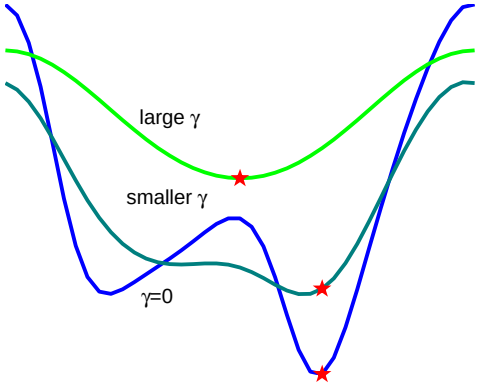


Figure 6: Illustration of the continuation method: the original objective function in blue has two local minima. By smoothing it, we find one global minimum (the red star on the green curve). By reducing the smoothing, the minimum moves toward the global minimum of the original function.

4.4 Newton S^{3}VM

One difficulty with the methods described in sections 4.2 and 4.3 is that their complexity scales as $O(n^3)$ because they employ an unlabeled loss function that does not have a linear part, for example, see (14). Compared to a method like S^{3}VM^{light} (see Section 3.2), which typically scales as $O(n_{\mathsf{sv}}^3 + n^2)$, this can make a large difference in efficiency when n_{sv} (the number of support vectors) is small. In this subsection we propose a new loss function for unlabeled points and an associated Newton method (along the lines of Chapelle, 2007) which brings down the $O(n^3)$ complexity of the ∇S^{3}VM method.

To make efficiency gains we employ method 2 described at the beginning of this section (note that method 1 requires an $O(n^3)$ preprocessing step) and perform the minimization on β, where $\mathbf{w} = \sum_{i=1}^{n} \beta_i \phi(\mathbf{x}_i)$. Note that the β_i are expansion coefficients and not the Lagrange multipliers α_i in standard SVMs. Let us consider general loss functions, ℓ_L for the labeled points, ℓ_U for the unlabeled points, replace $\mathbf{w}$ by β as the variables in (5), and get the following optimization problem,

$$\min_{\beta} \frac{1}{2}\beta^\top K\beta + C\sum_{i=1}^{l} \ell_L(y_i(K_i^\top \beta + b)) + C^\star \sum_{i=l+1}^{n} \ell_U(K_i^\top \beta + b), \tag{15}$$

where K is the kernel matrix with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and K_i is the i^{th} column of K.[9]

As we will see in detail below, computational time is dictated by n_{sv}, the number of points that lie in the domain of the loss function where curvature is non-zero. With this motivation we choose the differentiable loss function plotted in Figure 7 having several linear and flat parts which are smoothly connected by small quadratic components.[10]

9. Note that, the kernel elements used here correspond to the modified kernel elements in (13).
10. The CCCP method of Collobert et al. (2006) also considers a loss function with a flat middle part, but it was not motivated by computational gains.

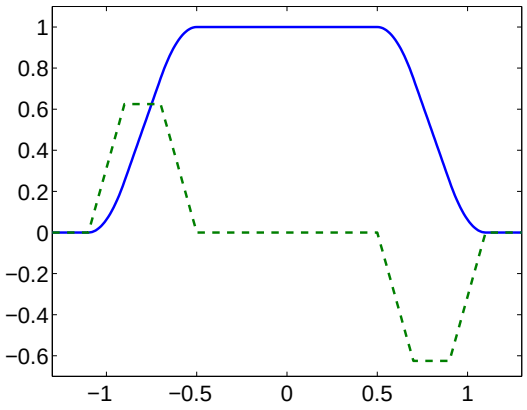


Figure 7: The piecewise quadratic loss function ℓ_U and its derivative (divided by 4; dashed line).

Consider the solution of (15) with this choice of loss function. The gradient of (15) is

$$K\mathbf{g} \text{ with } g_i = \begin{cases} \beta_i + C\ell'_L(y_i(K_i^\top \beta + b))y_i & 1 \le i \le l \\ \beta_i + C^\star \ell'_U(K_i^\top \beta + b) & l+1 \le i \le n \end{cases}. \tag{16}$$

Using a gradient based method like nonlinear conjugate gradient would sill be costly because each evaluation of the gradient requires $O(n^2)$ effort. To improve the complexity when n_{sv} is small, one can use Newton's method instead. Let us now go into these details.

The Hessian of (15) is

$$K + KDK, \text{ with } D \text{ diagonal, } D_{ii} = \begin{cases} C\ell''_L(y_i(K_i^\top \beta + b)) & 1 \le i \le l \\ C^\star \ell''_U(K_i^\top \beta + b) & l+1 \le i \le n \end{cases}. \tag{17}$$

The corresponding Newton update is $\beta \leftarrow \beta - (K + KDK)^{-1} K\mathbf{g}$. The advantage of Newton optimization on this problem is that the step can be computed in $O(n_{\mathsf{sv}}^3 + n^2)$ time[11] as we will see below (see also Chapelle, 2007, for a similar derivation). The number of Newton steps required is usually a small finite constant.

A problem in performing a Newton optimization with a non-convex optimization function is that the step might not be a descent direction because the Hessian is not necessarily positive definite. To avoid this problem, we use the Levenberg-Marquardt algorithm (Fletcher, 1987, Algorithm 5.2.7). Roughly speaking, this algorithm is the same as Newton minimization, but a large enough ridge is added to the Hessian such that it becomes positive definite. For computational reasons, instead of adding a ridge to the Hessian, we will add a constant times K.

The goal is to choose a $\lambda \ge 1$ such that $\lambda K + KDK$ is positive definite and solve $(\lambda K + KDK)^{-1} K\mathbf{g}$ efficiently. For this purpose, we reorder the points such that $D_{ii} \neq 0,\ i \le n_{\mathsf{sv}}$ and $D_{ii} = 0,\ i > n_{\mathsf{sv}}$. Let A be the Cholesky decomposition of K:[12] A is the upper triangular matrix satisfying $A^\top A = K$. We suppose that K (and thus A) is invertible. Let us write

$$\lambda K + KDK = A^\top(\lambda I_n + ADA^\top)A.$$

11. Consistent with the way we defined earlier, note here that n_{sv} is the number of "support vectors" where a support vector is defined as a point $\mathbf{x}_i$ such that $D_{ii} \neq 0$.
12. As we will see below, we will not need the Cholesky decomposition of K but only that of of K_{sv}.

The structure of K and D implies that

$$\lambda K + KDK \succ 0 \Leftrightarrow B := \lambda I_{n_{\mathsf{sv}}} + A_{\mathsf{sv}} D_{\mathsf{sv}} A_{\mathsf{sv}}^\top \succ 0,$$

where A_{sv} is the Cholesky decomposition of K_{sv} and K_{sv} is the matrix formed using the first n_{sv} rows and columns of K. After some block matrix algebra, we can also get the step as

$$-(\lambda K + KDK)^{-1} K\mathbf{g} = \begin{pmatrix} A_{\mathsf{sv}}^{-1} B^{-1} A_{\mathsf{sv}} (\mathbf{g}_{\mathsf{sv}} - \frac{1}{\lambda} D_{\mathsf{sv}} K_{\mathsf{sv},\mathsf{nsv}} \mathbf{g}_{\mathsf{nsv}}) \\ \frac{1}{\lambda} \mathbf{g}_{\mathsf{nsv}} \end{pmatrix}, \tag{18}$$

where nsv refers to the indices of the "non support vectors", that is, $\{i,\ D_{ii} = 0\}$. Computing this direction takes $O(n_{\mathsf{sv}}^3 + n^2)$ operations. The checking of the positive definiteness of B can be done by doing Cholesky decomposition of K_{sv}. This decomposition can then be reused to solve the linear system involving B. Full details, following the ideas in Fletcher (1987, Algorithm 5.2.7), are given in Algorithm 6.

Algorithm 6 Levenberg-Marquardt method

$\beta \leftarrow 0$.
$\lambda \leftarrow 1$.
repeat
  Compute $\mathbf{g}$ and D using (16) and (17)
  $\mathsf{sv} \leftarrow \{i,\ D_{ii} \neq 0\}$ and $\mathsf{nsv} \leftarrow \{i,\ D_{ii} = 0\}$.
  $A_{\mathsf{sv}} \leftarrow$ Cholesky decomposition of K_{sv}.
  Do the Cholesky decomposition of $\lambda I_{n_{\mathsf{sv}}} + A_{\mathsf{sv}} D_{\mathsf{sv}} A_{\mathsf{sv}}^\top$. If it fails, $\lambda \leftarrow 4\lambda$ and try again.
  Compute the step $\mathbf{s}$ as given by (18).
  $\rho \leftarrow \frac{\Omega(\beta+\mathbf{s}) - \Omega(\beta)}{\frac{1}{2}\mathbf{s}^\top (K+KDK)\mathbf{s} + \mathbf{s}^\top K\mathbf{g}}$. % If the obj fun Ω were quadratic, ρ would be 1.
  If $\rho > 0$, $\beta \leftarrow \beta + s$.
  If $\rho < 0.25$, $\lambda \leftarrow 4\lambda$.
  If $\rho > 0.75$, $\lambda \leftarrow \min(1, \frac{\lambda}{2})$.
until $\mathrm{Norm}(\mathbf{g}) \leq \varepsilon$

As discussed above, the flat part in the loss (cf. Figure 7) provides computational value by reducing n_{sv}. But we feel it may also possibly help in leading to better local minimum. Take for instance a Gaussian kernel and consider an unlabeled point far from the labeled points. At the beginning of the optimization, the output on that point will be 0.[13] This unlabeled point does not contribute to pushing the decision boundary one way or the other. This seems like a satisfactory behavior: it is better to wait to have more information before taking a decision on an unlabeled point for which we are unsure. In Table 3 we compare the performance of the flat loss in Figure 7 and the original quadratic loss used in (5). The flat loss yields a huge gain in performance on `2moons`. On the other data sets the two losses perform somewhat similarly. From a computational point of view, the flat part in the loss can sometimes reduce the training time by a factor 10 as shown in Table 3.

5. Experiments

This section is organized around a set of empirical issues:

13. This is true only for balanced problems; otherwise, the output is b.

	Error rate		Training time	
	Flat	Quadratic	Flat	Quadratic
g50c	6.1	5.3	15	39
Text	5.4	7.7	2180	2165
Uspst	18.6	15.9	233	2152
Isolet	32.2	27.1	168	1253
Coil20	24.1	24.7	152	1244
Coil3	61.5	58.4	6	8
2moons	11	66.4	1.7	1.2

Table 3: Comparison of the Newton-S^3VM method with two different losses: the one with a flat part in the middle (see Figure 7) and the standard quadratic loss (Figure 2). Left: error rates on the unlabeled set; right: average training time in seconds for one split and one binary classifier training (with annealing and dichotomic search on the threshold as explained in the experimental section). The implementations have not been optimized, so the training times only constitute an estimate of the relative speeds.

1. While S^3VMs have been very successful for text classification (Joachims, 1999), there are many data sets where they do not return state-of-the-art empirical performance (Chapelle and Zien, 2005). This performance variability is conjectured to be due to local minima problems. In Section 5.3, we discuss the suitability of the S^3VM objective function for semi-supervised learning. In particular, we benchmark current S^3VM implementations against the exact, globally optimal solution and we discuss whether one can expect significant improvements in generalization performance by better approaching the global solution.

2. Several factors influence the performance and behavior of S^3VM algorithms. In Section 5.4 we study their quality of optimization, generalization performance, sensitivity to hyperparameters, effect of annealing and the robustness to uncertainty in class ratio estimates.

3. S^3VMs were originally motivated by Transductive learning, the problem of estimating labels of unlabeled examples without necessarily producing a decision function over the entire input space. However, S^3VMs are also semi-supervised learners as they are able to handle unseen test instances. In Section 5.5, we run S^3VM algorithms in an inductive mode and analyze performance differences between unlabeled and test examples.

4. There is empirical evidence that S^3VMs exhibit poor performances on "manifold" type data (where graph-based methods typically excel) or when the data has many distinct sub-clusters (Chapelle and Zien, 2005). We explore the issue of data geometry and S^3VM performance in Section 5.6.

At the outset, we point out that this section does not provide an exhaustive cross-comparison between algorithms. Such a comparison would require, say, cross-validation over multiple hyperparameters, randomization over choices of labels, dichotomic search to neutralize balance constraint differences and handling different choices of annealing sequences. This is computationally quite demanding and, more seriously, statistically brittle due to the lack of labeled validation data in

semi-supervised tasks. Our goal, therefore, is not so much to establish a ranking of algorithms reviewed in this paper, but rather to observe their behavior under a neutral experimental protocol.

We next describe the data sets used in our experiments. Note that our choice of data sets is biased towards multi-class manifold-like problems which are particularly challenging for S^3VMs. Because of this choice, the experimental results do not show the typically large improvements one might expect over standard SVMs. We caution the reader not to draw the conclusion that S^3VM is a weak algorithm in general, but that it often does not return state-of-the-art performance on problems of this nature.

5.1 Data Sets

Most data sets come from Chapelle and Zien (2005). They are summarized in Table 4.

data set	classes	dims	points	labeled
g50c	2	50	550	50
Text	2	7511	1946	50
Uspst	10	256	2007	50
Isolet	9	617	1620	50
Coil20	20	1024	1440	40
Coil3	3	1024	216	6
2moons	2	102	200	2

Table 4: Basic properties of benchmark data sets.

The artificial data set g50c is inspired by Bengio and Grandvalet (2004): examples are generated from two standard normal multi-variate Gaussians, the labels correspond to the Gaussians, and the means are located in 50-dimensional space such that the Bayes error is 5%. The real world data sets consist of two-class and multi-class problems. The Text data set is defined using the classes mac and mswindows of the Newsgroup20 data set preprocessed as in Szummer and Jaakkola (2001). The Uspst set contains the test data part of the well-known USPS data on handwritten digit recognition. The Isolet is a subset of the *ISOLET spoken letter database* (Cole et al., 1990) containing the speaker sets 1, 2 and 3 and 9 confusing letters {B,C,D,E,G,P,T,V,Z}. In Coil20 (respectively Coil3), the data are gray-scale images of 20 (respectively 3) different objects taken from different angles, in steps of 5 degrees (Nene et al., 1996). The Coil3 data set has been used first by Chapelle et al. (2006c) and is particularly difficult since the 3 classes are 3 cars which look alike (see Figure 8).

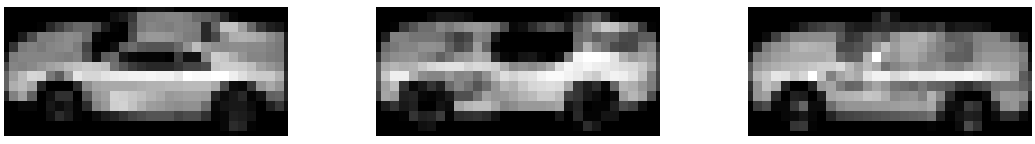


Figure 8: The 3 cars from the COIL data set, subsampled to 32×32

Finally, `2moons` has been used extensively in the semi-supervised learning literature (see for instance Zhu and Ghahramani, 2002, and Figure 1). For this data set, the labeled points are fixed and new unlabeled points are randomly generated for each repetition of the experiment.

5.2 Experimental Setup

To minimize the influence of external factors, unless stated otherwise, the experiments were run in the following normalized way:

Hyperparameters The Gaussian kernel $k(\mathbf{x},\mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2)$ was used. For simplicity the constant $C^\star$ in (1) was set to C.[14] The same hyperparameters C and σ have been used for all the methods. They are found with cross-validation by training an inductive SVM on the entire data set (the unlabeled points being assigned their real label). These values are reported in Table 5. Even though it seems fair to compare the algorithms with the same hyperparameters, there is a possibility that some regime of hyperparameter settings is more suitable for a particular algorithm than for another. We discuss this issue in section 5.4.3.

Multi-class Data sets with more than two classes are learned with a one-versus-the-rest approach. The reported objective value is the mean of the objective values of the different classifiers. We also conducted experiments on pair-wise binary classification problems (cf. Section 5.6.1) constructed from the multi-class data sets.

Objective value Even though the objective function that we want to minimize is (5), some algorithms like ∇S^{3}VM use another (differentiable) objective function. In order to compare the objective values of the different algorithms, we do the following: after training, we predict the labels of the unlabeled points and train a standard SVM on this augmented labeled set (with $p = 2$ in (2) and C, $C^\star$ weightings on originally labeled and unlabeled terms respectively). The objective value reported is the one of this SVM.

Balancing constraint For the sake of simplicity, we set r in the balancing constraint (3) to the true ratio of the positive points in the unlabeled set. This constraint is relatively easy to enforce for all algorithms presented in Section 3. It is more difficult for algorithms in Section 4 and, for them we used the dichotomic search described at the beginning of Section 4.

For a given data set, we randomly split the data into a labeled set and an unlabeled set. We refer to the error rate on the unlabeled set as the *unlabeled error* to differentiate it from the *test error* which would be computed on an unseen test set. Results are averaged over 10 random splits. The difference between unlabeled and test performance is discussed in Section 5.5.

5.3 Suitability of the S^{3}VM Objective Function

Table 1 shows unlabeled error rates for common S^{3}VM implementations on two small data sets `Coil3` and `2moons`. On these data sets, we are able to also run Branch-and-Bound and get the true globally optimal solution. We see that the global optimum corresponds to a perfect solution, while the local minima found by approximate implementations yield very poor accuracies. From these results, it appears that the minimization of the S^{3}VM objective function makes good sense, even

14. Alternatively, one could set $C^\star = C\frac{l}{u}$ to have equal contribution from labeled and unlabeled points.

	σ	C
g50c	38	19
Text	3.5	31
Uspst	7.4	38
Isolet	15	43
Coil20	2900	37
Coil3	3000	100
2moons	0.5	10

Table 5: Values of the hyperparameters used in the experiments.

though the performance of practical S^3VM may not consistently reflect this due to local minima problems.

Table 6 records the rank correlation between unlabeled error and objective function. The rank correlation has been computed in the following way. For each split, we take 10 different solutions and compute the associated unlabeled error and objective value. Ideally, we would like to sample these solutions at random around local minima. But since it is not obvious how to do such a sampling, we simply took the solution given by the different S^3VM algorithms as well as 4 "intermediate" solutions obtained as follows. A standard SVM is trained on the original labeled set and a fraction of the unlabeled set (with their true labels). The fraction was either 0, 10, 20 or 30%. The labels of the remaining unlabeled points are assigned through a thresholding of the real value outputs of the SVM. This threshold is such that the balancing constraint (3) is satisfied. Finally, an SVM is retrained using the entire training set. By doing so, we "sample" solutions varying from an inductive SVM trained on only the labeled set to the optimal solution. Table 6 provides evidence that the unlabeled error is correlated with the objective values.

	Coefficient
g50c	0.2
Text	0.67
Uspst	0.24
Isolet	0.23
Coil20	0.4
Coil3	0.17
2moons	0.45

Table 6: Kendall's rank correlation (Abdi, 2006) between the unlabeled error and the objective function averaged over the 10 splits (see text for details).

5.4 Behavior of S^3VM Algorithms

Several factors influence the performance and behavior of S^3VM algorithms. We study their quality of optimization, generalization performance, sensitivity to hyperparameters, effect of annealing and the robustness to uncertainty in class ratio estimates.

5.4.1 QUALITY OF MINIMIZATION

In Table 7 we compare different algorithms in terms of minimization of the objective function. ∇S^{3}VM and cS^{3}VM appear clearly to be the methods achieving the lowest objective values. However, as we will see in the next section, this does not necessarily translate into lower unlabeled errors.

∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton
1.7	1.9	4.5	4.9	4.3	3.7

Table 7: For each data set and each split, the algorithms were ranked according to the objective function value they attained. This table shows the average ranks. These ranks are only about objective function values; error rates are discussed below.

5.4.2 QUALITY OF GENERALIZATION

Table 8 reports the unlabeled errors of the different algorithms on our benchmark data sets. A first observation is that most algorithms perform quite well on `g50c` and `Text`. However, the unlabeled errors on the other data sets, `Uspst`, `Isolet`, `Coil20`, `Coil3`, `2moons` (see also Table 1) are poor and sometimes even worse than a standard SVM. As pointed out earlier, these data sets are particularly challenging for S^{3}VMs. Moreover, the honors are divided and no algorithm clearly outperforms the others. We therefore cannot give a recommendation on which one to use.

	∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton	SVM	SVM-5cv
`g50c`	6.7	6.4	6.3	6.2	7	6.1	8.2	4.9
`Text`	5.1	5.3	8.3	8.1	5.7	5.4	14.8	2.8
`Uspst`	15.6	36.2	16.4	15.5	27.2	18.6	20.7	3.9
`Isolet`	25.8	59.8	26.7	30	39.8	32.2	32.7	6.4
`Coil20`	25.6	30.7	26.6	25.3	12.3	24.1	24.1	0

Table 8: Unlabeled errors of the different S^{3}VMs implementations. The next to the last column is an SVM trained only on the labeled data, while the last column reports 5 fold cross-validation results of an SVM trained on the whole data set using the labels of the unlabeled points. The values in these two columns can be taken as upper and lower bounds on the best achievable error rates. See Table 1 for results on `Coil3` and `2moons`.

Note that these results may differ from the ones previously reported by Chapelle and Zien (2005), Collobert et al. (2006), Sindhwani et al. (2006), and Chapelle et al. (2006a) on the same data sets. Most of this difference comes from the choice of the hyperparameters. Indeed, as explained below, several algorithms are rather sensitive to the choice of hyperparameters. The exact experimental setting is also different. In results reported elsewhere, r is often estimated from the labeled set and the constraint is often a "soft" one. In Table 8, the hard balance constraint is enforced for all methods assuming r to be known exactly. Finally, in Chapelle et al. (2006a), only pair-wise

binary problems were considered for the cS^{3}VM algorithm, while results in Table 8 for multiclass data sets are obtained under a one-versus-the-rest setup.

5.4.3 SENSITIVITY TO HYPERPARAMETERS

As mentioned before, it is possible that different algorithms excel in different hyperparameter regimes. This is more likely to happen due to better local minima at different hyperparameters as opposed to a better global solution (in terms of error rates).

Due to computational constraints, instead of setting the three hyperparameters ($C, C^\star$ and the Gaussian kernel width, σ) by cross-validation for each of the methods, we explored the influence of these hyperparameters on one split of the data. More specifically, Table 9 shows the *relative* improvement (in %) that one can gain by selecting other hyperparameters. These numbers may be seen as a measure of the *robustness* of a method. Note that these results have to be interpreted carefully because they are only on one split: for instance, it is possible that"by chance" the method did not get stuck in a bad local minimum for one of the hyperparameter settings, leading to a larger number in Table 9.

	∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton
g50c	31.2	31.2	27.8	13.3	35	7.7
Text	22	7.1	29.2	19.9	34.4	1.1
Uspst	12	70.2	19.2	0	41	15.6
Isolet	7.6	57.6	8	0	4.9	2.6
Coil20	46.4	42.7	27.9	5.8	5.7	16.9
Coil3	32.6	39.2	15.3	20.2	5.8	24.6
2moons	45.6	50	54.1	13.5	23.5	0
Mean	28.2	42.6	25.9	10.4	21.5	9.8

Table 9: On the 1st split of each data set, 27 set of hyperparameters $(\sigma', C', C^{\star\prime})$ have been tested from $\sigma' \in \{\frac{1}{2}\sigma, \sigma, 2\sigma\}, C' \in \{\frac{1}{10}C, C, 10C\}, C^{\star\prime} \in \{\frac{1}{100}C', \frac{1}{10}C', C'\}$. The table shows the *relative* improvement (in %) by taking the best hyperparameters over default ones.

By measuring the variation of the unlabeled errors with respect to the choice of the hyperparameters, Table 10 records an indication of the sensitivity of the method with respect to that choice. From this point of view S^{3}VM^{light} appears to be the most stable algorithm.

∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton
6.8	8.5	6.5	2.7	8.4	8.7

Table 10: The variance of the unlabeled errors have been averaged over the 27 possible hyperparameters (cf. Table 9). The table shows those variance averaged over the data sets. A small number shows that a given method is not too sensitive to the choice of the hyperparameters.

Finally, from the experiments in Table 9, we observed that in some cases a smaller value of $C^\star$ is helpful. We have thus rerun the algorithms on all the splits with $C^\star$ divided by 100: see Table 11. The overall performance does not necessarily improve, but the very poor results become better (see for instance `Uspst,Isolet,Coil20` for cS^{3}VM).

	∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton
`g50c`	8.3	8.3	8.5	8.4	6.7	7.5
`Text`	5.7	5.8	8.5	8.1	6.5	14.5
`Uspst`	14.1	15.6	14.9	14.5	11	19.2
`Isolet`	27.8	28.5	25.3	29.1	26.9	32.1
`Coil20`	23.9	23.6	23.6	21.8	18.9	24.6
`Coil3`	60.8	55.0	56.3	59.2	60.6	60.5
`2moons`	65.0	49.8	66.3	68.7	30	33.5

Table 11: Same as Table 8, but with $C^\star$ divided by 100.

5.4.4 EFFECT OF ANNEALING SCHEDULE

All the algorithms described in this paper use some sort of annealing (e.g., gradually decreasing T in DA or increasing $C^\star$ in S^{3}VM^{light} in an outer loop) where the role of the unlabeled points is progressively increased. The three ingredients to fix are:

1. The starting point which is usually chosen in such a way that the unlabeled points have very little influence and the problem is thus almost convex.
2. The stopping criterion which should be such that the annealed objective function and the original objective function are very close.
3. The number of steps. Ideally, one would like to have as many steps as possible, but for computational reasons the number of steps is limited.

In the experimental results presented in Figure 9, we only varied the number of steps. The starting and final values are as indicated in the description of the algorithms. The original CCCP paper did not have annealing and we used the same scheme as for ∇S^{3}VM: $C^\star$ is increased exponentially from $10^{-3}C$ to C. For DA and ∇DA, the final temperature is fixed at a small constant and the decrease rate R is such that we have the desired number of steps. For all algorithms, one step means that there is no annealing.

One has to be cautious when drawing conclusion from this plot. Indeed, the results are only for one data set, one split and fixed hyperparameters. The goal is to get an idea of whether the annealing for a given method is useful; and if so, how many steps should be taken. From this plot, it seems that:

- All methods, except Newton, seem to benefit from annealing. However, on some other data sets, annealing improved the performances of Newton's method (results not shown).
- Most methods do not require a lot of steps. More precisely, we have noticed that the number of steps does not matter as much as the minimization around a "critical" value of the annealing parameter; if that point is missed, then the results can be bad.

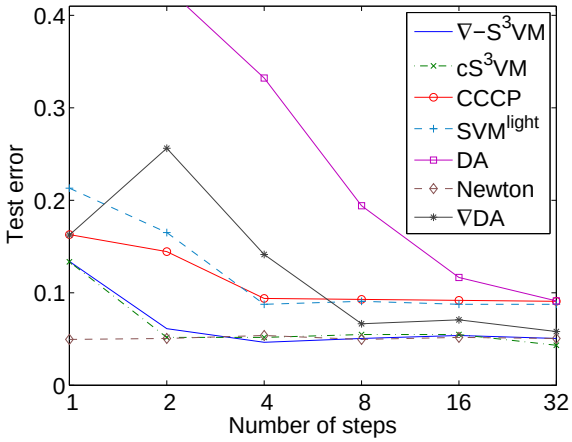


Figure 9: Influence of the number of annealing steps on the first split of `Text`.

	No annealing	Annealing
`g50c`	7.9	6.3
`Text`	14.7	8.3
`Uspst`	21.3	16.4
`Isolet`	32.4	26.7
`Coil20`	26.1	26.6
`Coil3`	49.3	56.6
`2moons`	67.1	63.1

Table 12: Performance of CCCP with and without annealing. The annealing schedule is the same as the one used for ∇S^{3}VM and Newton: $C^\star$ is increased in 10 steps from its final value divided by 1000.

- DA seems the method which relies the most on a relatively slow annealing schedule, while ∇DA can have a faster annealing schedule.

- CCCP was originally proposed without annealing, but it seems that its performance can be improved by annealing. To confirm this fact, Table 12 compares the results of CCCP with and without annealing on the 10 splits of all the data sets.

The results provided in Table 8 are with annealing for all methods.

5.4.5 ROBUSTNESS TO CLASS RATIO ESTIMATE

Several results reported in the previous tables are better than, for instance, the results in Chapelle and Zien (2005); Chapelle et al. (2006a). This is because (a) we took into account the knowledge of the true class ratio among the unlabeled examples, and (b) we enforced the constraint (3) exactly (with the dichotomic search described at the beginning of Section 4).

Of course, in practice the true number of positive points is unknown. One can estimate it from the labeled points as:

$$r = \frac{1}{2}\left(\frac{1}{l}\sum_{i=1}^{l} y_i + 1\right).$$

Table 13 presents the generalization performances in this more realistic context where class ratios are estimated as above and original soft balance constraint is used where applicable (recall the discussion in Section 4).

	∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton	SVM
g50c	7.2	6.6	6.7	7.5	8.4	5.8	9.1
Text	6.8	5	12.8	9.2	8.1	6.1	23.1
Uspst	24.1	41.5	24.3	24.4	29.8	25	24.2
Isolet	48.4	58.3	43.8	36	46	45.5	38.4
Coil20	35.4	51.5	34.5	25.3	12.3	25.4	26.2
Coil3	64.4	59.7	59.4	56.7	61.7	62.9	51.8
2moons	62.2	33.7	55.6	68.8	22.5	8.9	44.4

Table 13: Constant r estimated from the labeled set. For methods of Section 4, the original constraint (12) is used; there is no dichotomic search (see beginning of Section 4).

5.5 Transductive Versus Semi-supervised Learning

S^{3}VMs were introduced as *Transductive* SVMs, originally designed for the task of directly estimating labels of unlabeled points. However S^{3}VMs provide a decision boundary in the entire input space, and so they can provide labels of unseen test points as well. For this reason, we believe that S^{3}VMs are inductive semi-supervised methods and not strictly transductive. A discussion on the differences between semi-supervised learning and transduction can be found in Chapelle et al. (2006b, Chapter 25).[15]

We expect S^{3}VMs to perform equally well on the unlabeled set and on an unseen test set. To test this hypothesis, we took out 25% of the unlabeled set that we used as a unseen test set. We performed 420 experiments (6 algorithms, 7 data sets and 10 splits). Based on these 420 pairs of error rates, we did not observe a significant difference at the 5% confidence level. Also, for each of the 7 data sets (resp 6 algorithms), there was no statistical significant differences in the 60 (resp 70) pairs of error rates.

Similar experiments were performed by Collobert et al. (2006, Section 6.2). Based on 10 splits, the error rate was found to be better on the unlabeled set than on the test set. The authors made the hypothesis that when the test and training data are not identically distributed, transduction can be helpful. Indeed, because of small sample effect, the unlabeled and test set could appear as not coming from the same distribution (this is even more likely in high dimension).

We considered the g50c data set and biased the split between unlabeled and test set such that there is an angle between the principal directions of the two sets. Figure 10 shows a correlation

15. Paper is available at http://www.kyb.tuebingen.mpg.de/ssl-book/discussion.pdf.

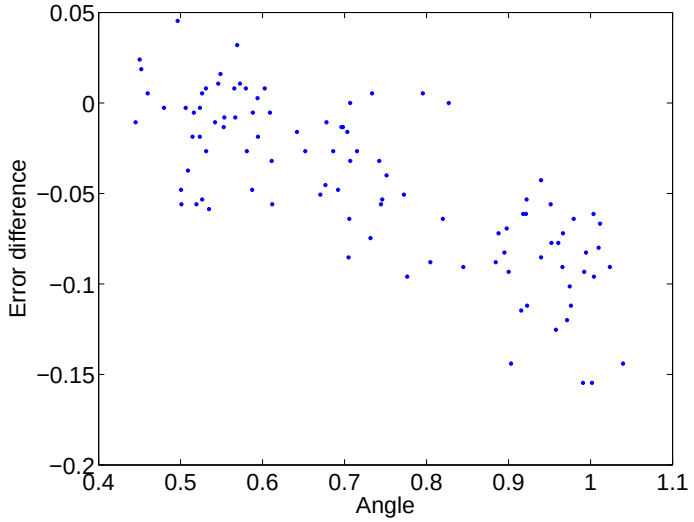


Figure 10: The test set of g50c has been chosen with a bias such that there is an angle between the principal directions of the unlabeled and test sets. The figure shows the difference in error rates (negative means better error rate on the unlabeled set) as a function of the angle for different random (biased) splits.

between the difference in error rates and this angle: the error on the test set deteriorates as the angle increases. This confirms the hypothesis stated above.

5.6 Role of Data Geometry

There is empirical evidence that S^3VMs exhibit poor performances on "manifold" type data or when the data has many distinct sub-clusters (Chapelle and Zien, 2005). We now explore this issue and propose an hybrid method combining the S^3VM and LapSVM (Sindhwani et al., 2005).

5.6.1 EFFECT OF MULTIPLE CLUSTERS

In Chapelle et al. (2006a), cS^3VM exhibited poor performance in multiclass problems with one-versus-the-rest training, but worked well on pairwise binary problems that were constructed (using all the labels) from the same multiclass data sets. Note that in practice it is not possible to do semi-supervised one-vs-one multiclass training because the labels of the unlabeled points are truly unknown.

We compared different algorithms on pairwise binary classification tasks for all the multiclass problems. Results are shown in the first 4 rows of Table 14. Except on Coil3, most S^3VM algorithms show improved performances. There are two candidate explanations for this behavior:

1. The binary classification problems in one-versus-the-rest are unbalanced and this creates difficulties for S^3VMs.

	∇S^{3}VM	cS^{3}VM	CCCP	S^{3}VM^{light}	∇DA	Newton	SVM	SVM-5cv
Uspst	1.9	2	2.8	2.9	4	1.9	5.3	0.9
Isolet	4.8	11.8	5.5	5.7	5	6.3	7.3	1.2
Coil20	2.8	3.9	2.9	3.1	2.7	2.4	3.3	0
Coil3	45.8	48.7	40.3	41.3	44.5	47.2	36.7	0
Uspst2	15.6	25.7	16.6	16.1	20	16	17.7	3

Table 14: Experiments in a pairwise binary setting. Uspst2 is the same set as Uspst but where the task is to classify digits 0 to 4 versus 5 to 9.

2. In a one-versus-the-rest approach, the negative class is the concatenation of several classes and is thus made up of several clusters. This might accentuate the local minimum problem of S^{3}VMs.

To test these hypothesis, we created a binary version of Uspst by classifying digits 0 to 4 versus 5 to 9: this data set (Uspst2 in Table 14) is balanced but each class is made of several clusters. The fact that the S^{3}VMs algorithms were not able to perform significantly better than the SVM baseline tends to accredit the second hypothesis: S^{3}VM results deteriorate when there are several clusters per class.

5.6.2 HYBRID S^{3}VM-GRAPH METHODS

Recall that the results in Table 8 boost the empirical evidence that S^{3}VMs do not return state-of-the-art performance on "manifold"-like data sets. On these data sets the cluster assumption is presumably satisfied under an intrinsic "geodesic" distance rather than the original euclidean distance between data points. It is reasonable, therefore, to attempt to combine S^{3}VMs with choices of kernels that conform to the particular geometry of the data.

LapSVM (Sindhwani et al., 2005) is a popular semi-supervised algorithm in which a kernel function is constructed from the Laplacian of an adjacency graph that models the data geometry; this kernel is then used with a standard SVM. This procedure was shown to be very effective on data sets with a manifold structure. In Table 15 we report results with a hybrid S^{3}VM-graph method: the kernel is constructed as in LapSVM,[16] but then it is plugged into S^{3}VM^{light}. Such a hybrid was first described in Chapelle et al. (2006a) where cS^{3}VM was combined with LapSVM.

The results of this hybrid method is very satisfactory, often outperforming both LapSVM and S^{3}VM^{light}.

Such a method complements the strengths of both S^{3}VM and Graph-based approaches: the S^{3}VM adds robustness to the construction of the graph, while the graph enforces the right cluster structure to alleviate local minima problems in S^{3}VMs. We believe that this kind of technique is probably one of the most robust and powerful way for a semi-supervised learning problem.

16. Hyperparameters were chosen based on experiments in Sindhwani et al. (2005) without any extensive tuning.

		Exact r	(Table 8 setting)	Estimated r	(Table 13 setting)
	LapSVM	S^3VM^{light}	LapSVM-S^3VM^{light}	S^3VM^{light}	LapSVM-S^3VM^{light}
g50c	6.4	6.2	4.6	7.5	6.1
Text	11	8.1	8.3	9.2	9.0
Uspst	11.4	15.5	8.8	24.4	19.6
Isolet	41.2	30.0	46.5	36.0	49.0
Coil20	11.9	25.3	12.5	25.3	12.5
Coil3	20.6	56.7	17.9	56.7	17.9
2moons	7.8	68.8	5.1	68.8	5.1

Table 15: Comparison of a Graph-based method, LapSVM (Sindhwani et al., 2005), with S^3VM^{light} and hybrid LapSVM-S^3VM^{light} results under the settings of Table 8 and 13.

6. A Note on Computational Complexity

Even though a detailed comparison of the computational complexity of the different algorithms is out of the scope of this paper, we can still give the following rough picture.

First, all methods use annealing and so the complexity depends on the number of annealing steps. This dependency is probably sublinear because when the number of steps is large, retraining after each (small) step is less costly. We can divide the methods in two categories:

1. Methods whose complexity is of the same order as that of a standard SVM trained with the predicted labels of the unlabeled points, which is $O(n_{\mathsf{sv}}^3 + n^2)$ where n_{sv} is the number of points which are in a non-linear part of the loss function. This is clearly the case for S^3VM^{light} since it relies on an SVM optimization. Note that the training time of this algorithm can be sped up by swapping labels in "batch" rather than one by one (Sindhwani and Keerthi, 2006). CCCP is also in this category as the optimization problem it has to solve at each step is very close to an SVM. Finally, even if the Newton method is not directly solved via SVM, its complexity is also $O(n_{\mathsf{sv}}^3 + n^2)$ and we include it in this category. For both CCCP and Newton, the possibility of having a flat part in the middle of the loss function can reduce n_{sv} and thus the complexity. We have indeed observed with the Newton method that the convergence can be an order of magnitude faster when the loss includes this flat part in the middle (see Table 3).

2. Gradient based methods, namely ∇S^3VM, cS^3VM and ∇DA do not have any linear part in the objective function and so they scale as $O(n^3)$. Note that it is possible to devise a gradient based method and a loss function that contains some linear parts. The complexity of such an algorithm would be $O(nn_{\mathsf{sv}}^2 + n^2)$.

The original DA algorithm alternates between optimization of $\mathbf{w}$ and $\mathbf{p}_u$ and can be understood as block coordinate optimization. We found that it was significantly slower than the other algorithms; its direct gradient-based optimization counterpart, ∇DA, is usually much faster.

Finally, note that even if the algorithms of the second category have a complexity of $O(n^3)$, one can find an approximate solution by reducing the dimensionality of the problem from n to m and get a complexity of $O(nm^2)$. For instance, Chapelle et al. (2006a) reports a speed-up of 100 times without loss in accuracy for cS^3VM.

Ultimately a semi-supervised learning algorithm should be able to handle data sets with millions of unlabeled points. The best way of scaling up S^3VMs is still an open question and should be the topic of future research.

7. Conclusion

When practical S^3VM implementations fail to give good results on a problem, one might suspect that either: (a) the cluster assumption does not hold; or, (b) the cluster assumption holds but local minima problems are severe; or, (c) the S^3VM objective function is unable to implement the cluster assumption. We began our empirical study by benchmarking current S^3VM implementations against a global optimizer. Our results (see Section 5.3) narrowed the cause for performance loss down to suboptimal local minima, and established a correlation between generalization performance and the S^3VM objective function. For problems where the cluster assumption is true, we expect the S^3VM objective to indeed be an appropriate quantity to minimize. Due to non-convexity however, this minimization is not completely straightforward—an assortment of optimization techniques have been brought to bear on this problem with varying degrees of success. In this paper, we have reviewed these techniques and studied them empirically, taking several subtle differences into account. In a neutral experimental protocol, we were unable to identify any single technique as being consistently superior to another in terms of generalization performance. We believe better methods are still needed to optimize the S^3VM objective function.

While S^3VMs return good performance on textual data sets, they are currently not competitive with graph-methods on domains such as image classification often characterized by multiple, highly non-Gaussian clusters. A particularly promising class of techniques (see Section 5.6.2) is based on combining S^3VMs with graph methods.

S^3VMs have been sparingly explored in domains other than text and image classification. New application domains may provide additional insight into the behavior of S^3VM methods while enhancing their general appeal for semi-supervised learning. Another major theme is the extension of S^3VMs for structured output problems, possibly building on one of several recent lines of work for handling complex supervised learning tasks. A first step towards such an extension would require a clear statement of the cluster assumption applicable to the semi-supervised structured output setting. These are fertile areas for future research.

References

H. Abdi. Kendall rank correlation. In N.J. Salkind, editor, *Encyclopedia of Measurement and Statistics*. SAGE, 2006.

A. Astorino and A. Fuduli. Nonsmooth optimization techniques for semi-supervised classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2135–2142, 2007.

Y. Bengio and Y. Grandvalet. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, volume 17, 2004.

K. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems 12*, 1998.

T. De Bie and N. Cristianini. Semi-supervised learning using semi-definite programming. In O. Chapelle, B. Schoëlkopf, and A. Zien, editors, *Semi-supervised Learning*. MIT Press, 2006.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, New York, NY, 1992.

O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002.

O. Chapelle, M. Chi, and A. Zien. A continuation method for semi-supervised SVMs. In *International Conference on Machine Learning*, 2006a.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006b. URL `http://www.kyb.tuebingen.mpg.de/ssl-book`.

O. Chapelle, V. Sindhwani, and S. Keerthi. Branch and bound for semi-supervised support vector machines. In *Advances in Neural Information Processing Systems*, 2006c.

Y. Chen, G. Wang, and S. Dong. Learning with progressive transductive support vector machine. *Pattern Recognition Letter*, 24(12):1845–1855, 2003.

R. Cole, Y. Muthusamy, and M. Fanty. The ISOLET spoken letter database. Technical Report CS/E 90-004, Oregon Graduate Institute, 1990.

R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive SVMs. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, 1987.

G. Fung and O. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, 15:29–44, 2001.

T. Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, 1999.

H. Liu and S.-T. Huang. Fuzzy transductive support vector machines for hypertext classification. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(1):21–36, 2004.

S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-20). Technical Report CUCS-005-96, Columbia Univ., USA, 1996.

B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

M. Seeger. A taxonomy of semi-supervised learning methods. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Lerning*. MIT Press, 2006.

M. Silva, T. Maia, and A. Braga. An evolutionary approach to transduction in support vector machines. In *Fifth International Conference on Hybrid Intelligent Systems*, pages 329–334, 2005.

V. Sindhwani and S. S. Keerthi. Large scale semi-supervised linear SVMs. In *SIGIR*, 2006.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: From transductive to semi-supervised learning. In *International Conference on Machine Learning*, 2005.

V. Sindhwani, S. Keerthi, and O. Chapelle. Deterministic annealing for semi-supervised kernel machines. In *International Conference on Machine Learning*, 2006.

M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

V. Vapnik and A. Sterin. On structural risk minimization or overall risk in a problem of pattern recognition. *Automation and Remote Control*, 10(3):1495–1503, 1977.

V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., New York, 1998.

L. Wang, X. Shen, and W. Pan. On transductive support vector machines. In J. Verducci, X. Shen, and J. Lafferty, editors, *Prediction and Discovery*. American Mathematical Society, 2007.

W. Wapnik and A. Tscherwonenkis. *Theorie der Zeichenerkennung*. Akademie Verlag, Berlin, 1979.

Z. Wu. The effective energy transformation scheme as a special continuation approach to global optimization with application to molecular conformation. *SIAM Journal on Optimization*, 6(3): 748–768, 1996.

L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In *Advances in Neural Information Processing Systems*, 2004.

A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15:915–936, 2003.

X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report 02-107, CMU-CALD, 2002.

Journal of Machine Learning Research 9 (2008) 235-284 Submitted 9/06; Revised 9/07; Published 2/08

Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies

Andreas Krause KRAUSEA@CS.CMU.EDU
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Ajit Singh AJIT@CS.CMU.EDU
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Carlos Guestrin GUESTRIN@CS.CMU.EDU
Computer Science Department and Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Editor: Chris Williams

Abstract

When monitoring spatial phenomena, which can often be modeled as Gaussian processes (GPs), choosing sensor locations is a fundamental task. There are several common strategies to address this task, for example, geometry or disk models, placing sensors at the points of highest entropy (variance) in the GP model, and A-, D-, or E-optimal design. In this paper, we tackle the combinatorial optimization problem of maximizing the *mutual information* between the chosen locations and the locations which are not selected. We prove that the problem of finding the configuration that maximizes mutual information is NP-complete. To address this issue, we describe a polynomial-time approximation that is within $(1-1/e)$ of the optimum by exploiting the *submodularity* of mutual information. We also show how submodularity can be used to obtain online bounds, and design branch and bound search procedures. We then extend our algorithm to exploit lazy evaluations and local structure in the GP, yielding significant speedups. We also extend our approach to find placements which are robust against node failures and uncertainties in the model. These extensions are again associated with rigorous theoretical approximation guarantees, exploiting the submodularity of the objective function. We demonstrate the advantages of our approach towards optimizing mutual information in a very extensive empirical study on two real-world data sets.

Keywords: Gaussian processes, experimental design, active learning, spatial learning; sensor networks

1. Introduction

When monitoring spatial phenomena, such as temperatures in an indoor environment as shown in Figure 1(a), using a limited number of sensing devices, deciding where to place the sensors is

a fundamental task. One approach is to assume that sensors have a fixed sensing radius and to solve the task as an instance of the art-gallery problem (cf. Hochbaum and Maas, 1985; Gonzalez-Banos and Latombe, 2001). In practice, however, this geometric assumption is too strong; sensors make noisy measurements about the nearby environment, and this "sensing area" is not usually characterized by a regular disk, as illustrated by the temperature correlations in Figure 1(b). In addition, note that correlations can be both positive and negative, as shown in Figure 1(c), which again is not well-characterized by a disk model. Fundamentally, the notion that a single sensor needs to predict values in a nearby region is too strong. Often, correlations may be too weak to enable prediction from a single sensor. In other settings, a location may be "too far" from existing sensors to enable good prediction if we only consider one of them, but combining data from multiple sensors we can obtain accurate predictions. This notion of combination of data from multiple sensors in complex spaces is not easily characterized by existing geometric models.

An alternative approach from spatial statistics (Cressie, 1991; Caselton and Zidek, 1984), making weaker assumptions than the geometric approach, is to use a pilot deployment or expert knowledge to learn a *Gaussian process* (GP) model for the phenomena, a non-parametric generalization of linear regression that allows for the representation of uncertainty about predictions made over the sensed field. We can use data from a pilot study or expert knowledge to learn the (hyper-)parameters of this GP. The learned GP model can then be used to predict the effect of placing sensors at particular locations, and thus optimize their positions.[1]

Given a GP model, many criteria have been proposed for characterizing the quality of placements, including placing sensors at the points of highest entropy (variance) in the GP model, and A-, D-, or E-optimal design, and mutual information (cf. Shewry and Wynn, 1987; Caselton and Zidek, 1984; Cressie, 1991; Zhu and Stein, 2006; Zimmerman, 2006). A typical sensor placement technique is to greedily add sensors where uncertainty about the phenomena is highest, that is, the highest entropy location of the GP (Cressie, 1991; Shewry and Wynn, 1987). Unfortunately, this criterion suffers from a significant flaw: entropy is an *indirect* criterion, not considering the prediction quality of the selected placements. The highest entropy set, that is, the sensors that are most uncertain about each other's measurements, is usually characterized by sensor locations that are as far as possible from each other. Thus, the entropy criterion tends to place sensors along the borders of the area of interest (Ramakrishnan et al., 2005), for example, Figure 4. Since a sensor usually provides information about the area around it, a sensor on the boundary "wastes" sensed information.

An alternative criterion, proposed by Caselton and Zidek (1984), *mutual information*, seeks to find sensor placements that are most informative about unsensed locations. This optimization criterion *directly* measures the effect of sensor placements on the posterior uncertainty of the GP. In this paper, we consider the combinatorial optimization problem of selecting placements which maximize this criterion. We first prove that maximizing mutual information is an NP-complete problem. Then, by exploiting the fact that mutual information is a *submodular* function (cf. Nemhauser et al., 1978), we design the first approximation algorithm that guarantees a *constant-factor approximation* of the best set of sensor locations in polynomial time. To the best of our knowledge, no such guarantee exists for any other GP-based sensor placement approach, and for any other criterion. This guarantee

1. This initial GP is, of course, a rough model, and a sensor placement strategy can be viewed as an inner-loop step for an *active learning* algorithm (MacKay, 2003). Alternatively, if we can characterize the uncertainty about the parameters of the model, we can explicitly optimize the placements over possible models (Zidek et al., 2000; Zimmerman, 2006; Zhu and Stein, 2006).

holds both for placing a fixed number of sensors, and in the case where each sensor location can have a different cost.

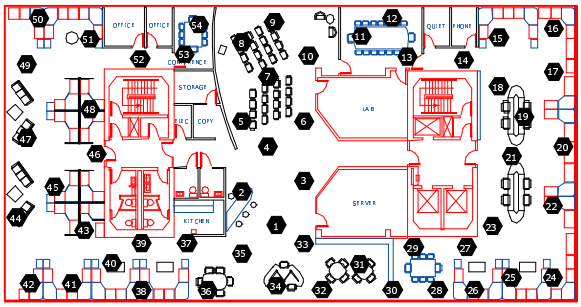
Though polynomial, the complexity of our basic algorithm is relatively high—$O(kn^4)$ to select k out of n possible sensor locations. We address this problem in two ways: First, we develop a lazy evaluation technique that exploits submodularity to reduce significantly the number of sensor locations that need to be checked, thus speeding up computation. Second, we show that if we exploit locality in sensing areas by trimming low covariance entries, we reduce the complexity to $O(kn)$.

We furthermore show, how the submodularity of mutual information can be used to derive tight online bounds on the solutions obtained by any algorithm. Thus, if an algorithm performs better than our simple proposed approach, our analysis can be used to bound how far the solution obtained by this alternative approach is from the optimal solution. Submodularity and these online bounds also allow us to formulate a mixed integer programming approach to compute the optimal solution using Branch and Bound. Finally, we show how mutual information can be made robust against node failures and model uncertainty, and how submodularity can again be exploited in these settings.

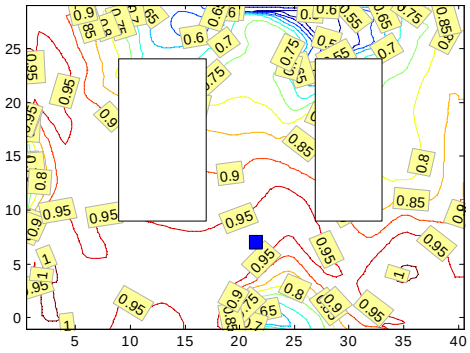
We provide a very extensive experimental evaluation, showing that data-driven placements outperform placements based on geometric considerations only. We also show that the *mutual information* criterion leads to improved prediction accuracies with a reduced number of sensors compared to several more commonly considered experimental design criteria, such as an entropy-based criterion, and A-optimal, D-optimal and E-optimal design criteria.

In summary, our main contributions are:

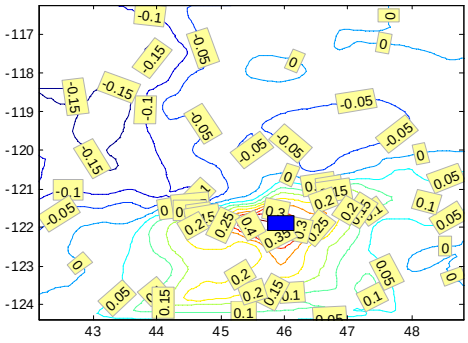
- We tackle the problem of maximizing the information-theoretic *mutual information* criterion of Caselton and Zidek (1984) for optimizing sensor placements, empirically demonstrating its advantages over more commonly used criteria.
- Even though we prove NP-hardness of the optimization problem, we present a polynomial time approximation algorithm with constant factor approximation guarantee, by exploiting *submodularity*. To the best of our knowledge, no such guarantee exists for any other GP-based sensor placement approach, and for any other criterion.
- We also show that submodularity provides online bounds for the quality of our solution, which can be used in the development of efficient branch-and-bound search techniques, or to bound the quality of the solutions obtained by other algorithms.
- We provide two practical techniques that significantly speed up the algorithm, and prove that they have no or minimal effect on the quality of the answer.
- We extend our analysis of mutual information to provide theoretical guarantees for placements that are robust against failures of nodes and uncertainties in the model.
- Extensive empirical evaluation of our methods on several real-world sensor placement problems and comparisons with several classical design criteria.



(a) *54 node sensor network deployment*



(b) *Temperature correlations*



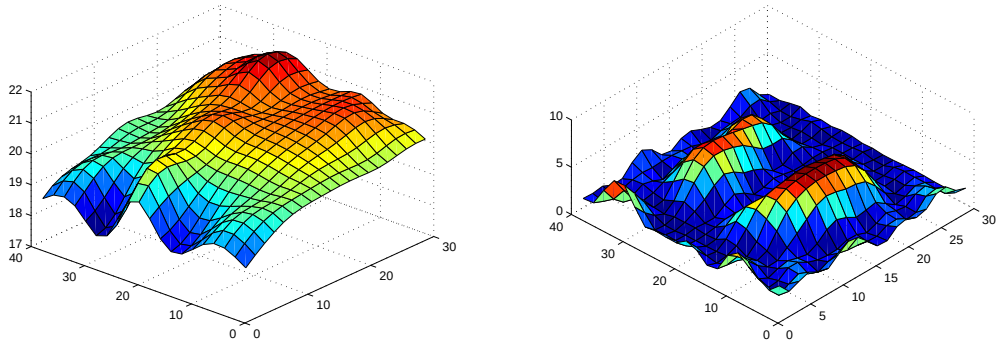
(c) *Precipitation correlations*

Figure 1: (a) A deployment of a sensor network with 54 nodes at the Intel Berkeley Lab. Correlations are often nonstationary as illustrated by (b) temperature data from the sensor network deployment in Figure 1(a), showing the correlation between a sensor placed on the blue square and other possible locations; (c) precipitation data from measurements made across the Pacific Northwest, Figure 11(b).

The paper is organized as follows. In Section 2, we introduce Gaussian Processes. We review mutual information criterion in Section 3, and describe our approximation algorithm to optimize mutual information in Section 4. Section 5 presents several approaches towards making the optimization more computationally efficient. In Section 6, we discuss how we can extend mutual information to be robust against node failures and uncertainty in the model. Section 8 relates our approach to other possible optimization criteria, and Section 7 describes related work. Section 9 presents our experiments.

2. Gaussian Processes

In this section, we review *Gaussian Processes*, the probabilistic model for spatial phenomena that forms the basis of our sensor placement algorithms.



(a) *Temperature prediction using GP* (b) *Variance of temperature prediction*

Figure 2: Posterior mean and variance of the temperature GP estimated using all sensors: (a) Predicted temperature; (b) predicted variance.

2.1 Modeling Sensor Data Using the Multivariate Normal Distribution

Consider, for example, the sensor network we deployed as shown in Figure 1(a) that measures a temperature field at 54 discrete locations. In order to predict the temperature at one of these locations from the other sensor readings, we need the joint distribution over temperatures at the 54 locations. A simple, yet often effective (cf. Deshpande et al., 2004), approach is to assume that the temperatures have a (multivariate) Gaussian joint distribution. Denoting the set of locations as $\mathcal{V}$, in our sensor network example $|\mathcal{V}| = 54$, we have a set of $n = |\mathcal{V}|$ corresponding random variables $X_{\mathcal{V}}$ with joint distribution:

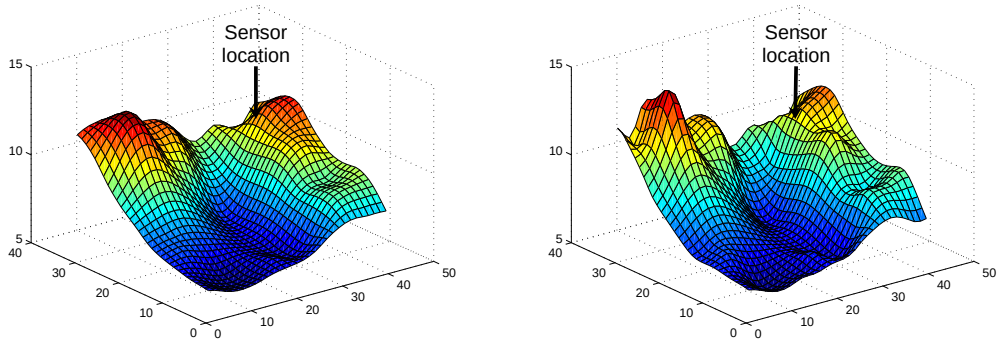
$$P(X_{\mathcal{V}} = \mathbf{x}_{\mathcal{V}}) = \frac{1}{(2\pi)^{n/2}|\Sigma_{\mathcal{V}\mathcal{V}}|} e^{-\frac{1}{2}(\mathbf{x}_{\mathcal{V}}-\mu_{\mathcal{V}})^T \Sigma^{-1}_{\mathcal{V}\mathcal{V}}(\mathbf{x}_{\mathcal{V}}-\mu_{\mathcal{V}})},$$

where $\mu_{\mathcal{V}}$ is the mean vector and $\Sigma_{\mathcal{V}\mathcal{V}}$ is the covariance matrix. Interestingly, if we consider a subset, $\mathcal{A} \subseteq \mathcal{V}$, of our random variables, denoted by $X_{\mathcal{A}}$, then their joint distribution is also Gaussian.

2.2 Modeling Sensor Data Using Gaussian Processes

In our sensor network example, we are not just interested in temperatures at sensed locations, but also at locations where no sensors were placed. In such cases, we can use regression techniques to perform prediction (Golub and Van Loan, 1989; Hastie et al., 2003). Although linear regression often gives excellent predictions, there is usually no notion of uncertainty about these predictions, for example, for Figure 1(a), we are likely to have better temperature estimates at points near existing sensors, than in the two central areas that were not instrumented. A *Gaussian process* (GP) is a natural generalization of linear regression that allows us to consider uncertainty about predictions.

Intuitively, a GP generalizes multivariate Gaussians to an infinite number of random variables. In analogy to the multivariate Gaussian above where the index set $\mathcal{V}$ was finite, we now have a (possibly uncountably) infinite index set $\mathcal{V}$. In our temperature example, $\mathcal{V}$ would be a subset of $\mathbb{R}^2$, and



(a) *Example kernel function.* (b) *Data from the empirical covariance matrix.*

Figure 3: Example kernel function learned from the Berkeley Lab temperature data: (a) learned covariance function $\mathcal{K}(x,\cdot)$, where x is the location of sensor 41; (b) "ground truth", interpolated empirical covariance values for the same sensors. Observe the close match between predicted and measured covariances.

each index would correspond to a position in the lab. GPs have been widely studied (cf. MacKay, 2003; Paciorek, 2003; Seeger, 2004; O'Hagan, 1978; Shewry and Wynn, 1987; Lindley and Smith, 1972), and generalize Kriging estimators commonly used in geostatistics (Cressie, 1991).

An important property of GPs is that for every finite subset $\mathcal{A}$ of the indices $\mathcal{V}$, which we can think about as locations in the plane, the joint distribution over the corresponding random variables $X_\mathcal{A}$ is Gaussian, for example, the joint distribution over temperatures at a finite number of sensor locations is Gaussian. In order to specify this distribution, a GP is associated with a *mean function* $\mathcal{M}(\cdot)$, and a symmetric positive-definite *kernel function* $\mathcal{K}(\cdot,\cdot)$, often called the covariance function. For each random variable with index $u \in \mathcal{V}$, its mean μ_u is given by $\mathcal{M}(u)$. Analogously, for each pair of indices $u,v \in \mathcal{V}$, their covariance σ_{uv} is given by $\mathcal{K}(u,v)$. For simplicity of notation, we denote the mean vector of some set of variables $X_\mathcal{A}$ by $\mu_\mathcal{A}$, where the entry for element u of $\mu_\mathcal{A}$ is $\mathcal{M}(u)$. Similarly, we denote their covariance matrix by $\Sigma_{\mathcal{A}\mathcal{A}}$, where the entry for u,v is $\mathcal{K}(u,v)$.

The GP representation is extremely powerful. For example, if we observe a set of sensor measurements $X_\mathcal{A} = \mathbf{x}_\mathcal{A}$ corresponding to the finite subset $\mathcal{A} \subset \mathcal{V}$, we can predict the value at any point $y \in \mathcal{V}$ conditioned on these measurements, $P(X_y \mid x_\mathcal{A})$. The distribution of X_y given these observations is a Gaussian whose conditional mean $\mu_{y|\mathcal{A}}$ and variance $\sigma^2_{y|\mathcal{A}}$ are given by:

$$\mu_{y|\mathcal{A}} = \mu_y + \Sigma_{y\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}(x_\mathcal{A} - \mu_\mathcal{A}), \quad (1)$$

$$\sigma^2_{y|\mathcal{A}} = \mathcal{K}(y,y) - \Sigma_{y\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\Sigma_{\mathcal{A}y}, \quad (2)$$

where $\Sigma_{y\mathcal{A}}$ is a covariance vector with one entry for each $u \in \mathcal{A}$ with value $\mathcal{K}(y,u)$, and $\Sigma_{\mathcal{A}y} = \Sigma^T_{y\mathcal{A}}$. Figure 2(a) and Figure 2(b) show the posterior mean and variance derived using these equations on 54 sensors at Intel Labs Berkeley. Note that two areas in the center of the lab were not instrumented. These areas have higher posterior variance, as expected. An important property of GPs is that the posterior variance (2) does not depend on the actual observed values $x_\mathcal{A}$. Thus, for a given kernel function, the variances in Figure 2(b) will not depend on the observed temperatures.

2.3 Nonstationarity

In order to compute predictive distributions using (1) and (2), the mean and kernel functions have to be known. The mean function can usually be estimated using regression techniques. Estimating kernel functions is difficult, and usually, strongly limiting assumptions are made. For example, it is commonly assumed that the kernel $\mathcal{K}(u,v)$ is *stationary*, which means that the kernel depends only on the difference between the locations, considered as vectors v, u, that is, $\mathcal{K}(u,v) = \mathcal{K}_\theta(u-v)$. Hereby, θ is a set of parameters. Very often, the kernel is even assumed to be *isotropic*, which means that the covariance only depends on the distance between locations, that is, $\mathcal{K}(u,v) = \mathcal{K}_\theta(||u-v||_2)$. Common choices for isotropic kernels are the exponential kernel, $\mathcal{K}_\theta(\delta) = \exp(-\frac{|\delta|}{\theta})$, and the Gaussian kernel, $\mathcal{K}_\theta(\delta) = \exp(-\frac{\delta^2}{\theta^2})$. These assumptions are frequently strongly violated in practice, as illustrated in the real sensor data shown in Figures 1(b) and 1(c). In Section 8.1, we discuss how placements optimized from models with isotropic kernels reduce to geometric covering and packing problems.

In this paper, we *do not* assume that $\mathcal{K}(\cdot,\cdot)$ is stationary or isotropic. Our approach is general, and can use *any* kernel function. In our experiments, we use the approach of Nott and Dunsmuir (2002) to estimate nonstationary kernels from data collected by an initial deployment. More specifically, their assumption is that an estimate of the empirical covariance $\Sigma_{\mathcal{A}\mathcal{A}}$ at a set of observed locations is available, and that the process can be locally described by a collection of isotropic processes, associated with a set of reference points. An example of a kernel function estimated using this method is presented in Figure 3(a). In Section 9.2, we show that placements based on such nonstationary GPs lead to far better prediction accuracies than those obtained from isotropic kernels.

3. Optimizing Sensor Placements

Usually, we are limited to deploying a small number of sensors, and thus must carefully choose where to place them. In spatial statistics this optimization is called *sampling* or *experimental design*: finding the k best sensor locations out of a finite subset $\mathcal{V}$ of possible locations, for example, out of a grid discretization of $\mathbb{R}^2$.

3.1 The Entropy Criterion

We first have to define what a good design is. Intuitively, we want to place sensors which are most informative with respect to the entire design space. A natural notion of uncertainty is the conditional entropy of the unobserved locations $\mathcal{V}\setminus\mathcal{A}$ after placing sensors at locations $\mathcal{A}$,

$$H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}}) = -\int p(\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}}, \mathbf{x}_{\mathcal{A}}) \log p(\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathbf{x}_{\mathcal{A}}) d\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}} d\mathbf{x}_{\mathcal{A}}, \tag{3}$$

where we use $\mathcal{X}_{\mathcal{A}}$ and $\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}$ to refer to sets of random variables at the locations $\mathcal{A}$ and $\mathcal{V}\setminus\mathcal{A}$. Intuitively, minimizing this quantity aims at finding the placement which results in the lowest uncertainty about all uninstrumented locations $\mathcal{V}\setminus\mathcal{A}$ after observing the placed sensors $\mathcal{A}$. A good placement would therefore minimize this conditional entropy, that is, we want to find

$$\mathcal{A}^* = \operatorname{argmin}_{\mathcal{A}\subset\mathcal{V}:|\mathcal{A}|=k} H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}}).$$

Using the identity $H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}}) = H(\mathcal{X}_{\mathcal{V}}) - H(\mathcal{X}_{\mathcal{A}})$, we can see that

$$\mathcal{A}^* = \text{argmin}_{\mathcal{A}\subset\mathcal{V}:|\mathcal{A}|=k} H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}}) = \text{argmax}_{\mathcal{A}\subset\mathcal{V}:|\mathcal{A}|=k} H(\mathcal{X}_{\mathcal{A}}).$$

So we can see that we need to find a set of sensors $\mathcal{A}$ which is most uncertain about each other. Unfortunately, this optimization problem, often also referred to as D-optimal design in the experiment design literature (cf. Currin et al., 1991), has been shown to be NP-hard (Ko et al., 1995):

Theorem 1 (Ko et al., 1995) *Given rational M and rational covariance matrix $\Sigma_{\mathcal{V}\mathcal{V}}$ over Gaussian random variables $\mathcal{V}$, deciding whether there exists a subset $\mathcal{A} \subseteq \mathcal{V}$ of cardinality k such that $H(\mathcal{X}_{\mathcal{A}}) \geq M$ is NP-complete.*

Therefore, the following greedy heuristic has found common use (McKay et al., 1979; Cressie, 1991): One starts from an empty set of locations, $\mathcal{A}_0 = \emptyset$, and greedily adds placements until $|\mathcal{A}| = k$. At each iteration, starting with set $\mathcal{A}_i$, the greedy rule used is to add the location $y^*_H \in \mathcal{V} \setminus \mathcal{A}$ that has highest conditional entropy,

$$y^*_H = \text{argmax}_y H(\mathcal{X}_y \mid \mathcal{X}_{\mathcal{A}_i}), \tag{4}$$

that is, the location we are most uncertain about given the sensors placed thus far. If the set of selected locations at iteration i is $\mathcal{A}_i = \{y_1, \ldots, y_i\}$, using the chain-rule of entropies, we have that:

$$H(\mathcal{X}_{\mathcal{A}_i}) = H(\mathcal{X}_{y_i} \mid \mathcal{X}_{\mathcal{A}_{i-1}}) + \ldots + H(\mathcal{X}_{y_2} \mid \mathcal{X}_{\mathcal{A}_1}) + H(\mathcal{X}_{y_1} \mid \mathcal{X}_{\mathcal{A}_0}).$$

Note that the (differential) entropy of a Gaussian random variable $\mathcal{X}_y$ conditioned on some set of variables $\mathcal{X}_{\mathcal{A}}$ is a monotonic function of its variance:

$$H(\mathcal{X}_y \mid \mathcal{X}_{\mathcal{A}}) = \frac{1}{2}\log(2\pi e \sigma^2_{\mathcal{X}_y|\mathcal{X}_{\mathcal{A}}}) = \frac{1}{2}\log\sigma^2_{\mathcal{X}_y|\mathcal{X}_{\mathcal{A}}} + \frac{1}{2}(\log(2\pi)+1), \tag{5}$$

which can be computed in closed form using Equation (2). Since for a fixed kernel function, the variance does not depend on the observed values, this optimization can be done before deploying the sensors, that is, a sequential, closed-loop design taking into account previous measurements bears no advantages over an open-loop design, performed before any measurements are made.

3.2 An Improved Design Criterion: Mutual Information

The entropy criterion described above is intuitive for finding sensor placements, since the sensors that are most uncertain about each other should cover the space well. Unfortunately, this entropy criterion suffers from the problem shown in Figure 4, where sensors are placed far apart along the boundary of the space. Since we expect predictions made from a sensor measurement to be most precise in a region around it, such placements on the boundary are likely to "waste" information. This phenomenon has been noticed previously by Ramakrishnan et al. (2005), who proposed a weighting heuristic. Intuitively, this problem arises because the entropy criterion is *indirect*: the criterion only considers the entropy of the selected sensor locations, rather than considering prediction quality over the space of interest. This indirect quality of the entropy criterion is surprising, since

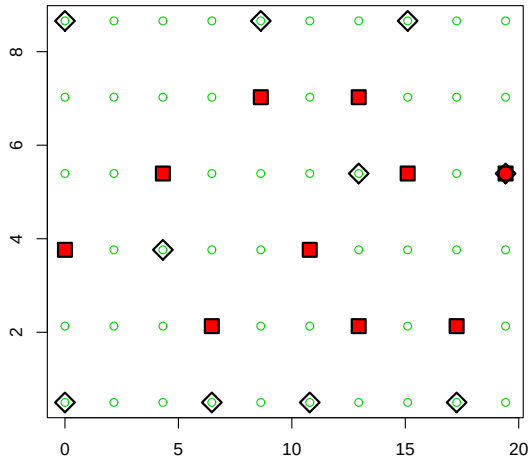


Figure 4: An example of placements chosen using entropy and mutual information criteria on a subset of the temperature data from the Intel deployment. Diamonds indicate the positions chosen using entropy; squares the positions chosen using MI.

the criterion was derived from the "predictive" formulation $H(\mathcal{V} \setminus \mathcal{A} \mid \mathcal{A})$ in Equation (3), which is equivalent to maximizing $H(\mathcal{A})$.

Caselton and Zidek (1984) proposed a different optimization criterion, which searches for the subset of sensor locations that most significantly reduces the uncertainty about the estimates in the rest of the space. More formally, we consider our space as a discrete set of locations $\mathcal{V} = \mathcal{S} \cup \mathcal{U}$ composed of two parts: a set $\mathcal{S}$ of possible positions where we can place sensors, and another set $\mathcal{U}$ of positions of interest, where no sensor placements are possible. The goal is to place a set of k sensors that will give us good predictions at all uninstrumented locations $\mathcal{V} \setminus \mathcal{A}$. Specifically, we want to find

$$\mathcal{A}^* = \operatorname{argmax}_{\mathcal{A} \subseteq \mathcal{S}: |\mathcal{A}| = k} H(\mathcal{X}_{\mathcal{V} \setminus \mathcal{A}}) - H(\mathcal{X}_{\mathcal{V} \setminus \mathcal{A}} \mid \mathcal{X}_{\mathcal{A}}),$$

that is, the set $\mathcal{A}^*$ that maximally reduces the entropy over the rest of the space $\mathcal{V} \setminus \mathcal{A}^*$. Note that this criterion $H(\mathcal{X}_{\mathcal{V} \setminus \mathcal{A}}) - H(\mathcal{X}_{\mathcal{V} \setminus \mathcal{A}} \mid \mathcal{X}_{\mathcal{A}})$ is equivalent to finding the set that maximizes the *mutual information* $I(\mathcal{X}_{\mathcal{A}}; \mathcal{X}_{\mathcal{V} \setminus \mathcal{A}})$ between the locations $\mathcal{A}$ and the rest of the space $\mathcal{V} \setminus \mathcal{A}$. In their follow-up work, Caselton et al. (1992) and Zidek et al. (2000), argue against the use of mutual information in a setting where the entropy $H(\mathcal{X}_{\mathcal{A}})$ in the observed locations constitutes a significant part of the total uncertainty $H(\mathcal{X}_{\mathcal{V}})$. Caselton et al. (1992) also argue that, in order to compute $\mathrm{MI}(\mathcal{A})$, one needs an accurate model of $P(\mathcal{X}_{\mathcal{V}})$. Since then, the entropy criterion has been dominantly used as a placement criterion. Nowadays however, the estimation of complex nonstationary models for $P(\mathcal{X}_{\mathcal{V}})$, as well as computational aspects, are very well understood and handled. Furthermore, we show empirically, that even in the sensor selection case, mutual information outperforms entropy on several practical placement problems.

On the same simple example in Figure 4, this mutual information criterion leads to intuitively appropriate central sensor placements that do not have the "wasted information" property of the entropy criterion. Our experimental results in Section 9 further demonstrate the advantages in performance of the mutual information criterion. For simplicity of notation, we will often use $\mathrm{MI}(\mathcal{A}) = I(\mathcal{X}_{\mathcal{A}}; \mathcal{X}_{\mathcal{V} \setminus \mathcal{A}})$ to denote the mutual information objective function. Notice that in this no-

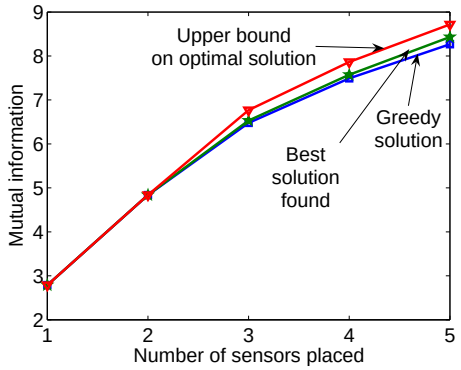


Figure 5: Comparison of the greedy algorithm with the optimal solutions on a small problem. We select from 1 to 5 sensor locations out of 16, on the Intel Berkeley temperature data set as discussed in Section 9. The greedy algorithm is always within 95 percent of the optimal solution.

tation the process $\mathcal{X}$ and the set of locations $\mathcal{V}$ is implicit. We will also write $H(\mathcal{A})$ instead of $H(\mathcal{X}_\mathcal{A})$.

The mutual information is also hard to optimize:

Theorem 2 *Given rational M and a rational covariance matrix $\Sigma_{\mathcal{V}\mathcal{V}}$ over Gaussian random variables $\mathcal{V} = \mathcal{S} \cup \mathcal{U}$, deciding whether there exists a subset $\mathcal{A} \subseteq \mathcal{S}$ of cardinality k such that $\mathrm{MI}(\mathcal{A}) \geq M$ is NP-complete.*

Proofs of all results are given in Appendix A. Due to the problem complexity, we cannot expect to find optimal solutions in polynomial time. However, if we implement the simple greedy algorithm for the mutual information criterion (details given below), and optimize designs on real-world placement problems, we see that the greedy algorithm gives almost optimal solutions, as presented in Figure 5. In this small example, where we could compute the optimal solution, the performance of the greedy algorithm was at most five percent worse than the optimal solution. In the following sections, we will give theoretical bounds and empirical evidence justifying this near-optimal behavior.

4. Approximation Algorithm

Optimizing the mutual information criterion is an NP-complete problem. We now describe a polynomial time algorithm with a constant-factor approximation guarantee.

4.1 The Algorithm

Our algorithm is greedy, simply adding sensors in sequence, choosing the next sensor which provides the maximum increase in mutual information. More formally, using $\mathrm{MI}(\mathcal{A}) = I(\mathcal{X}_\mathcal{A}; \mathcal{X}_{\mathcal{V}\setminus\mathcal{A}})$,

Input: Covariance matrix $\Sigma_{\mathcal{V}\mathcal{V}}$, k, $\mathcal{V} = \mathcal{S} \cup \mathcal{U}$
Output: Sensor selection $\mathcal{A} \subseteq \mathcal{S}$
begin
  $\mathcal{A} \leftarrow \emptyset$;
  for $j = 1$ **to** k **do**
    for $y \in \mathcal{S} \setminus \mathcal{A}$ **do** $\delta_y \leftarrow \dfrac{\sigma_y^2 - \Sigma_{y\mathcal{A}}\Sigma_{\mathcal{A}\mathcal{A}}^{-1}\Sigma_{\mathcal{A}y}}{\sigma_y^2 - \Sigma_{y\bar{\mathcal{A}}}\Sigma_{\bar{\mathcal{A}}\bar{\mathcal{A}}}^{-1}\Sigma_{\bar{\mathcal{A}}y}}$;
1
    $y^* \leftarrow \operatorname{argmax}_{y \in \mathcal{S}\setminus\mathcal{A}} \delta_y$;
2
    $\mathcal{A} \leftarrow \mathcal{A} \cup y^*$;
end

Algorithm 1: Approximation algorithm for maximizing mutual information.

our goal is to greedily select the next sensor y that maximizes:

$$\begin{aligned} \mathrm{MI}(\mathcal{A}\cup y) - \mathrm{MI}(\mathcal{A}) &= H(\mathcal{A}\cup y) - H(\mathcal{A}\cup y \mid \bar{\mathcal{A}}) - \left[H(\mathcal{A}) - H(\mathcal{A} \mid \bar{\mathcal{A}}\cup y)\right] \\ &= H(\mathcal{A}\cup y) - H(\mathcal{V}) + H(\bar{\mathcal{A}}) - \left[H(\mathcal{A}) - H(\mathcal{V}) + H(\bar{\mathcal{A}}\cup y)\right] \\ &= H(y \mid \mathcal{A}) - H(y \mid \bar{\mathcal{A}}), \end{aligned} \tag{6}$$

where, to simplify notation, we write $\mathcal{A}\cup y$ to denote the set $\mathcal{A}\cup\{y\}$, and use $\bar{\mathcal{A}}$ to mean $\mathcal{V}\setminus(\mathcal{A}\cup y)$. Note that the greedy rule for entropy in Equation (4) only considers the $H(y \mid \mathcal{A})$ part of Equation (6), measuring the uncertainty of location y with respect to the placements $\mathcal{A}$. In contrast, the greedy mutual information trades off this uncertainty with $-H(y \mid \bar{\mathcal{A}})$, which forces us to pick a y that is "central" with respect to the unselected locations $\bar{\mathcal{A}}$, since those "central" locations will result in the least conditional entropy $H(y \mid \bar{\mathcal{A}})$. Using the definition of conditional entropy in Equation (5), Algorithm 1 shows our greedy sensor placement algorithm.

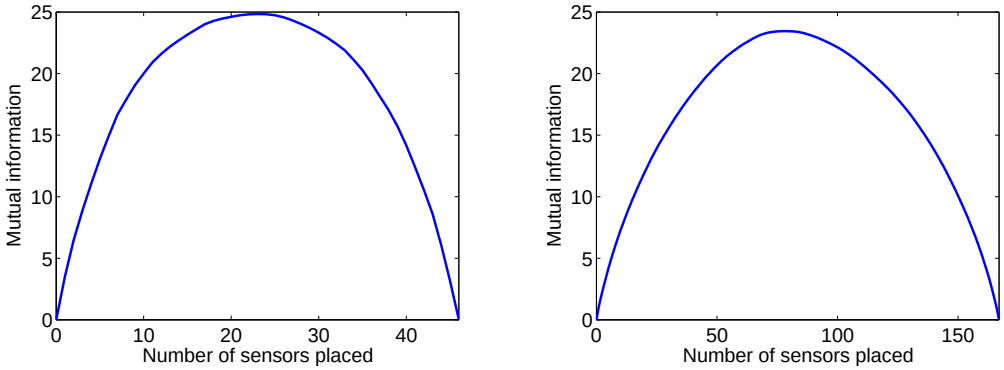
4.2 An Approximation Bound

We now prove that, if the discretization $\mathcal{V}$ of locations of interest in the Gaussian process is fine enough, our greedy algorithm gives a $(1-1/e)$ approximation, approximately 63% of the optimal sensor placement: If the algorithm returns set $\widehat{A}$, then

$$\mathrm{MI}(\widehat{A}) \geq (1-1/e) \max_{\mathcal{A}\subset\mathcal{S},|\mathcal{A}|=k} \mathrm{MI}(\mathcal{A}) - k\varepsilon,$$

for some small $\varepsilon > 0$. To prove this result, we use *submodularity* (cf. Nemhauser et al., 1978). Formally, a set function F is called *submodular*, if for all $\mathcal{A},\mathcal{B} \subseteq \mathcal{V}$ it holds that $F(\mathcal{A}\cup\mathcal{B}) + F(\mathcal{A}\cap\mathcal{B}) \leq F(\mathcal{A}) + F(\mathcal{B})$. Equivalently, using an induction argument as done by Nemhauser et al. (1978), a set function is submodular if for all $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathcal{V}$ and $y \in \mathcal{V}\setminus\mathcal{A}'$ it holds that $F(\mathcal{A}\cup y) - F(\mathcal{A}) \geq F(\mathcal{A}'\cup y) - F(\mathcal{A}')$. This second characterization intuitively represents "diminishing returns": adding a sensor y when we only have a small set of sensors $\mathcal{A}$ gives us more advantage than adding y to a larger set of sensors $\mathcal{A}'$. Using the "information never hurts" bound, $H(y \mid \mathcal{A}) \geq H(y \mid \mathcal{A}\cup\mathcal{B})$ (Cover and Thomas, 1991), note that our greedy update rule maximizing $H(y \mid \mathcal{A}) - H(y \mid \bar{\mathcal{A}})$ implies

$$\mathrm{MI}(\mathcal{A}'\cup y) - \mathrm{MI}(\mathcal{A}') \leq \mathrm{MI}(\mathcal{A}\cup y) - \mathrm{MI}(\mathcal{A}),$$



(a) *Temperature data* (b) *Precipitation data*

Figure 6: Mutual information of greedy sets of increasing size. It can be seen that clearly mutual information is not monotonic. MI is monotonic, however, in the initial part of the curve corresponding to small placements. This allows us to prove approximate monotonicity.

whenever $\mathcal{A} \subseteq \mathcal{A}'$, and any $y \in \mathcal{V} \setminus \mathcal{A}'$, that is, adding y to $\mathcal{A}$ helps more than adding y to $\mathcal{A}'$. In fact, this inequality holds for arbitrary sets $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathcal{V}$ and $y \in \mathcal{V} \setminus \mathcal{A}'$, not just for the sets considered by the greedy algorithm. Hence we have shown:

Lemma 3 *The set function $\mathcal{A} \mapsto \mathrm{MI}(\mathcal{A})$ is submodular.*

A submodular set function F is called monotonic if $F(\mathcal{A} \cup y) \geq F(\mathcal{A})$ for $y \in \mathcal{V}$. For such functions, Nemhauser et al. (1978) prove the following fundamental result:

Theorem 4 (Nemhauser et al., 1978) *Let F be a monotone submodular set function over a finite ground set $\mathcal{V}$ with $F(\emptyset) = 0$. Let $\mathcal{A}_G$ be the set of the first k elements chosen by the greedy algorithm, and let* $\mathrm{OPT} = \max_{\mathcal{A} \subset \mathcal{V}, |\mathcal{A}| = k} F(\mathcal{A})$. *Then*

$$F(\mathcal{A}_G) \geq \left(1 - \left(\frac{k-1}{k}\right)^k\right) \mathrm{OPT} \geq (1 - 1/e)\,\mathrm{OPT}\,.$$

Hence the greedy algorithm guarantees a performance guarantee of $(1-1/e)\,\mathrm{OPT}$, where OPT is the value of the optimal subset of size k. This greedy algorithm is defined by selecting in each step the element $y^* = \mathrm{argmax}_y F(\mathcal{A} \cup y) - F(\mathcal{A})$. This is exactly the algorithm we proposed in the previous section for optimizing sensor placements (Algorithm 1).

Clearly, $\mathrm{MI}(\emptyset) = I(\emptyset; \mathcal{V}) = 0$, as required by Theorem 4. However, the monotonicity of mutual information is not apparent. Since $\mathrm{MI}(\mathcal{V}) = I(\mathcal{V}, \emptyset) = 0$, the objective function will increase and then decrease, and, thus, is not monotonic, as shown in Figures 6(a) and 6(b). Fortunately, the proof of Nemhauser et al. (1978) does not use monotonicity for all possible sets, it is sufficient to prove that MI is monotonic for all sets of size up to $2k$. Intuitively, mutual information is not monotonic when the set of sensor locations approaches $\mathcal{V}$. If the discretization level is significantly larger than $2k$ points, then mutual information should meet the conditions of the proof of Theorem 4.

Thus the heart of our analysis of Algorithm 1 will be to prove that if the discretization of the Gaussian process is fine enough, then mutual information is *approximately monotonic* for sets of size up to $2k$. More precisely, we prove the following result:

Lemma 5 *Let $\mathcal{X}$ be a Gaussian process on a compact subset $\mathcal{C}$ of $\mathbb{R}^m$ with a positive-definite, continuous covariance kernel $\mathcal{K}: \mathcal{C} \times \mathcal{C} \rightarrow \mathbb{R}_0^+$. Assume the sensors have a measurement error with variance at least σ^2. Then, for any $\varepsilon > 0$, and any finite maximum number k of sensors to place there exists a discretization $\mathcal{V} = \mathcal{S} \cup \mathcal{U}$, $\mathcal{S}$ and $\mathcal{U}$ having mesh width δ such that $\forall y \in \mathcal{V} \setminus \mathcal{A}, \mathrm{MI}(\mathcal{A} \cup y) \geq \mathrm{MI}(\mathcal{A}) - \varepsilon$ for all $\mathcal{A} \subseteq \mathcal{S}$, $|\mathcal{A}| \leq 2k$.*

If the covariance function is Lipschitz-continuous, such as the Gaussian Radial Basis Function (RBF) kernel, the following corollary gives a bound on the required discretization level with respect to the Lipschitz constant:

Corollary 6 *If $\mathcal{K}$ is Lipschitz-continuous with constant L, then the required discretization is*

$$\delta \leq \frac{\varepsilon \sigma^6}{4kLM\left(\sigma^2 + 2k^2M + 6k^2\sigma^2\right)},$$

where $M = \max_{x \in C} \mathcal{K}(x,x)$, for $\varepsilon < \min(M, 1)$.

Corollary 6 guarantees that for any $\varepsilon > 0$, a polynomial discretization level is sufficient to guarantee that mutual information is $\varepsilon-$approximately monotonic. These bounds on the discretization are, of course, worst case bounds. The worst-case setting occurs when the sensor placements $\mathcal{A}$ are arbitrarily close to each other, since the entropy part $H(y \mid \mathcal{A})$ in Equation (6) can become negative. Since most GPs are used for modeling physical phenomena, both the optimal sensor placement and the sensor placement produced by the greedy algorithm can be expected to be spread out, and not condensed to a small region of the sensing area. Hence we expect the bounds to be very loose in the situations that arise during normal operation of the greedy algorithm.

Combining our Lemmas 3 and 5 with Theorem 4, we obtain our constant-factor approximation bound on the quality of the sensor placements obtained by our algorithm:

Theorem 7 *Under the assumptions of Lemma 5, Algorithm 1 is guaranteed to select a set $\mathcal{A}$ of k sensors for which*

$$\mathrm{MI}(\mathcal{A}) \geq (1 - 1/e)(\mathrm{OPT} - k\varepsilon),$$

where OPT *is the value of the mutual information for the optimal placement.*

Note that our bound has two implications: First, it shows that our greedy algorithm has a guaranteed minimum performance level of $1 - 1/e$ when compared to the optimal solution. Second, our approach also provides an upper-bound on the value of the optimal placement, which can be used to bound the quality of the placements by other heuristic approaches, such as local search, that may perform better than our greedy algorithm on specific problems.

4.3 Sensor Placement with Non-constant Cost Functions

In many real-world settings, the cost of placing a sensor depends on the specific location. Such cases can often be formalized by specifying a total budget L, and the task is to select placements $\mathcal{A}$ whose total cost $c(\mathcal{A})$ is within our budget. Recently, the submodular function maximization approach of Nemhauser et al. (1978) has been extended to address this budgeted case (Sviridenko, 2004; Krause and Guestrin, 2005), in the case of modular cost functions, that is, $c(\mathcal{A}) = \sum_{i=1}^{k} c(\mathcal{X}_i)$, where $\mathcal{A} = \{\mathcal{X}_1, \ldots, \mathcal{X}_k\}$ and $c(\mathcal{X}_i)$ is the cost for selecting element $\mathcal{X}_i$. The combination of the

analysis in this paper with these new results also yields a constant-factor $(1-1/e)$ approximation guarantee for the sensor placement problem with non-uniform costs.

The algorithm for this budgeted case first enumerates all subsets of cardinality at most three. For each of these candidate subsets, we run a greedy algorithm, which adds elements until the budget is exhausted. The greedy rule optimizes a benefit cost ratio, picking the element for which the increase of mutual information divided by the cost of placing the sensor is maximized: More formally, at each step, the greedy algorithm adds the element y^* such that

$$y^* = \text{argmax}_{y \in \mathcal{S} \setminus \mathcal{A}} \frac{H(y \mid \mathcal{A}) - H(y \mid \bar{\mathcal{A}})}{c(y)}.$$

Krause and Guestrin (2005) show that this algorithm achieves an approximation guarantee of

$$(1-1/e)\,\text{OPT} - \frac{2L\varepsilon}{c_{\min}},$$

where L is the available budget, and $c_{\min}$ is the minimum cost of all locations. A requirement for this result to hold is that mutual information is ε-monotonic up to sets of size $\frac{2L}{c_{\min}}$. The necessary discretization level can be established similarly as in Corollary 6, with k replaced by $\frac{L}{c_{\min}}$.

4.4 Online Bounds

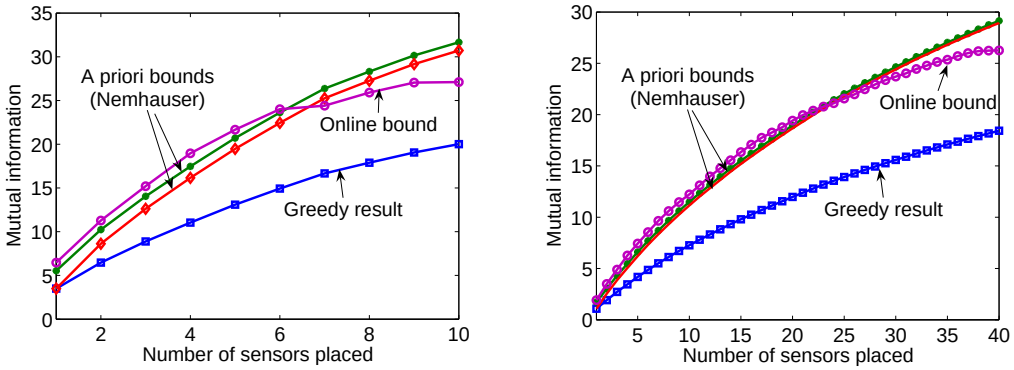
Since mutual information is approximately monotonic and submodular, Theorem 7 proves an *a priori* approximation guarantee of $(1-1/e)$. For most practical problems however, this bound is very loose. The following observation allows to compute online bounds on the optimal value:

Proposition 8 *Assume that the discretization is fine enough to guarantee ε-monotonicity for mutual information, and that the greedy algorithm returns an approximate solution $\mathcal{A}_k$, $|\mathcal{A}_k| = k$. For all $y \in \mathcal{S}$, let $\delta_y = \text{MI}(\mathcal{A} \cup y) - \text{MI}(\mathcal{A})$. Sort the δ_y in decreasing order, and consider the sequence $\delta^{(1)}, \ldots, \delta^{(k)}$ of the first k elements. Then* $\text{OPT} \leq \text{MI}(\mathcal{A}_k) + \sum_{i=1}^{k} \delta^{(i)} + k\varepsilon$.

The proof of this proposition follows directly from submodularity and ε-monotonicity. In many applications, especially for large placements, this bound can be much tighter than the bound guaranteed by Theorem 7. Figures 7(a) and 7(b) compare the a priori and online bounds for the data sets discussed in Section 9.1.

4.5 Exact Optimization and Tighter Bounds Using Mixed Integer Programming

There is another way to get even tighter bounds, or even compute the optimal solution. This approach is based on branch & bound algorithm for solving a mixed integer program for monotonic submodular functions (Nemhauser and Wolsey, 1981). We used this algorithm to bound the value of the optimal solution in Figure 5.



(a) *Temperature data* (b) *Precipitation data*

Figure 7: Online bounds: mutual information achieved by the greedy algorithm, the $(1-1/e)$ and $1-(1-1/k)^k$ a priori bounds and the online bound described in Section 4.4.

The mixed integer program is given by:

$$\begin{aligned}
&\max \eta; \\
&\quad \eta \;\leq\; \mathrm{MI}(\mathcal{B}) + \sum_{y_i \in \mathcal{S}\setminus\mathcal{B}} \alpha_i [\mathrm{MI}(\mathcal{B}\cup y_i) - \mathrm{MI}(\mathcal{B})], \quad \forall \mathcal{B} \subseteq \mathcal{S}; && (7)\\
&\sum_i \alpha_i \;\leq\; k, \quad \forall i; && (8)\\
&\quad \alpha_i \;\in\; \{0,1\}, \quad \forall i;
\end{aligned}$$

where $\alpha_i = 1$ means that location y_i should be selected. Note that this MIP can be easily extended to handle the case in which each location can have a different cost, by replacing the constraint (8) by $\sum_i \alpha_i c_i \leq L$, where L is the budget and $c_i = c(y_i)$.

Unfortunately, this MIP has exponentially many constraints. Nemhauser and Wolsey (1981) proposed the following constraint generation algorithm: Let $\alpha^{\mathcal{A}}$ denote an assignment to $\alpha_1, \ldots, \alpha_n$ such that $\alpha_i = 1$ iff $y_i \in \mathcal{A}$. Starting with no constraints of type (7), the MIP is solved, and one checks whether the current solution $(\eta, \alpha^{\mathcal{B}})$ satisfies $\eta \leq \mathrm{MI}(\mathcal{B})$. If it does not, a violated constraint has been found. Since solving individual instances (even with only polynomially many constraints) is NP-hard, we need to resort to search heuristics such as Branch and Bound and Cut during the constraint generation process.

The analysis of this MIP, as presented by Nemhauser and Wolsey (1981), assumes monotonicity. In the case of mutual information, the objective is only approximately monotonic. In particular, consider a a placement defined by $\alpha^{\mathcal{A}}$. Then, by submodularity, for all $\mathcal{B}$, we have that:

$$\begin{aligned}
\mathrm{MI}(\mathcal{B}) + \sum_{y_i \in \mathcal{S}\setminus\mathcal{B}} \alpha_i^{\mathcal{A}} [\mathrm{MI}(\mathcal{B}\cup y_i) - \mathrm{MI}(\mathcal{B})] &= \mathrm{MI}(\mathcal{B}) + \sum_{y_i \in \mathcal{A}\setminus\mathcal{B}} [\mathrm{MI}(\mathcal{B}\cup y_i) - \mathrm{MI}(\mathcal{B})], \\
&\geq \mathrm{MI}(\mathcal{A}\cup\mathcal{B}).
\end{aligned}$$

By approximate monotonicity:

$$\mathrm{MI}(\mathcal{A}\cup\mathcal{B}) \geq \mathrm{MI}(\mathcal{A}) - k\varepsilon.$$

Thus, $(\hat{\eta}, \alpha^{\mathcal{A}})$, for $\hat{\eta} \leq \mathrm{MI}(\mathcal{A}) - k\varepsilon$ is a feasible solution for the mixed integer program. Since we are maximizing η, for the optimal solution $(\eta^*, \alpha^{\mathcal{A}^*})$ of the MIP it holds that

$$\mathrm{MI}(\mathcal{A}^*) \geq \mathrm{OPT} - k\varepsilon.$$

There is another MIP formulation for maximizing general submodular functions without the ε-monotonicity requirement. The details can be found in Nemhauser and Wolsey (1981). We however found this formulation to produce much looser bounds, and to take much longer to converge.

5. Scaling Up

Greedy updates for both entropy and mutual information require the computation of conditional entropies using Equation (5), which involves solving a system of $|\mathcal{A}|$ linear equations. For entropy maximization, where we consider $H(y \mid \mathcal{A})$ alone, the complexity of this operation is $O(k^3)$. To maximize the mutual information, we also need $H(y \mid \bar{\mathcal{A}})$ requiring $O(n^3)$, for $n = |\mathcal{V}|$. Since we need to recompute the score of all possible locations at every iteration of Algorithm 1, the complexity of our greedy approach for selecting k sensors is $O(kn^4)$, which is not computationally feasible for very fine discretizations (large n). In Section 5.1 we propose a lazy strategy which often allows to reduce the number of evaluations of the greedy rule, thereby often reducing the complexity to $O(kn^3)$. In Section 5.2 we present a way of exploiting the problem structure by using local kernels, which often reduces the complexity to $O(kn)$. Both approaches can be combined for even more efficient computation.

5.1 Lazy Evaluation Using Priority Queues

It is possible to improve the performance of Algorithm 1 directly under certain conditions by lazy evaluation of the incremental improvements in Line 1. A similar algorithm has been proposed by Robertazzi and Schwartz (1989) in the context of D-optimal design. At the start of the algorithm, all δ_y will be initialized to $+\infty$. The algorithm will maintain information about which δ_y are current, that is, have been computed for the current locations $\mathcal{A}$. Now, the greedy rule in Line 2 will find the node y largest δ_y. If this δ_y has not been updated for the current $\mathcal{A}$, the value is updated and reintroduced into the queue. This process is iterated until the location with maximum δ_y is has an updated value. The algorithm is presented in Algorithm 2. The correctness of this lazy procedure directly follows from submodularity: For a fixed location y, the sequence δ_y must be monotonically decreasing during course of the algorithm.

To understand the efficacy of this procedure, consider the following intuition: If a location y^* is selected, nearby locations will become significantly less desirable and their marginal increases δ_y will decrease significantly. When this happens, these location will not be considered as possible maxima for the greedy step for several iterations. This approach can save significant computation time—we have noticed a decrease of mutual information computations by a factor of six in our experiments described in Section 9.6.

This approach can be efficiently implemented by using a priority queue to maintain the advantages δ_y. Line 2 calls **deletemax** with complexity $O(\log n)$ and Line 3 uses the **insert** operation with com-

Input: Covariance matrix $\Sigma_{\mathcal{V}\mathcal{V}}$, k, $\mathcal{V} = \mathcal{S} \cup \mathcal{U}$
Output: Sensor selection $\mathcal{A} \subseteq \mathcal{S}$
begin
  $\mathcal{A} \leftarrow \emptyset$;
  foreach $y \in \mathcal{S}$ **do** $\delta_y \leftarrow +\infty$;
  for $j = 1$ **to** k **do**
1 **foreach** $y \in \mathcal{S} \setminus \mathcal{A}$ **do** $current_y \leftarrow$ **false**;
  while true do
2 $y^* \leftarrow \text{argmax}_{y \in \mathcal{S} \setminus \mathcal{A}} \delta_y$;
  if $current_{y^*}$ **then break**;
3 $\delta_{y^*} \leftarrow H(y \mid \mathcal{A}) - H(y \mid \bar{\mathcal{A}})$;
  $current_{y^*} \leftarrow$ **true**
  $\mathcal{A} \leftarrow \mathcal{A} \cup y^*$;
end

Algorithm 2: Approximation algorithm for maximizing mutual information efficiently using lazy evaluation.

plexity $O(1)$. Also, as stated Line 1 has an $O(n)$ complexity, and was introduced for simplicity of exposition. In reality, we annotate the δ_y's with the last iteration that they were updated, completely eliminating this step.

5.2 Local Kernels

In this section, we exploit locality in the kernel function to speed up the algorithm significantly: First, we note that, for many GPs, correlation decreases exponentially with the distance between points. Often, variables which are far apart are actually independent. These weak dependencies can be modeled using a covariance function $\mathcal{K}$ for which $\mathcal{K}(x,\cdot)$ has compact support, that is, that has non-zero value only for a small portion of the space. For example, consider the following isotropic covariance function proposed by Storkey (1999):

$$\mathcal{K}(x,y) = \begin{cases} \frac{(2\pi-\Delta)(1+(\cos\Delta)/2)+\frac{3}{2}\sin\Delta}{3\pi}, & \text{for } \Delta < 2\pi, \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where $\Delta = \beta\|x-y\|_2$, for $\beta > 0$. This covariance function resembles the Gaussian kernel $\mathcal{K}(x,y) = \exp(-\beta\|x-y\|_2^2/(2\pi))$ as shown in Figure 8, but is zero for distances larger than $2\pi/\beta$.

Even if the covariance function does not have compact support, it can be appropriate to compute $H(y \mid \tilde{\mathcal{B}}) \approx H(y \mid \mathcal{B})$ where $\tilde{\mathcal{B}}$ results from removing all elements x from $\mathcal{B}$ for which $|\mathcal{K}(x,y)| \leq \varepsilon$ for some small value of ε. This truncation is motivated by noting that:

$$\sigma^2_{y|\mathcal{B}\setminus x} - \sigma^2_{y|\mathcal{B}} \leq \frac{\mathcal{K}(y,x)^2}{\sigma_x^2} \leq \frac{\varepsilon^2}{\sigma_x^2}.$$

This implies that the decrease in entropy $H(y \mid \mathcal{B}\setminus x) - H(y \mid \mathcal{B})$ is at most $\varepsilon^2/(\sigma^2\sigma_x^2)$ (using a similar argument as the one in the proof of Lemma 5), assuming that each sensor has independent Gaussian

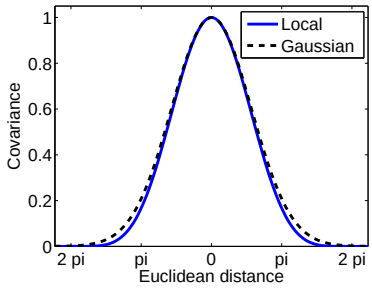


Figure 8: Comparison of local and Gaussian kernels.

Input: Covariance $\Sigma_{\mathcal{V}\mathcal{V}}$, k, $\mathcal{V} = \mathcal{S} \cup \mathcal{U}, \varepsilon > 0$
Output: Sensor selection $\mathcal{A} \subseteq \mathcal{S}$
begin
  $\mathcal{A} \leftarrow \emptyset$;
  foreach $y \in \mathcal{S}$ **do**
1 $\delta_y \leftarrow H(y) - \tilde{H}_\varepsilon(y \mid \mathcal{V} \setminus y)$;
  for $j = 1$ **to** k **do**
2 $y^* \leftarrow \arg\max_y \delta_y$;
  $\mathcal{A} \leftarrow \mathcal{A} \cup y^*$;
  foreach $y \in N(y^*; \varepsilon)$ **do**
3 $\delta_y \leftarrow \tilde{H}_\varepsilon(y \mid \mathcal{A}) - \tilde{H}_\varepsilon(y \mid \bar{\mathcal{A}})$;
end

Algorithm 3: Approximation algorithm for maximizing mutual information using local kernels.

measurement error of at least σ^2. The total decrease of entropy $H(y \mid \tilde{\mathcal{B}}) - H(y \mid \mathcal{B})$ is bounded by $n\varepsilon^2/\sigma^4$. This truncation allows to compute $H(y \mid \bar{\mathcal{A}})$ much more efficiently, at the expense of this small absolute error. In the special case of isotropic kernels, the number d of variables x with $\mathcal{K}(x,y) > \varepsilon$ can be computed as a function of the discretization and the covariance kernel. This reduces the complexity of computing $H(y \mid \bar{\mathcal{A}})$ from $O(n^3)$ to $O(d^3)$, which is a constant.

Our truncation approach leads to the more efficient optimization algorithm shown in Algorithm 3. Here, $\tilde{H}_\varepsilon$ refers to the truncated computation of entropy as described above, and $N(y^*; \varepsilon) \leq d$ refers to the set of elements $x \in \mathcal{S}$ for which $|\mathcal{K}(y^*, x)| > \varepsilon$. Using this approximation, our algorithm is significantly faster: Initialization (Line 1) requires $O(nd^3)$ operations. For each one of the k iterations, finding the next sensor (Line 2) requires $O(n)$ comparisons, and adding the new sensor y^* can only change the score of its neighbors ($N(y^*; \varepsilon) \leq d$), thus Line 3 requires $O(d \cdot d^3)$ operations. The total running time of Algorithm 3 is $O(nd^3 + kn + kd^4)$, which can be significantly lower than the $O(kn^4)$ operations required by Algorithm 1. Theorem 9 summarizes our analysis:

Theorem 9 *Under the assumptions of Lemma 5, guaranteeing ε_1-approximate monotonicity and truncation parameter ε_2, Algorithm 3 selects $\mathcal{A} \subseteq \mathcal{S}$ such that*

$$\mathrm{MI}(\mathcal{A}) \geq (1 - 1/e)(\mathrm{OPT} - k\varepsilon_1 - 2kn\varepsilon_2/\sigma^4),$$

in time $O(nd^3 + nk + kd^4)$.

This approach can be efficiently implemented by using a priority queue to maintain the advantages δ_y. Using for example a Relaxed Heaps data structure, the running time can be decreased to $O(nd^3 + kd \log n + kd^4)$: Line 1 uses the **insert** operation with complexity $O(1)$, Line 2 calls **deletemax** with complexity $O(\log n)$, and Line 3 uses **delete** and **insert**, again with complexity $O(\log n)$. This complexity improves on Algorithm 3 if $d \log n \ll n$. This assumption is frequently met in practice, since d can be considered a constant as the size n of the sensing area grows. Of course, this procedure can also be combined with the lazy evaluations described in the previous section for further improvement in running time.

6. Robust Sensor Placements

In this section, we show how the mutual information criterion can be extended to optimize for placements which are robust against failures of sensor nodes, and against uncertainty in the model parameters. The submodularity of mutual information will allow us to derive approximation guarantees in both cases.

6.1 Robustness Against Failures of Nodes

As with any physical device, sensor nodes are susceptible to failures. For example, the battery of a wireless sensor can run out, stopping it from making further measurements. Networking messages containing sensor values can be lost due to wireless interference. In the following, we discuss how the presented approach can handle such failures. We associate with each location $y_i \in \mathcal{S}$ a discrete random variable Z_i such that $Z_i = 0$ indicates that a sensor placed at location y_i has failed and will not produce any measurements, and $Z_i = 1$ indicates that the sensor is working correctly. For a placement $\mathcal{A} \subset \mathcal{S}$, denote by $\mathcal{A}_{\mathbf{z}}$ the subset of locations $y_i \in \mathcal{A}$ such that $z_i = 1$, that is, the subset of functional sensors. Then, the robust mutual information

$$\mathrm{MI}_R(\mathcal{A}) = \mathbb{E}_{\mathbf{Z}}[\mathcal{A}_{\mathbf{z}}] = \sum_{\mathbf{z}} P(\mathbf{z}) \, \mathrm{MI}(\mathcal{A}_{\mathbf{z}}),$$

is an expectation of the mutual information for placement $\mathcal{A}$ where all possible failure scenarios are considered.

Proposition 10 $\mathrm{MI}_R(\mathcal{A})$ *is submodular and, under the assumptions of Lemma 5, approximately monotonic.*

Proof This is a straightforward consequence of the fact that the class of submodular functions are closed under taking expectations. The approximate monotonicity can be verified directly from the approximate monotonicity of mutual information. ■

Unfortunately, the number of possible failure scenarios grows exponentially in $|\mathcal{S}|$. However, if the Z_i are i.i.d., and the failure probability $P(Z_i = 0) = \theta$ is low enough, MI_R can be approximated well, for example, by only taking into account scenarios were none or at most one sensor fails. This simplification often works in practice (Lerner and Parr, 2001). These $|\mathcal{S}| + 1$ scenarios can easily

be enumerated. For more complex distributions over Z, or higher failure probabilities θ, one might have to resort to sampling in order to compute MI_R.

The discussion above presents a means for explicitly optimizing placements for robustness. However, we can show that even if we do not specifically optimize for robustness, our sensor placements will be inherently robust:

Proposition 11 (Krause et al., 2006) *Consider a submodular function $F(\cdot)$ on a ground set $\mathcal{S}$, a set $\mathcal{B} \subseteq \mathcal{S}$, and a probability distribution over subsets $\mathcal{A}$ of $\mathcal{B}$ with the property that, for some constant ρ, we have $\Pr[v \in \mathcal{A}] \geq \rho$ for all $v \in \mathcal{B}$. Then $\mathbb{E}[F(\mathcal{A})] \geq \rho F(\mathcal{B})$.*

When applying this proposition, the set $\mathcal{B}$ will correspond to the selected sensor placement. The (randomly chosen) set $\mathcal{A}$ denotes the set of fully functioning nodes. If each node fails independently with probability $1-\rho$, that implies that $\Pr[c \in \mathcal{A}] \geq \rho$, and hence the expected mutual information of the functioning nodes, $\mathbb{E}[\mathrm{MI}(\mathcal{A})]$, is at least ρ times the mutual information $\mathrm{MI}(\mathcal{B})$, that is, when no nodes fail. Proposition 11 even applies if the node failures are not independent, but for example are spatially correlated, as can be expected in practical sensor placement scenarios.

6.2 Robustness Against Uncertainty in the Model Parameters

Often, the parameters θ of the GP prior, such as the amount of variance and spatial correlation in different areas of the space, are not known. Consequently, several researcher (Caselton et al., 1992; Zimmerman, 2006; Zhu and Stein, 2006) have proposed approaches to explicitly address the uncertainty in the model parameters, which are discussed in Section 7.

We want to exploit submodularity in order to get performance guarantees on the placements. We take a Bayesian approach, and equip θ with a prior. In this case, the objective function becomes

$$\mathrm{MI}_M(\mathcal{A}) = \mathbb{E}_\theta[I(\mathcal{A}; \mathcal{V} \setminus \mathcal{A} \mid \theta)] = \int p(\theta) I(\mathcal{A}; \mathcal{V} \setminus \mathcal{A} \mid \theta) d\theta.$$

Since the class of submodular functions is closed under expectations, MI_M is still a submodular function. However, the approximate monotonicity requires further assumptions. For example, if the discretization meshwidth is fine enough to guarantee approximate monotonicity for all values of θ for which $p(\theta) > 0$, then approximate monotonicity still holds, since

$$\begin{aligned}\mathrm{MI}_M(\mathcal{A} \cup y) - \mathrm{MI}_M(\mathcal{A}) &= \int p(\theta)[I(\mathcal{A} \cup y; \mathcal{V} \setminus (\mathcal{A} \cup y) \mid \theta) - I(\mathcal{A}; \mathcal{V} \setminus \mathcal{A} \mid \theta)] d\theta \\ &\geq \int p(\theta)[-\varepsilon] d\theta = -\varepsilon.\end{aligned}$$

A weaker assumption also suffices: If there exists a (nonnegative) function $\eta(\theta)$ such that $I(\mathcal{A} \cup y; \mathcal{V} \setminus (\mathcal{A} \cup y) \mid \theta) - I(\mathcal{A}; \mathcal{V} \setminus \mathcal{A} \mid \theta) \geq -\eta(\theta)$, and $\int p(\theta)[-\eta(\theta)] d\theta \geq -\varepsilon$, then MI_M is still ε-approximately monotonic. Such a function would allow the level ε of ε-approximately monotonicity to vary for different values of θ.

Note that in this setting however, the predictive distributions (1) and (2) cannot be computed in closed form anymore, and one has to resort to approximate inference techniques (cf. Rasmussen and Williams, 2006).

The advantage of exploiting submodularity for handling uncertainty in the model parameters is that the offline and online bounds discussed in Section 4.4 still apply. Hence, contrary to existing work, our approach provides strong theoretical guarantees on the achieved solutions.

7. Related Work

There is a large body of work related to sensor placement, and to the selection of observations for the purpose of coverage and prediction. Variations of this problem appear in spatial statistics, active learning, and experimental design. Generally, the methods define an objective function (Section 7.1), such as area coverage or predictive accuracy, and then apply a computational procedure (Section 7.2) to optimize this objective function. We also review related work on extensions to this basic scheme (Section 7.3), the related work in Machine Learning in particular (Section 7.4), and our previous work in this area (Section 7.5).

7.1 Objective Functions

We distinguish *geometric* and *model-based* approaches, which differ according to their assumptions made about the phenomenon to be monitored.

7.1.1 GEOMETRIC APPROACHES

Geometric approaches do not build a probabilistic model of the underlying process, but instead use geometric properties of the space in which the process occurs. The goal is typically a sensor placement that covers the space. The most common approaches for optimizing sensor placements using geometric criteria assume that sensors have a fixed region (cf. Hochbaum and Maas, 1985; Gonzalez-Banos and Latombe, 2001; Bai et al., 2006). These regions are usually convex or even circular. Furthermore, it is assumed that everything within this region can be perfectly observed, and everything outside cannot be measured by the sensors. In Section 8.1, we relate these geometric approaches to our GP-based formulation.

In the case where the sensing area is a disk (the *disk model*), Kershner (1939) has shown that an arrangement of the sensors in the centers of regular hexagons is asymptotically optimal, in the sense that a given set is fully covered by uniform disks. In Section 9.3, we experimentally show that when we apply the disk model to nonstationary placement problems, as considered in this paper, the geometric disk model approach leads to worse placements in terms of prediction accuracy, when compared to model-based approaches.

If many sensors are available then one can optimize the deployment density instead of the placement of individual sensors (Toumpis and Gupta, 2005). The locations of placed sensors are then assumed to be randomly sampled from this distribution. In the applications we consider, sensors are quite expensive, and optimal placement of a small set of them is desired.

7.1.2 MODEL-BASED APPROACHES

This paper is an example of a model-based method, one which takes a model of the world (here, a GP) and places sensors to optimize a function of that model (here, mutual information).

Many different objective functions have been proposed for model-based sensor placement. In the statistics community, classical and Bayesian experimental design focused on the question of selecting observations to maximize the quality of parameter estimates in linear models (cf. Atkinson, 1988; Lindley, 1956). In spatial statistics, information-theoretic measures, notably entropy, have been frequently used (Caselton and Hussain, 1980; Caselton and Zidek, 1984; Caselton et al., 1992; Shewry and Wynn, 1987; Federov and Mueller, 1989; Wu and Zidek, 1992; Guttorp et al., 1992). These objectives minimize the uncertainty in the prediction, after the observations are made.

Classical Experimental Design Criteria. In the statistics literature, the problem of optimal experimental design has been extensively studied (cf. Atkinson, 1988, 1996; Pukelsheim, 1987; Boyd and Vandenberghe, 2004). The problem commonly addressed there is to estimate the parameters θ of a function,

$$y = f_\theta(\mathbf{x}) + w,$$

where w is normally distributed measurement noise with zero mean and variance σ^2, y a scalar output and $\mathbf{x}$ a vectorial input. The assumption is, that the input $\mathbf{x}$ can be selected from a menu of design options, $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Each input corresponds to a possible experiment which can be performed. In our sensor placement case, one $\mathbf{x}$ would be associated with each location, y would be the measurement at the location, and θ would correspond to the values of the phenomenon at the unobserved locations. Usually, the assumption is that f_θ is linear, that is, $y = \theta^T \mathbf{x} + w$.

For the linear model $y = \theta^T \mathbf{x} + w$, if all n observations were available, then

$$\begin{aligned} \mathrm{Var}(\hat{\theta}) &= \sigma^2 (X^T X)^{-1} \\ \mathrm{Var}(\hat{y}_i) &= \sigma^2 \mathbf{x}_i^T (X^T X)^{-1} \mathbf{x}_i, \end{aligned} \tag{10}$$

where X is the design matrix, which consists of the inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ as its rows. We can see that the variance of both the parameter estimate $\hat{\theta}$ and the predictions $\hat{y}_i$ depends on the matrix $M = (X^T X)^{-1}$, which is called the inverse moment matrix. If this matrix is "small", then the parameter estimates and predictions will be accurate. A design consists of a selection $\mathcal{A}$ of the inputs (with repetitions allowed). We write $\mathcal{X}_\mathcal{A}$ to denote the selected experiments, and $M_\mathcal{A}$ for the corresponding inverse moment matrix. Classical experimental design considers different notions of "smallness" for this inverse moment matrix $M_\mathcal{A}$; D-optimality refers to the determinant, A-optimality to the trace and E-optimality to the spectral radius (the maximum eigenvalue). There are several more scalarizations of the inverse moment matrix, and they are commonly referred to as "alphabetical" optimality criteria.

An example of the relationship between this formalism and sensor placements in GPs, as well as experimental comparisons, are presented in Section 9.5.

Equation (10) shows that the distribution of the test data is not taken into account, when attempting to minimizing the inverse moment matrix $M_\mathcal{A}$. Yu et al. (2006) extend classical experimental design

to the transductive setting, which takes the distribution of test data into account. The information-theoretic approaches, which we use in this paper, also directly take into account the unobserved locations, as they minimize the uncertainty in the posterior $P(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}})$.

Bayesian Design Criteria. Classical experimental design is a Frequentist approach, which attempts to minimize the estimation error of the maximum likelihood parameter estimate. If one places a prior on the model parameters, one can formalize a Bayesian notion of experimental design. In its general form, Bayesian experimental design was pioneered by Lindley (1956). The users encode their preferences in a utility function $U(P(\Theta), \theta^\star)$, where the first argument, $P(\Theta)$, is a distribution over states of the world (i.e., the parameters) and the second argument, $\theta^\star$, is the true state of the world. Observations $\mathbf{x}_{\mathcal{A}}$ are collected, and the change in expected utility under the prior $P(\Theta)$ and posterior $P(\Theta \mid \mathcal{X}_{\mathcal{A}} = \mathbf{x}_{\mathcal{A}})$ can be used as a design criterion. By using different utility functions, Bayesian versions of *A*-, *D*-, and *E*- optimality can be developed (Chaloner and Verdinelli, 1995). If we have the posterior covariance matrix $\Sigma_{\theta|A}$, whose maximum eigenvalue is $\lambda_{\max}$, then Bayesian *A*-, *D*-, and *E*- optimality minimizes $\mathrm{tr}\left(\Sigma_{\theta|A}\right)$, $\det\left(\Sigma_{\theta|A}\right)$, and $\lambda_{\max}\left(\Sigma_{\theta|A}\right)$, respectively.

Usually, Bayesian experimental design considers the task of parameter estimation (Sebastiani and Wynn, 2000; Paninski, 2005; Ylvisaker, 1975). Lindley (1956) suggested using negative Shannon information, which is equivalent to maximizing the expected Kullback-Leibler divergence between the posterior and prior over the parameters:

$$\int P(\mathbf{x}_{\mathcal{A}}) \int P(\theta \mid \mathbf{x}_{\mathcal{A}}) \log \frac{P(\theta \mid \mathbf{x}_{\mathcal{A}})}{P(\theta)} \, d\theta d\mathbf{x}_{\mathcal{A}}. \tag{11}$$

If we consider distributions $P(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}})$ over the unobserved locations $\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}$ instead of distributions over parameters $P(\Theta)$, (11) leads to the following criterion:

$$\int P(\mathbf{x}_{\mathcal{A}}) \int P(\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathbf{x}_{\mathcal{A}}) \log \frac{P(\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathbf{x}_{\mathcal{A}})}{P(\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}})} \, d\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}} d\mathbf{x}_{\mathcal{A}}. \tag{12}$$

Note that Equation (12) is exactly the mutual information between the observed and unobserved sensors, $I(\mathcal{A}; \mathcal{V} \setminus \mathcal{A})$. For a linear-Gaussian model, where the mean and covariance are known, we get the mutual information criterion of Caselton and Zidek (1984), which we use in this paper.

Information-Theoretic Criteria. The special case of Bayesian experimental design, where an information-theoretic functional (such as entropy or mutual information) is used as a utility function, and where the predictive uncertainty in the unobserved variables is concerned (as in Equation 12) is of special importance for spatial monitoring.

Such information-theoretic criteria have been used as design criteria in a variety of fields and applications. Maximizing the entropy $H(\mathcal{A})$ of a set of observations, as discussed in Section 3, has been used in the design of computer experiments (Sacks et al., 1989; Currin et al., 1991), function interpolation (O'Hagan, 1978) and spatial statistics (Shewry and Wynn, 1987). This criterion is sometimes also referred to as D-optimality, since the scalarization of the posterior variance in the spatial literature and the scalarization of the parameter variance in classical experimental design

both involve a determinant. In the context of parameter estimation in linear models and independent, homoscedastic noise, maximizing the entropy $H(\mathcal{A})$ is equivalent to Bayesian D-optimal design (which maximizes the information gain $H(\Theta) - H(\Theta \mid \mathcal{A})$ about the parameters), as discussed by Sebastiani and Wynn (2000) (see also Section 8.4).

Maximizing mutual information between sets of random variables has a long history of use in statistics (Lindley, 1956; Bernardo, 1979), machine learning (Luttrell, 1985; MacKay, 1992). The specific form addressed in this paper, $I(\mathcal{A}; \mathcal{V} \setminus \mathcal{A})$, has been used in spatial statistics (Caselton and Zidek, 1984; Caselton et al., 1992). Mutual information requires an accurate estimate of the joint model $P(\mathcal{X}_{\mathcal{V}})$, while entropy only requires an accurate estimate at the selected locations, $P(\mathcal{X}_{\mathcal{A}})$. Caselton et al. (1992) argue that latter is easier to estimate from a small amount of data, thus arguing against mutual information. We however contend that nowadays effective techniques for learning complex nonstationary spatial models are available, such as the ones used in our experiments, thus mitigating these concerns and enabling the optimization of mutual information.

7.2 Optimization Techniques

All of the criteria discussed thus far yield challenging combinatorial optimization problems. Several approaches are used to solve them in the literature, which can be roughly categorized into those that respect the integrality constraint and those which use a continuous relaxation.

7.2.1 COMBINATORIAL SEARCH

For both geometric and model-based approaches, one must search for the best design or set of sensor locations among a very (usually exponentially) large number of candidate solutions. In a classical design, for example, the inverse moment matrix on a set of selected experiments $\mathcal{X}_{\mathcal{A}}$ can be written as

$$M_{\mathcal{A}} = \left(\sum_{i=1}^{n} k_i \mathbf{x}_i \mathbf{x}_i^T \right)^{-1},$$

where k_i is the number of times experiment $\mathbf{x}_i$ is performed in design $\mathcal{A}$. Since k_i must be an integer, a combinatorial number of potential experimental designs has to be searched. Similarly, when placing a set $\mathcal{A}$ of k sensors out of a set $\mathcal{V}$ of possible locations, as we do in this paper, all sets of size k have to be searched. For both entropy (Ko et al., 1995) and mutual information (this paper), this search has been shown to be NP-hard, hence efficient exact solutions are likely not possible.

Since exhaustive search is usually infeasible, local, heuristic searches without theoretical guarantees have commonly been applied. Approaches to the difficult combinatorial optimization include simulated annealing (Meyer and Nachtsheim, 1988), pairwise exchange (Fedorov, 1972; Mitchell, 1974a,b; Cook and Nachtsheim, 1980; Nguyen and Miller, 1992), forward and backward greedy heuristics (MacKay, 1992; Caselton and Zidek, 1984). All these approaches provide no guarantees about the quality of the solution. Since optimal solutions are highly desirable, branch-and-bound approaches to speed up the exhaustive search have been developed (Welch, 1982; Ko et al., 1995).

Although they enable exhaustive search for slightly larger problem instances, the computational complexity of the problems puts strong limits on their effectiveness.

By exploiting submodularity of mutual information, in this paper, we provide the first approach to information-theoretic sensor placement which has guarantees both on the runtime and on the quality of the achieved solutions.

7.2.2 CONTINUOUS RELAXATION

In some formulations, the integrality constraint is relaxed. For example, in classical experimental design, the number of experiments to be selected is often large compared to the number of design choices. In these cases, one can find a fractional design (i.e., a non-integral solution defining the proportions by which experiments should be performed), and round the fractional solutions. In the fractional formulation, A-, D-, and E-optimality criteria can be solved exactly using a semi-definite program (Boyd and Vandenberghe, 2004). There are however no known bounds on the integrality gap, that is, the loss incurred by this rounding process.

In other approaches (Seo et al., 2000; Snelson and Ghahramani, 2005), a set of locations is chosen not from a discrete, but a continuous space. If the objective function is differentiable with respect to these locations, gradient-based optimization can be used instead of requiring combinatorial search techniques. Nevertheless, optimality of the solution is not guaranteed since there is no known bound on the discrepancy between local and global optima.

Another method that yields a continuous optimization, in the case of geometric objective functions, is the potential field approach (Heo and Varshney, 2005; Howard et al., 2002). An energy criterion similar to a spring model is used. This optimization results in uniformly distributed (in terms of inter-sensor distances), homogeneous placements. The advantage of these approaches is that they can adapt to irregular spaces (such as hallways or corridors), where a simple grid-based deployment is not possible. Since the approach uses coordinate ascent, it can be performed using a distributed computation, making it useful for robotics applications where sensors can move.

7.3 Related Work on Extensions

In this section, we discuss prior work related to our extensions on sensor placement under model uncertainty (Section 6) and on the use of non-constant cost functions (Section 4.3).

7.3.1 PLACEMENT WITH MODEL UNCERTAINTY

The discussion thus far has focused on the case where the joint model $P(\mathcal{X}_\mathcal{V})$ is completely specified, that is, the mean and covariance of the GP are known.[2] With model uncertainty, one has to distinguish between observation selection for predictive accuracy in a fixed model and observation selection for learning parameters. Model uncertainty also introduces computational issues. If the mean and covariance are fixed in a Gaussian process then the posterior is Gaussian. This makes it

2. Or one assumes the uncertainty on these parameters is small enough that their contribution to the predictive uncertainty is negligible.

easy to compute quantities such as entropy and mutual information. If the mean and covariance are unknown, and we have to learn hyperparameters (e.g., kernel bandwidth of an isotropic process), then the predictive distributions and information-theoretic quantities often lack a closed form.

Caselton et al. (1992) extend their earlier work on maximum entropy sampling to the case where the mean and covariance are unknown by using a conjugate Bayesian analysis. The limitations of this approach are that the conjugate Bayesian analysis makes spatial independence assumptions in the prior and that complete data with repeated observations are required at every potential sensing site. This leads to a determinant maximization problem, much like *D*-optimality, that precludes the use of submodularity.

Another approach is the development of hybrid criteria, which balance parameter estimation and prediction. For example, Zimmerman (2006) proposes local EK-optimality, a linear combination of the maximum predictive variance and a scalarization of the covariance of the maximum likelihood parameter estimate. While this criterion selects observations which reduce parameter uncertainty and predictive uncertainty given the *current parameter*, it does not take into account the effect of parameter uncertainty on prediction error. To address this issue, Zhu and Stein (2006) derive an iterative algorithm which alternates between optimizing the design for covariance estimation and spatial prediction. This procedure does not provide guarantees on the quality of designs.

An alternative approach to addressing model uncertainty, in the context of classical experimental design, is presented by Flaherty et al. (2006). There, instead of committing to a single value, the parameters of a linear model are constrained to lie in a bounded interval. Their robust design objective, which is based on E-optimality, is then defined with respect to the worst-case parameter value. Flaherty et al. (2006) demonstrate how a continuous relaxation of this problem can be formulated as a SDP, which can be solved exactly. No guarantees are given however on the integrality gap on this relaxation.

In our approach, as discussed in Section 6, we show how submodularity can be exploited even in the presence of parameter uncertainty. We do not address the computational issues, which depend on the particular parameterization of the GP used. However, in special cases (e.g., uncertainty about the kernel bandwidth), one can apply sampling or numerical integration, and still get guarantees about the achieved solution.

7.3.2 NON-CONSTANT COST FUNCTIONS

In Section 4.3, we discuss the case where every sensor can have a different cost, and one has a budget which one can spend. An alternate approach to sensor costs is presented by Zidek et al. (2000). They propose a criterion that makes a trade off between achieved reduction in entropy using an entropy-to-cost conversion factor, that is, they optimize the sum of the entropy with a factor times the cost of the placements. This criterion yields an unconstrained optimization problem. Our approach to sensor costs (Section 4.3) yields a constrained optimization, maximizing our criteria given a fixed budget that can be spent when placing sensors. Such a budget-based approach seems more natural in real problems (where one often has a fixed number of sensors or amount of money to spend). Moreover, our approach provides strong a priori theoretical guarantees and tighter online bounds, which are not available for the approach of Zidek et al. (2000).

7.4 Related Work in Machine Learning

In Machine Learning, several related techniques have been developed for selecting informative features, for active learning and for speeding up GP inference.

7.4.1 FEATURE SELECTION AND DIMENSION REDUCTION

Given that the joint distribution of X_A and $X_{V \setminus A}$ is Gaussian, their mutual information is also

$$\mathrm{MI}(A) = -\frac{1}{2}\sum_i \log\left(1-\rho_i^2\right) \tag{13}$$

where $\rho_1^2 \geq \cdots \geq \rho_{|V|}^2$ are the canonical correlation coefficients between X_A and $X_{V \setminus A}$ (Caselton and Zidek, 1984). McCabe (1984) show that maximizing the canonical correlations between observed and unobserved variables can be interpreted as a form of principal components analysis, where one realizes that selecting subsets of variables is a special kind of linear projection. A similar analysis is presented for entropy and other common design criteria. Using Equation (13), a similar relationship can be made to canonical correlation analysis (CCA; Hotelling, 1936), which finds linear projections for $V \setminus A$ and A that maximize the correlations in the lower dimensional space. By considering these lower-dimensional projections, one can determine how much variance is shared (jointly explained) by $V \setminus A$ and A.

While dimension reduction techniques such as Principal Component Analysis (PCA) or CCA can be used to find a lower dimensional representation of a high dimensional problem, these techniques usually find projections which are non-sparse, that is, which are linear combinations of (almost) all input variables. However, for interpretation purposes (and considering data acquisition cost), one often desires *sparse* projections, which are linear combinations of only a small subset of input variables. Moghaddam et al. (2005) and Moghaddam et al. (2006) consider the problem of selecting such sparse linear projections (subject to a constraint on the number of nonzero entries) of minimum reconstruction error (for PCA) and class separation (for LDA). In order to find these sparse projections, they propose two approaches: A mixed integer program, which can solve the problem optimally—albeit generally not in polynomial time, and a heuristic approach, using a greedy forward search followed by a greedy backward elimination. They provide several theoretical bounds, including a guarantee that this backward greedy algorithm achieves a solution of at least $\frac{k}{n}\lambda_{\max}$ where $n = |V|$, and k is the number of chosen observations (Moghaddam, 2007).

7.4.2 ACTIVE LEARNING

In the machine learning community, information-theoretic criteria have been used for active learning, techniques which allow the learning algorithm to influence the choice of training samples. For example, information-theoretic criteria have been used in the analysis of query-by-committee to select samples (Sollich, 1996; Freund et al., 1997; Axelrod et al., 2001). Following Lindley (1956), MacKay (1992) proposes selecting observations that maximize expected information gain, either in terms of entropy or cross entropy, using Federov exchange. As opposed to this paper, which addresses the optimization problem, MacKay (1992) focuses on comparing the different objective

criteria. Cohn (1994) proposes scoring each potential observation by measuring the average reduction in predicted variance at a set of reference points. There is some evidence which suggests that this approach can improve prediction in Gaussian process regression (Seo et al., 2000). Common to all these active learning approaches, as well as to this paper, is the problem of selecting a set of most informative observation. Unlike this paper, we are not aware of any prior work in this area which provides rigorous approximation guarantees for this problem.

7.4.3 FAST GAUSSIAN PROCESS METHODS

Information-theoretic criteria are also used in sparse GP modeling, which attempts to reduce the cost of inference by selecting a representative subset of the training data. Sample selection criteria have included KL-divergence (Seeger et al., 2003) and entropy (Lawrence et al., 2003). In contrast to sensor placement, where locations are chosen to minimize predictive uncertainty, in sparse GP methods, the samples are chosen such that the approximate posterior matches the true posterior (which uses the entire training set) as accurately as possible. Instead of choosing a subset of the training data, Snelson and Ghahramani (2005) propose to optimize the location of a set of "hallucinated" inputs. This approach results in a continuous optimization problem, which appears to be easier to solve (albeit with no performance guarantees) than the discrete subset selection problem.

7.5 Relationship to Previous Work of the Authors

An earlier version of this paper appeared as (Guestrin et al., 2005). The present version is substantially extended by new experiments on nonstationarity (Section 9.3, Section 9.2) and comparisons to classical experimental design (Section 9.5). New are also the discussion of robust placements in Section 6 and several extensions in Section 4 and Section 5.

Additionally, Krause et al. (2006) presented an approximation algorithm for optimizing node placements for sensor networks using GPs that takes into account both the informativeness of placements (analogously to the discussion in this paper) and the communication cost required to retrieve these measurements. Their approach uses GPs both for modeling the monitored phenomenon as well as the link qualities of the sensor network. Singh et al. (2007) consider the case of planning informative paths for multiple robots. Here, the goal is to select observations which are both informative, but also lie on a collection of paths, one for each robot, of bounded length. They develop an approximation algorithm with theoretical guarantees on the quality of the solution. In the setting of Krause et al. (2006) and Singh et al. (2007)—unlike the case considered in this paper, where there are no constraints on the location of the sensors—the greedy algorithm performs arbitrarily badly, and the papers describe more elaborate optimization algorithms. In these algorithms, the submodularity of mutual information is again the crucial property which allows the authors to obtain approximation guarantees for their approach.

8. Notes on Optimizing Other Objective Functions

In this section, we discuss some properties of alternative optimality criteria for sensor placement.

8.1 A Note on the Relationship with the Disk Model

The disk model for sensor placement (cf. Hochbaum and Maas, 1985; Bai et al., 2006) assumes that each sensor can perfectly observe everything within a radius of r and nothing else. Hence we can associate with every location $y \in \mathcal{V}$ a sensing region D_y, which, for a discretization of the space, corresponds to the locations contained within radius r of location y. For a set of locations $\mathcal{A}$, we can define the coverage $F_D(\mathcal{A}) = \bigcup_{y \in \mathcal{A}} D_y$. It can be easily seen that this criterion is monotonic, submodular and $F(\emptyset) = 0$. Hence optimizing placements for the disk model criterion is a submodular maximization problem, and the greedy algorithm can guarantee a constant factor $(1 - 1/e)\,\mathrm{OPT}$ approximation guarantee for finding the placement of k sensors with maximum coverage.

There is a sense, in which the approach of sensor placements in GPs can be considered a generalization of the disk model. If we assume an isotropic GP with local kernel function as the one presented in Figure 8, then a sensor measurement is correlated exactly with the locations within a disk around its location. If the process has constant variance, then the greedy algorithm will, for the first few sensors placed, only try to achieve a disjoint placement of the disks, and as such behave just like the greedy algorithm for disk covering.

However, once enough sensors have been placed so that these "disks" start to overlap, the behavior of the two approaches begins to differ: in a disk model there is no advantage in placing sensors that lead to overlapping disks. In a GP model, even an isotropic one, "overlapping disks" lead to better predictions in the overlapping area, a very natural consequence of the representation of the uncertainty in the process.

8.2 A Note on Maximizing the Entropy

As noted by Ko et al. (1995), entropy is also a submodular set function, suggesting a possible application of the result of Nemhauser et al. (1978) to the entropy criterion. The corresponding greedy algorithm adds the sensor y maximizing $H(\mathcal{A} \cup y) - H(\mathcal{A}) = H(y \mid \mathcal{A})$. Unfortunately, our analysis of approximate monotonicity does not extend to the entropy case: Consider $H(y \mid \mathcal{A})$ for $\mathcal{A} = \{z\}$, for sufficiently small measurement noise σ^2, we show that $H(y \mid \mathcal{A})$ can become arbitrarily negative as the mesh width of the discretization decreases. Thus, (even approximate) monotonicity does not hold for entropy, suggesting that the direct application of the result of Nemhauser et al. (1978) is not possible. More precisely, our negative result about the entropy criterion is:

Remark 12 *Under the same assumptions as in Lemma 5, for any $\varepsilon > 0$, there exists a mesh discretization width $\delta > 0$ such that for any discretization level δ', where $0 < \delta' \leq \delta$, entropy violates the monotonicity criterion by at least ε, if $\sigma^2 < \frac{1}{4\pi e}$.*

8.3 A Note on Maximizing the Information Gain

Another objective function of interest is the *information gain* of a sensor placement with respect to some distinguished variables of interest $\mathcal{U}$, that is, $\mathrm{IG}(\mathcal{A}) = I(\mathcal{A}; \mathcal{U}) = H(\mathcal{U}) - H(\mathcal{U} \mid \mathcal{A})$. Unfortunately, this objective function is *not* submodular, even in the case of multivariate normal distributions: Let $\mathcal{X} = (\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3)$ be distributed according to a multivariate Gaussian with zero mean and

covariance

$$\Sigma = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix}.$$

Let $\mathcal{U} = \{X_3\}$. Here, X_2 and X_3 are marginally independent. Thus, alone, X_2 provides no gain in information. However, X_2 does provide information about X_1. Thus, if we first place a sensor at position 1, placing a sensor at position 2 does help. More formally, $\mathrm{IG}(\{X_2\}) - \mathrm{IG}(\emptyset) = H(X_3) - H(X_3 \mid X_2) < H(X_3 \mid X_1) - H(X_3 \mid X_1, X_2) = \mathrm{IG}(\{X_1, X_2\}) - \mathrm{IG}(\{X_1\})$. Hence adding X_2 to the empty set achieves strictly less increase in information gain than adding X_2 to the singleton set containing X_1, contradicting the submodularity assumption.

Remark 13 *The information gain,* $\mathrm{IG}(\mathcal{A}) = I(\mathcal{A}; \mathcal{U})$ *is not submodular in* $\mathcal{A}$.

8.4 A Note on Using Experimental Design for Sensor Placement

As discussed in Section 7.1, the goal of classical experimental design is to find a set $\mathcal{A}$ of experimental stimuli $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ such that the parameter estimation error covariance $M_{\mathcal{A}} = (X_{\mathcal{A}}^T X_{\mathcal{A}})^{-1}$ is as small as possible, where $X_{\mathcal{A}}$ is a matrix containing the $\mathbf{x}_i$ as rows. The different optimality criteria vary in the criteria used for measuring the smallness of $M_{\mathcal{A}}$. Consider the case where the number p of parameters θ is greater than 1, as in the sensor placement setting, where θ are the uninstrumented locations. If we select less than p observations, that is, $|\mathcal{A}| \leq p$, then $(X_{\mathcal{A}}^T X_{\mathcal{A}})$ is not full rank, and $M_{\mathcal{A}}$ is infinite. Hence, for $|\mathcal{A}| < p$, all alphabetical optimality criteria are infinite. Consequently, the A-, D- and E-optimality criteria are infinite as well for all discrete designs $\mathcal{A}$ of size less than p, and hence two such designs are incomparable under these criteria. This incomparability implies that the greedy algorithm will have no notion of improvement, and cannot be used for optimizing discrete classical experimental designs. Hence, discrete classical design cannot be addressed using the concept of submodularity.[3]

In the case of Bayesian experimental design, the parameters are equipped with a prior, and hence the posterior error covariance will not be infinite. In this case however, none of Bayesian A-, D- and E-optimality can be optimized using the result by Nemhauser et al. (1978) in general:

- **Bayesian A-optimality.** Let $y = \theta^T X + w$ where $X = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \end{bmatrix}$. Also let $\theta = (\theta_1, \theta_2)$ as well as the noise $w = (w_1, w_2, w_3)$ have independent normal priors with mean 0 and variance 1. Then the joint distribution of (y, θ) is multivariate normal with mean 0 and covariance

$$\Sigma = \begin{bmatrix} 3 & 3 & 3 & 1 & 1 \\ 3 & 6 & 4 & 1 & 2 \\ 3 & 4 & 6 & 2 & 1 \\ 1 & 1 & 2 & 1 & 0 \\ 1 & 2 & 1 & 0 & 1 \end{bmatrix}.$$

3. Note that generally, GPs are infinite-dimensional objects, so using classical experimental design for finite linear models only makes limited sense for sensor placement, for example, if an appropriate discretization is chosen.

The goal of A-optimality is to maximally reduce the variance in the posterior distribution over θ, that is, we want to select $\mathcal{A} \subseteq \mathcal{V} = \{y_1, y_2, y_3\}$ in order to maximize $F(\mathcal{A}) = \mathrm{tr}(\Sigma_\theta) - \mathrm{tr}(\Sigma_{\theta|\mathcal{A}})$. Now, we can verify that $F(\{y_1,y_2,y_3\}) - F(\{y_1,y_2\}) \geq F(\{y_1,y_3\}) - F(\{y_1\})$, contradicting submodularity.

- **Bayesian D-optimality.** As shown, for example, by Sebastiani and Wynn (2000), in the case of Bayesian experimental design for linear regression with independent, homoscedastic noise, D-optimality is actually equivalent to the entropy criterion, that is, $\mathrm{argmax}_{|\mathcal{A}|\leq k} I(\Theta, \mathcal{A}) = \mathrm{argmax}_{|\mathcal{A}|\leq k} H(\mathcal{A})$. Hence, the same counterexample as in Section 8.2 shows, that, while Bayesian D-optimality is submodular in this case, it is arbitrarily non-monotonic, and hence the result of Nemhauser et al. (1978) does not apply. In the more general case of dependent, heteroscedastic noise, D-optimality is equivalent to the information gain criterion with $\mathcal{U} = \{\theta\}$ and the counterexample of Section 8.3 applies, contradicting submodularity.

- **Bayesian E-optimality.** Consider the case where $\theta = (\theta_1, \theta_2)$ equipped with a normal prior with zero mean and covariance $diag([1,1])$. Let $y_i = \theta^T \mathbf{x}_i + w$ where w is Gaussian noise with mean zero and variance ε. Let $\mathbf{x}_1 = (1,0)^T$ and $\mathbf{x}_2 = (0,1)^T$. In this setting, the goal of E-optimality is to maximize $F(\mathcal{A}) = \lambda_{\max}(\Sigma_\theta) - \lambda_{\max}(\Sigma_{\theta|\mathcal{A}})$. If we perform no experiment ($\mathcal{A} = \emptyset$), then the posterior error covariance, $\Sigma_{\theta|\mathcal{A}} = \Sigma_\theta$, and hence the maximum eigenvalue is 1. If we observe either y_1 or y_2, the largest eigenvalue is still $\lambda_{\max}(\Sigma_{\theta|x_\mathcal{A}}) = 1$, and hence $F(\emptyset) = F(\{y_1\}) = F(\{y_2\}) = 0$. But if we observe both y_1 and y_2, then $\lambda_{\max}(\Sigma_{\theta|\mathcal{A}}) = \frac{\varepsilon}{1+\varepsilon}$, and $F(\{y_1,y_2\}) > 0$. Hence $F(\{y_2,y_1\}) - F(\{y_1\}) > F(\{y_2\}) - F(\emptyset)$, that is, adding y_2 helps more if we add it to y_1 than if we add it to the empty set, contradicting submodularity.

Remark 14 *The analysis of Nemhauser et al. (1978) applies to neither of Bayesian A-, D-, and E-optimality in general.*

However, Das and Kempe (2008) show that in certain cases, under a condition of *conditional suppressor-freeness*, the variance reduction F (and hence Bayesian A-optimality) can indeed be shown to be submodular.

9. Experiments

We performed experiments on two real-world data sets, which are described in Section 9.1. In Section 9.2 we compare placements on stationary and nonstationary GP models. In Sections 9.3, 9.4 and 9.5 we compare mutual information with the disk model, with entropy and with other classical experimental design criteria, in terms of prediction accuracy. In 9.6 we compare the performance of the greedy algorithm with other heuristics, and in Section 9.7 we analyze the effect of exploiting local kernels.

9.1 Data Sets

We first introduce the data sets we consider in our experiments. In our first data set, we analyze temperature measurements from the network of 46 sensors shown in Figure 1(a). Our training data consisted of samples collected at 30 sec. intervals on 3 consecutive days (starting Feb. 28th 2004), the testing data consisted of the corresponding samples on the two following days.

Our second data set consists of precipitation data collected during the years 1949 - 1994 in the states of Washington and Oregon (Widmann and Bretherton, 1999). Overall 167 regions of equal area, approximately 50 km apart, reported the daily precipitation. To ensure the data could be reasonably modeled using a Gaussian process we applied a log-transformation, removed the daily mean, and only considered days during which rain was reported. After this preprocessing, we selected the initial two thirds of the data as training instances, and the remaining samples for testing purposes. From the training data, we estimated the mean and empirical covariance, and regularized it by adding independent measurement noise[4] of $\sigma^2 = 0.1$.

We computed error bars for the prediction accuracies in all of our experiments, but due to the violated independence of the collected samples (which are temporally correlated), these error bars are overconfident and hence not reported here. The estimated standard errors under the independence assumption are too small to be visible on the plots.

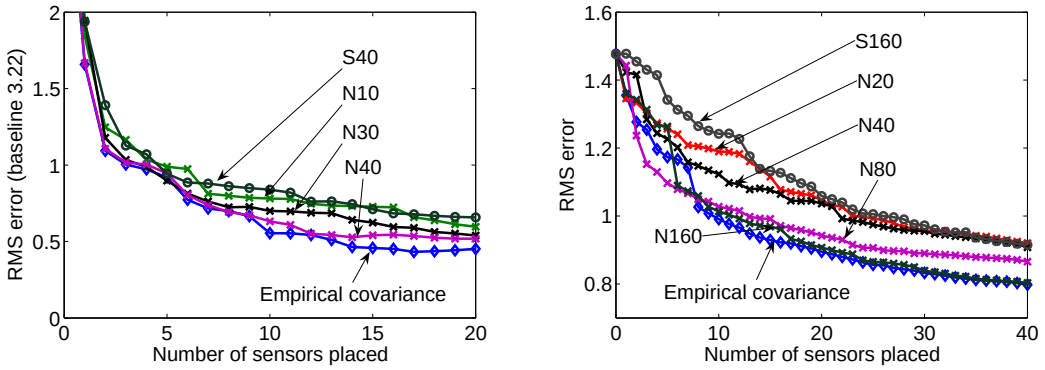
9.2 Comparison of Stationary and Non-stationary Models

To see how well both the stationary and nonstationary models capture the phenomenon, we performed the following experiment: We learned both stationary and non-stationary GP models from an increasing number of sensors. The model with stationary correlation function used an isotropic Exponential kernel with bandwidth fitted using least-squares fit of the empirical variogram (Cressie, 1991). We also learned a nonstationary GP using the technique from Nott and Dunsmuir (2002). Both GP models were estimated from an initial deployment of an increasing number of sensors. We used non-stationary the same variance process for both stationary and nonstationary models (i.e., giving more information to the stationary model than commonly done). Since with increasing amount of data, the empirical covariance matrix will exactly capture the underlying process, we consider the empirical covariance as the ground truth both for placements and prediction. Hence we also selected placements using the entire estimated covariance matrix.

We optimized the designs using mutual information on all the models. We evaluate the prediction accuracy for an increasing number of near-optimally placed sensors, using the estimated model and the measured values for the selected sensors. Figure 9(a) presents the results for the temperature data set. We can see that the nonstationary model learned from 10 sensors performs comparably to the stationary model with 40 sensors, even with non-stationary variance process. As we increase the number of sensors in the initial deployment, the Root Mean Squared error (RMS) prediction accuracies we get for placements of increasing size converge to those obtained for optimizing the placements based on the empirical covariance matrix.

Figure 9(b) presents the results of the same experiment for the precipitation data. Here we can see that the nonstationary model estimated using 20 sensors leads to better RMS accuracies than the stationary model, even if latter is estimated using 160 sensors.

4. The measurement noise σ^2 was chosen by cross-validation.



(a) *Comparison of fits for temperature data.* (b) *Comparison of fits for precipitation data.*

Figure 9: RMS curves for placements of increasing size, optimized using stationary and nonstationary GPs. Prediction is done using estimated models. (a) Stationary GP estimated for the temperature data from 40 sensors (S40), nonstationary GPs estimated from 10, 30 and 40 sensors (N10, N30, N40). (b) Stationary GP estimated for the precipitation data from 160 sensors (S160), nonstationary GPs estimated from 20, 40, 80 and 160 sensors (N20, N40, N80, N160).

9.3 Comparison of Data-driven Placements with Geometric Design Criteria

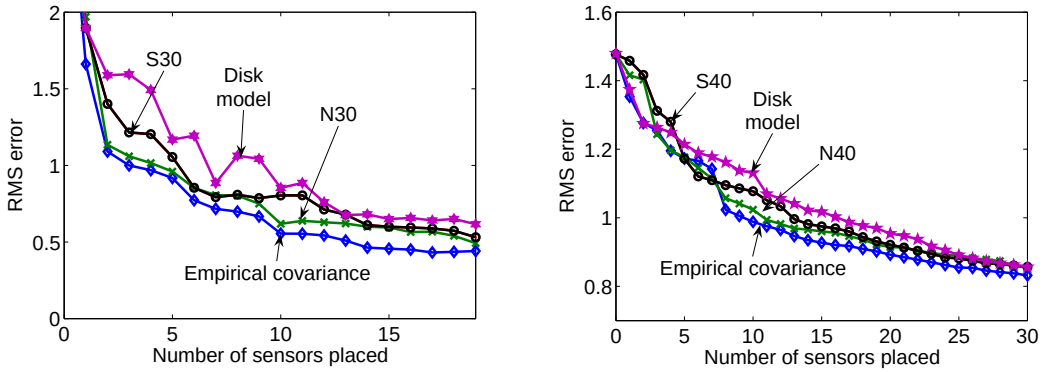
We now compare placements based on our data-driven GP models with those based on the traditional disk model. This model assumes that every sensor can perfectly measure the phenomenon within a radius of r, and have no information outside this radius. Since choosing an appropriate radius for the disk model is very difficult in practice, we decided to choose $r = 5m$ since for this radius 20 sensors could just spatially cover the entire space. We also learned stationary and non-stationary GP models as discussed in Section 9.2.

For the disk model, we used the greedy set covering algorithm. The design on both GP models was done using our greedy algorithm to maximize mutual information. For an increasing number of sensors, we compute the Root Mean Squares (RMS) prediction error on the test data. In order to separate the placement from the prediction task, we used the empirical covariance matrix estimated from the training data on all 46 locations for prediction, for all three placements.

Figure 10(a) presents the results of this experiment. We can see that the geometrical criterion performs poorly compared to the model based approaches. We can see that the placements based on the empirical covariance matrix perform best, quite closely followed by the accuracies obtained by the designs based on the nonstationary process. Figure 10(b) shows the results for the same experiment on the precipitation data set.

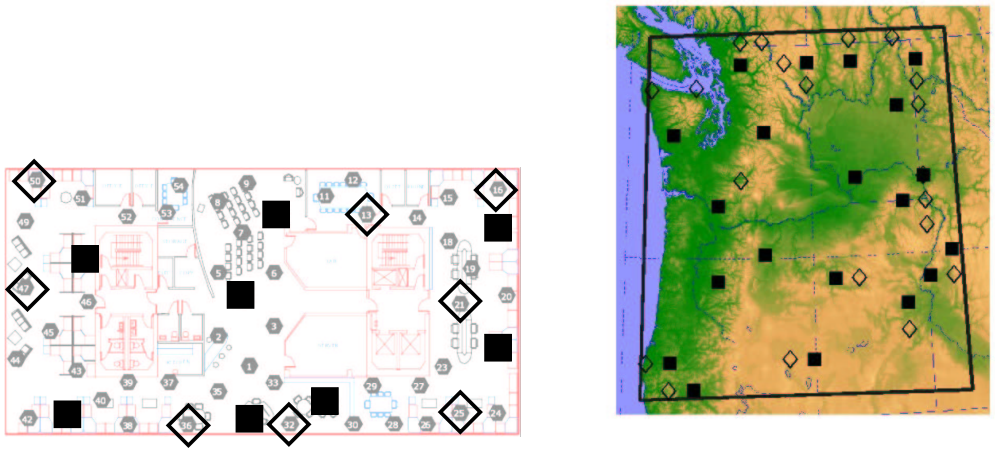
9.4 Comparison of the Mutual Information and Entropy Criteria

We also compared the mutual information criterion to other design criteria. We first compare it against the entropy (variance) criterion. Using the empirical covariance matrix as our process, we use the greedy algorithm to select placements of increasing size, both for mutual information and for entropy. Figure 12(a) and Figure 12(b) show the results of this comparison on models estimated



(a) *Comparison with disk model (Temperature)* (b) *Comparison with disk model (Precipitation)*

Figure 10: RMS curves for placements of increasing size, optimized using the disk model, stationary and nonstationary GPs. Prediction for all placements is done using the empirical covariance. Stationary GPs and nonstationary GPs estimated from 30 sensors (N30, S30, for temperature data) or 40 sensors (N40, S40, for precipitation data).

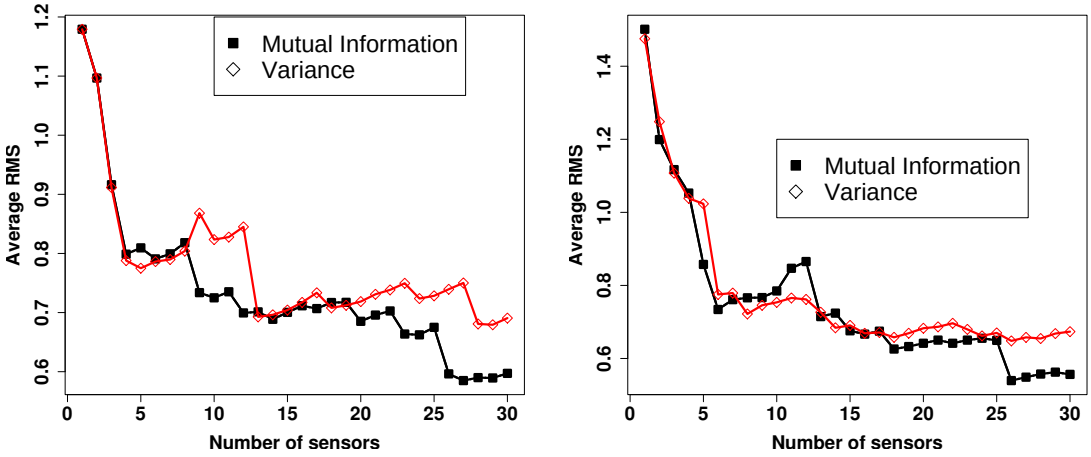


(a) *Placements of temperature sensors* (b) *Placements of rain sensors*

Figure 11: Example sensor placements for temperature and precipitation data. Squares indicate locations selected by mutual information, diamonds indicate those selected by entropy. Notice how entropy places sensors closer to the border of the sensing field.

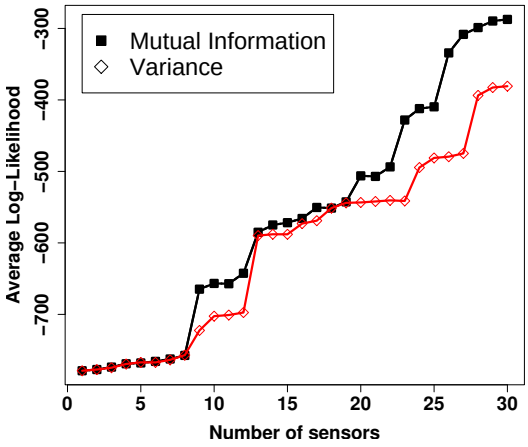
for the morning (between 8 am and 9 am) and noon (between 12 pm and 1 pm) in the Intel lab data. Figure 12(a) and Figure 12(b) plot the log-likelihood of the test set observations with increasing number of sensors for both models. Figure 12(e) presents the RMS error for a model estimated for the entire day. We can see that mutual information in almost all cases outperforms entropy, achieving better prediction accuracies with a smaller number of sensors.

Figure 12(f) presents the same results for the precipitation data set. Mutual information significantly outperforms entropy as a selection criterion—often several sensors would have to be additionally placed for entropy to reach the same level of prediction accuracy as mutual information. Figure 11(b) shows where both objective values would place sensors to measure precipitation. It can be seen that entropy is again much more likely to place sensors around the border of the sensing area than mutual information.

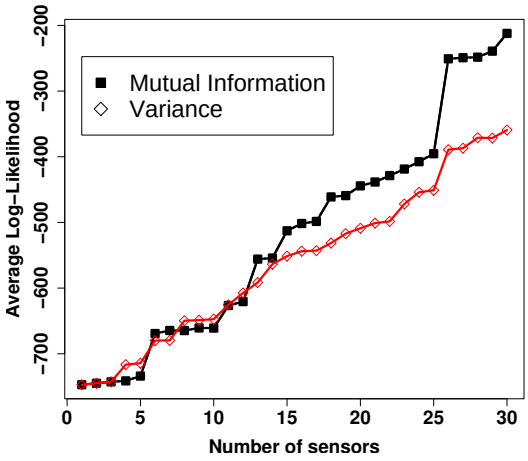


(a) *Morning RMS*

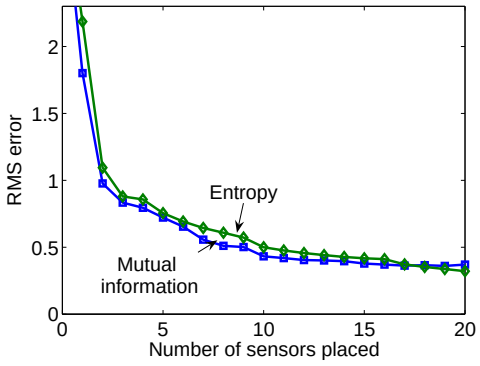
(b) *Noon RMS*



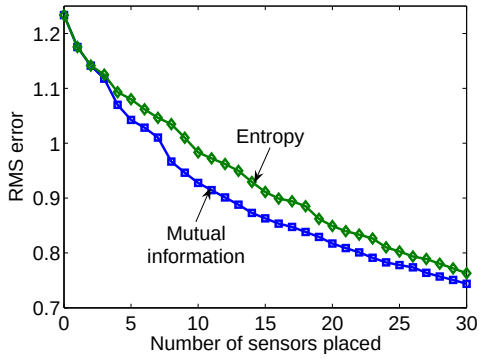
(c) *Morning log-likelihood*



(d) *Noon log-likelihood*



(e) *Entire day*



(f) *Precipitation RMS*

Figure 12: Prediction error and log-likelihood on test data for temperatures (a-e) and precipitation (f) in sensor network deployment, for an increasing number of sensors.

To gain further insight into the qualitative behavior of the selection criteria we learned a GP model using all sensors over one hour starting at noon. The model was fit with a isotropic Gaussian kernel and quadratic trend for the mean, using the *geoR* Toolkit (Ribeiro Jr. and Diggle, 2001). Fig-

ures 13(a) and 13(b) show the posterior mean and variance for the model. Using our algorithms, 22 sensors were chosen using the entropy and mutual information criteria. For each set of selected sensors, additional models were trained using only the measurements of the selected sensors. Predicted temperature surfaces for the entropy and mutual information configurations are presented in Figures 13(c) and 13(d). Entropy tends to favor placing sensors near the boundary as observed in Section 3, while mutual information tends to place the sensors on the top and bottom sides, which exhibited the most complexity and should have a higher sensor density. The predicted variances for each model are shown in figures 13(e) and 13(f). The mutual information version has significantly lower variance than the entropy version almost everywhere, displaying, as expected, higher variance in the unsensed areas in the center of the lab.

9.5 Comparison of Mutual Information with Classical Experimental Design Criteria

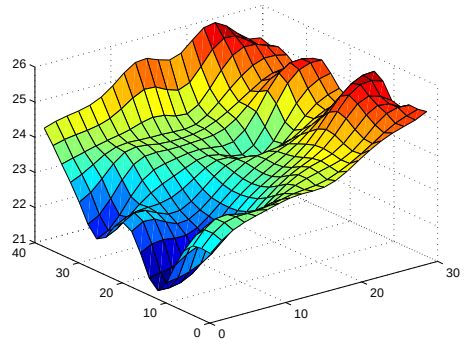
In order to compare the mutual information placements with the classical optimality criteria, we performed the following experiment. We uniformly selected 12 target locations $\mathcal{U}$ in the lab as locations of interest. We then set up the linear model

$$\mathbf{x}_{\mathcal{S}} = \Sigma_{\mathcal{S}\mathcal{U}}\Sigma_{\mathcal{U}\mathcal{U}}^{-1}\,\mathbf{x}_{\mathcal{U}} + \mathbf{w}.$$

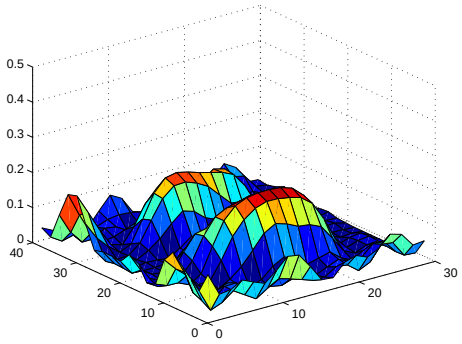
Hereby, $\mathbf{x}_{\mathcal{S}}$ denotes measurements at the locations $\mathcal{S}$, among which we choose our placement, $\mathbf{x}_{\mathcal{U}}$ are the values at the locations of interest (no sensors can be placed there), and $\mathbf{w}$ models independent normal measurement noise with constant variance. After subtraction of the training set mean, this model uses the Best Linear Unbiased (Kriging) estimator for predicting $\mathbf{x}_{\mathcal{S}}$ from $\mathbf{x}_{\mathcal{U}}$.

The problem becomes to select the sensor locations $\mathcal{A} \subset \mathcal{S}$ which allow most precise prediction of the variables of interest, in the sense of minimizing the error covariance $\frac{1}{\sigma^2}(A^TA)^{-1}$, where $A = \Sigma_{\mathcal{S}\mathcal{U}}\Sigma_{\mathcal{U}\mathcal{U}}^{-1}$. The different classical design criteria vary in how the scalarization of the error covariance is done. D-optimal design minimizes the log-determinant, A-optimal design minimizes the trace, and E-optimal design minimizes the spectral radius (the magnitude of the largest eigenvalue) of the error covariance. Note that this problem formulation favors the classical design criteria, which are tailored to minimize the error of predicting the values at the target locations $\mathcal{U}$, whereas mutual information and entropy just try to decrease the uncertainty in the entire space.

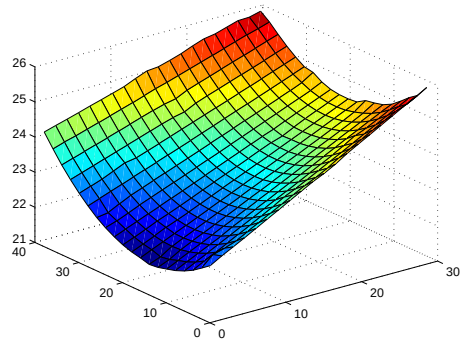
In order to solve the classical experimental design problems, we use the formulations as a semidefinite program (SDP) as discussed by Boyd and Vandenberghe (2004). We use SeDuMi (Sturm, 1999) for solving these SDPs. Since the integral optimization is hard, we solve the SDP relaxation to achieve a fractional design. This fractional solution defines the best way to distribute an infinite (or very large) budget of experiments to the different choices on the design menu (the variables in $\mathcal{S}$). In the sensor selection problem however, we have to solve the integral problem, since we face the binary decision of whether a sensor should be placed at a particular location or not. This is a hard combinatorial optimization problem. Since no near-optimal solution is known, we select the locations corresponding to the top k coefficients of the design menu, as is common practice. We compare the placements using the classical design criteria to those using the mutual information and entropy criteria, and evaluate each of them on the RMS prediction accuracy on the hold-out locations $\mathcal{U}$.



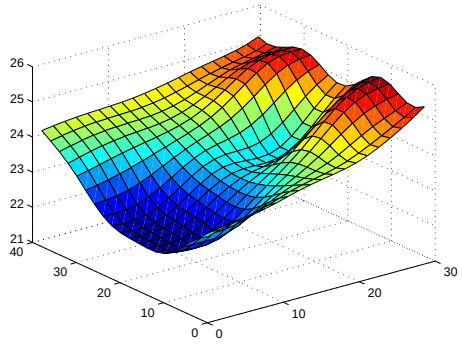
(a) *Isotropic model*



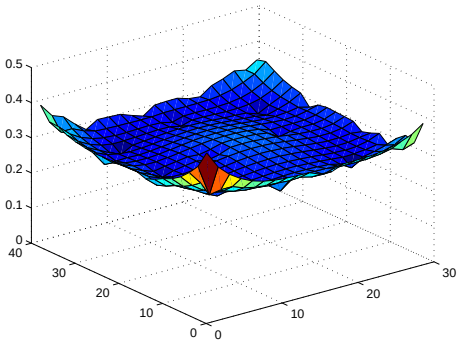
(b) *Predicted variance*



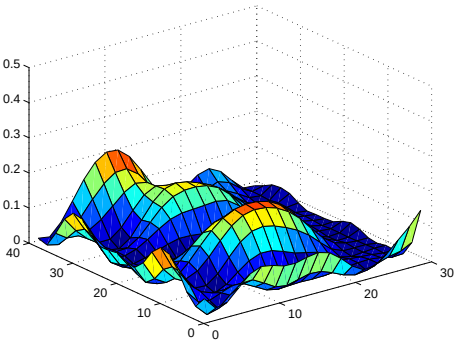
(c) *Temperature (entropy)*



(d) *Temperature (mutual inf.)*



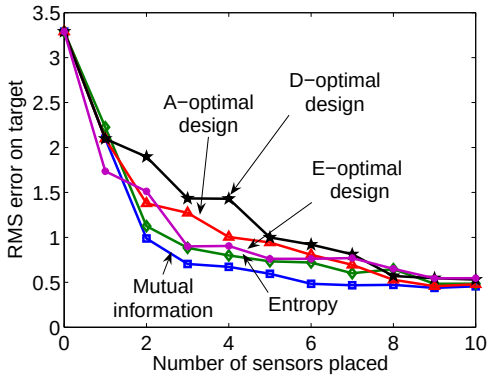
(e) *Variance (entropy)*



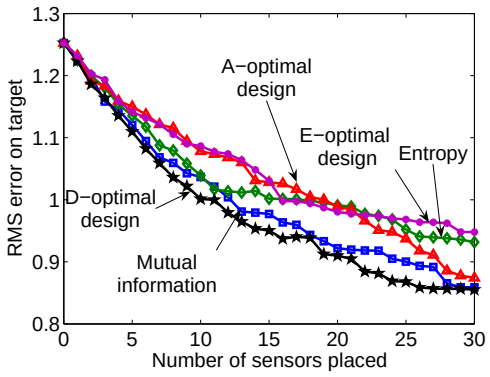
(f) *Variance (MI)*

Figure 13: Comparison of predictive quality of subsets selected using MI and entropy.

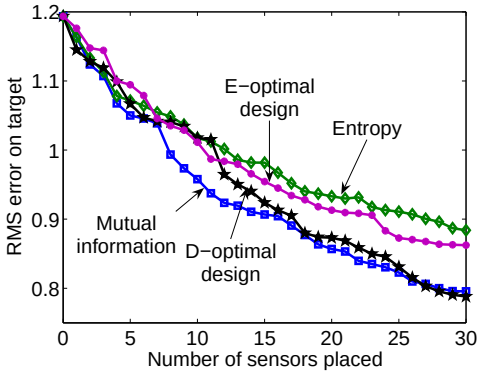
Figure 14(a) presents the results of this experiment on the temperature data. We can see that even though mutual information optimizes for prediction accuracy in the entire space and not specifically for the target locations $\mathcal{U}$, it incurs the least RMS prediction error, apart from the placements consisting only of a single sensor. E-optimal design performs comparably with the entropy criterion, and D- and A-optimality perform worse.



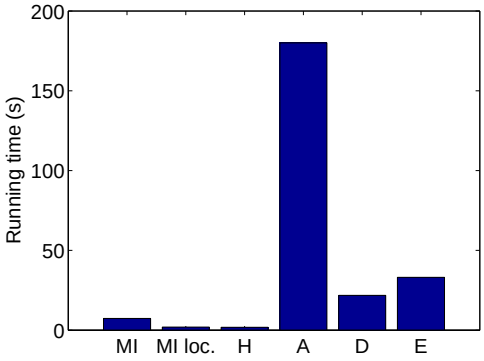
(a) *Comparison with A-, D- and E-optimality on temperature data*



(b) *Comparison with A-, D- and E-optimality on precipitation data, 111 node subsample*



(c) *Comparison with D- and E-optimality on precipitation data*

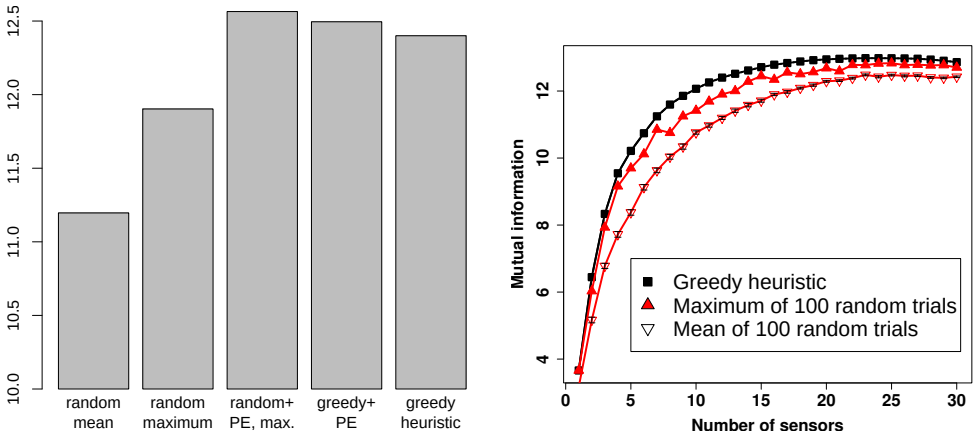


(d) *Running time for 111 node precipitation subsample*

Figure 14: Comparison with classical experimental design. We plot RMS prediction error on 25% hold out target locations $\mathcal{U}$.

When we performed the same experiment with the precipitation data, SeDuMi ran out of memory (1 GB) for the SDP required to solve the A-optimality criterion. The largest subsample we could solve for all A-, D- and E-optimality on this data set was limited to 111 locations. Figure 14(b) presents the results. For the entire data set of 167 locations, we could still solve the D- and E-optimality SDPs. The results are presented in Figure 14(c). We can observe that for the 111 locations, D-optimality slightly outperforms mutual information. We have to consider, however, that the classical criteria are optimized to minimize the error covariance with respect to the locations $\mathcal{U}$ of interest, whereas mutual information merely tries to achieve uniformly low uncertainty over the entire space. For the full set of 167 locations, mutual information outperforms the other design criteria.

Figure 14(d) presents the running time for optimizing A-, D-, E-optimality, and mutual information, mutual information with truncation parameter $\varepsilon = 1$ and entropy on the 111 node subsample of the precipitation data on a Pentium M 1.7 GHz processor. We can see that optimizing entropy is fastest, closely followed by the truncated mutual information criterion described in Section 5.2 that is further evaluated in Section 9.7. Even without truncation, optimizing mutual information is three times faster than (fractionally) optimizing D-optimality and 24 times faster than A-optimality.



(a) *Objective values* (b) *Comparison of heuristics*

Figure 15: Comparison of the greedy algorithm with several heuristics.

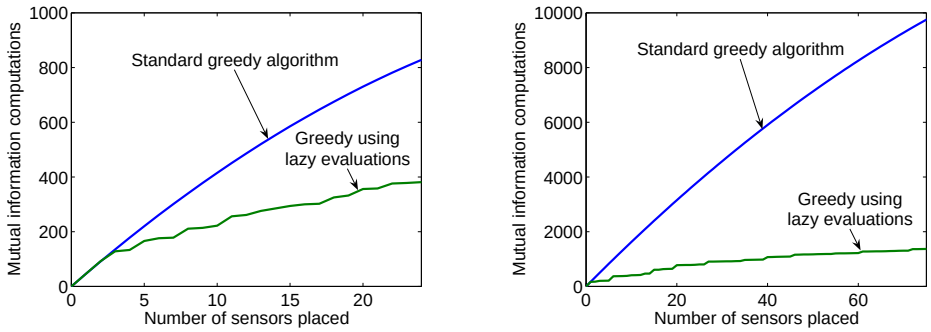
9.6 Empirical Analysis of the Greedy Algorithm

To study the effectiveness of the greedy algorithm, we compared the mutual information of the sets selected by our greedy algorithm to random selections, to a hill climbing method that uses a pairwise exchange heuristic, and—for small subsamples—to the bounds proved by the MIP as discussed in Section 4.5.

In this experiment, we used the empirical covariance matrix as the input to the algorithms. Figure 15(b) shows that the greedy algorithm provided significantly better results than the random selections, and even the maximum of a hundred random placements did not reach the quality of the greedy placements. Furthermore, we enhanced the random and greedy selections with the pairwise exchange (PE) heuristic, which iteratively finds exchanges of elements $y \in \mathcal{A}$ and $y' \in \mathcal{S} \setminus \mathcal{A}$ such that exchanging y and y' improves the mutual information score. Figure 15(a) presents objective values of these enhanced selection methods for a subset size of 12, for which the maximum over 100 random selections enhanced with PE actually exceeded the greedy score (unlike with most other subset sizes, where random + PE did about as well as the greedy algorithm). Typically, the objective values of random + PE, greedy + PE and greedy did not differ much. Note that as mentioned in Section 4, the performance guarantee for the greedy algorithm always provides an online approximation guarantee for the other heuristics.

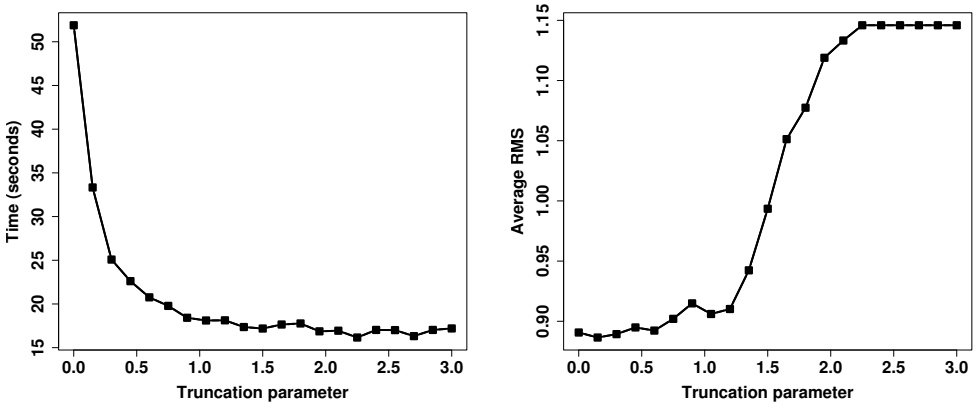
For a 16 node subsample of the temperature data set, we used the MIP from Section 4.5 to compute bounds on the optimal mutual information. Figure 5 presents the results. It can be seen, that for this small subsample, the greedy solution is never more than 5 percent away from the optimal solution, which is a much tighter bound than the a priori approximation factor of $(1-1/e)$.

We also experimented with the lazy evaluation strategy discussed in Section 5.1. For example when picking placements of size 50 for the precipitation data set, the number of mutual information computations decreased from 7125 to 1172, and the computation time on a Pentium M 1.7 GHz processor decreased from 41.3 seconds to 8.7 seconds. The results for both temperature and precipitation data sets are presented in Figures 16(a) and 16(b).



(a) *Temperature data* (b) *Precipitation data*

Figure 16: Performance improvements by using lazy evaluations of mutual information.



(a) *Running time* (b) *Approximation error*

Figure 17: Analysis of the experiments with local kernels. (a) running times for increasing level of truncation. (b) increase of average RMS error with increasing level of truncation. Note that for a truncation between 0.5 and 1.2, a good tradeoff between running time and error is achieved.

9.7 Results on Local Kernels

We also performed experiments to assess the running time versus quality trade-off incurred by using approximate local kernels. To provide intuition about the covariance structure, note that the 25, 50 and 75 percentiles of the absolute covariance entries were 0.122, 0.263 and 0.442, the maximum was 3.51, the minimum was $8.78E{-}6$. For the variance (the diagonal entries), the median was 1.70, and the minimum was 0.990. Figure 17(a) shows that the computation time can be drastically decreased as we increase the truncation parameter ε from 0 to the maximum variance. Figure 17(b) shows the RMS prediction accuracy for the 20 element subsets selected by Algorithm 3. According to the graphs, the range $\varepsilon \in [0.5, 1]$ seems to provide the appropriate trade-off between computation time and prediction accuracy.

In order to study the effect of local kernels on the placements, we performed the following experiment. We created a regular 7 by 7 grid with unit distance between neighboring grid points, and generated covariance matrices using two different GPs, one using the Gaussian (squared exponential) kernel, and the other using the local kernel (Equation 9). We exponentially increased the bandwidth in eight steps from 0.1 to 12.8. Figures 18 and 19 show the corresponding place-

ments using mutual information to select the locations. From this experiment, we can see that the placements obtained using the non-local Gaussian kernel tend to be spread out slightly more, as one might expect. Overall, however, the placements appear to be very similar. In light of the computational advantages provided by local kernels, these results provide further evidence in the spirit of Section 9.7, namely that local kernels can be a valuable tool for developing efficient model-based sensor placement algorithms.

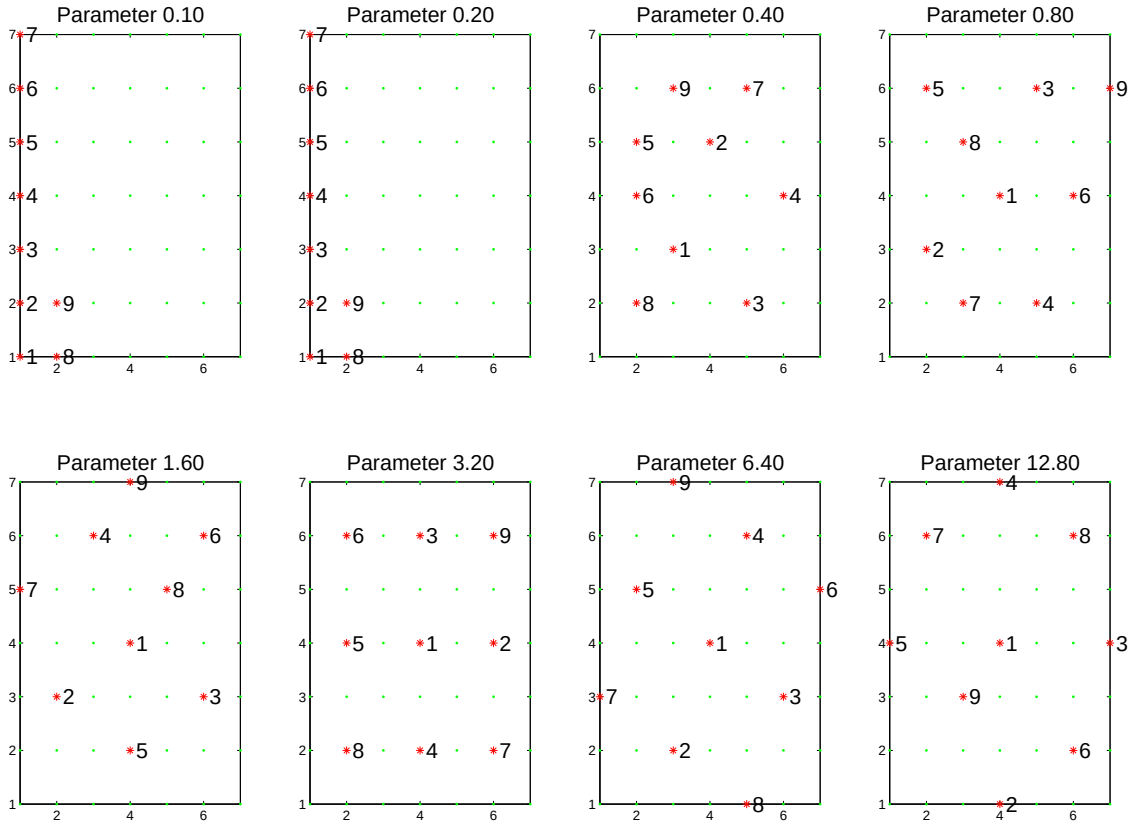


Figure 18: Placements under Gaussian kernel, mutual information criterion, increasing bandwidth

10. Future Work

There are several interesting possible extensions to the present work. Since the predictive variance in (2) does not depend on the actual observations, any closed-loop strategy which sequentially decides on the next location to measure, surprisingly, is equivalent to an open loop placement strategy which selects locations to make observations independently of the measured values. If there is uncertainty about the model parameters however, such as about the kernel bandwidths, then this is no longer true. In this case, we expect a sequential, closed-loop strategy to be more effective for predicting spatial phenomena. Krause and Guestrin (2007) present bounds comparing the performance of the optimal sequential strategy with the optimal fixed placement. This bound essentially depends on the

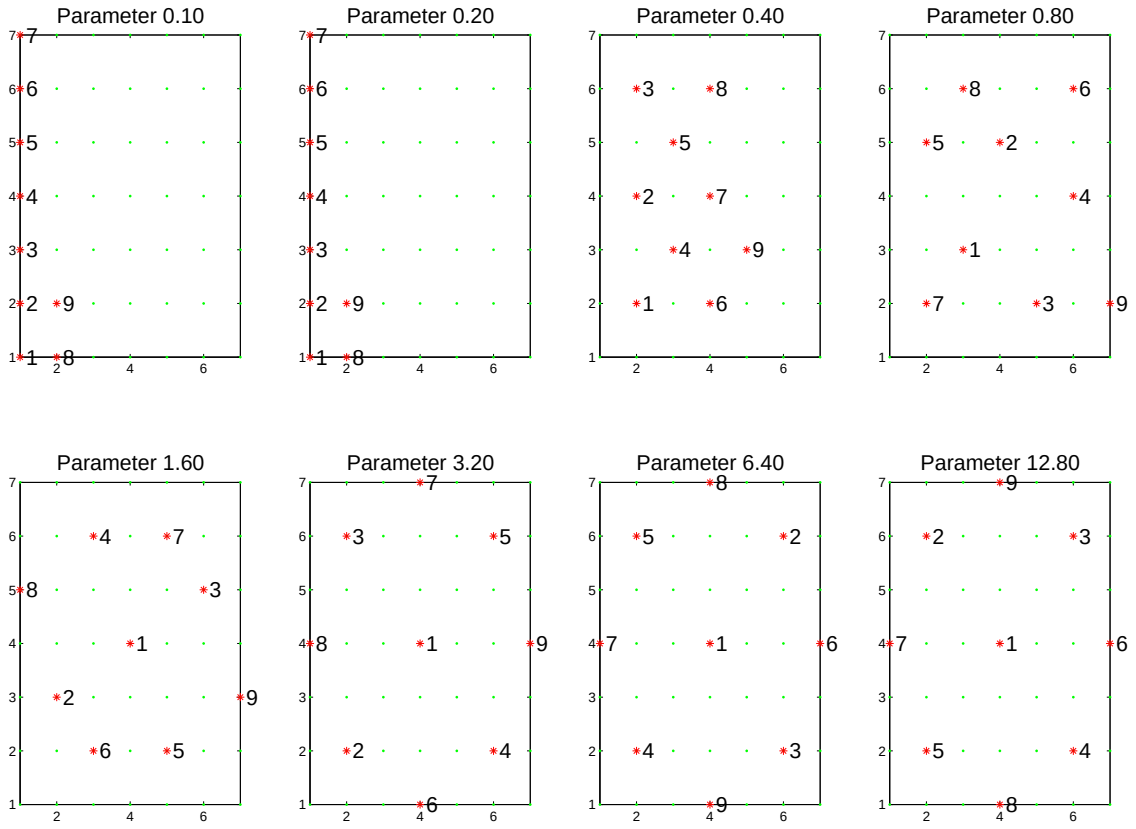


Figure 19: Placements under local kernel, mutual information criterion, increasing bandwidth

parameter entropy. We consider several exploration strategies for effectively reducing this parameter entropy and present sample complexity bounds. However, more work is needed in this area.

Another interesting open question is whether an approximation algorithm can be found for optimizing sensor placements subject to *submodular* cost functions—usually, the more sensors we have to buy, the cheaper they become per unit. To address this problem, Narasimhan and Bilmes (2006) present a submodular-supermodular procedure for bicriteria-optimization of a submodular function of which a submodular cost is subtracted. This procedure, while elegant, unfortunately can not provide approximation guarantees for this problem.

Of further interest are also constrained sensor placement problems, in which, for example, the placed sensors have to be connected in a routing tree, or have to lie on a collection of paths. Krause et al. (2006) provide evidence that submodularity can be leveraged to derive approximation algorithms for sensor placement even in these combinatorially even more challenging constrained optimization problems. However, there are still many open issues subject to further research.

11. Conclusions

In this paper, we tackle the problem of maximizing mutual information in order to optimize sensor placements. We prove that the exact optimization of mutual information is NP-complete, and provide an approximation algorithm that is within $(1-1/e)$ of the maximum mutual information configuration by exploiting the submodularity in the criterion. We also illustrate that submodularity can be used to obtain online bounds, which are useful for bounding the quality of the solutions obtained by any optimization method, and for designing branch and bound algorithms for the mutual information criterion. In order to scale up the application of our approach, show how to exploit lazy evaluations and local structure in GPs to provide significant speed-ups. We also extend our submodularity-based analysis of mutual information to incorporate robustness to sensor failures and model uncertainty.

Our very extensive empirical results indicate that data-driven placements can significantly improve the prediction accuracy over geometric models. We find, in contrast to previous work (Caselton et al., 1992; Zidek et al., 2000), that the mutual information criterion is often better than entropy and other classical experimental design criteria, both qualitatively and in prediction accuracy. In addition, the results show that a simple greedy algorithm for optimizing mutual information provides performance that is very close to the optimal solution in problems that are small enough to be solved exactly, and comparable to more complex heuristics in large problems.

We believe this work can be used to increase the efficacy of monitoring systems, and is a step towards well-founded active learning algorithms for spatial and structured data.

Acknowledgments

We would like to thank the Action Editor (Chris Williams) and the anonymous referees for their valuable comments and suggestions on this work. This work was partially supported by NSF Grant Nos. CNS-0509383, CNS-0625518 and a gift from Intel Corporation. Carlos Guestrin was partly supported by an Alfred P. Sloan Fellowship and an IBM Faculty Fellowship. Andreas Krause was partly supported by a Microsoft Research Graduate Fellowship. Ajit Singh acknowledges the support of NSERC.

Appendix A. Proofs

Proof [Theorem 2] Our reduction builds on the proof by Ko et al. (1995), who show that for any graph G, there exists a polynomially related, symmetric positive-definite matrix Σ such that Σ has a subdeterminant (of a submatrix resulting from the selection of k rows and columns $i_1,\ldots,i_k$) greater than some M if G has a clique of size at least k, and Σ does not have a subdeterminant greater than $M-\varepsilon$ for some (polynomially-large) $\varepsilon > 0$ if G does not have such a clique. Let G be a graph, and let Σ be the matrix constructed in Ko et al. (1995). We will consider Σ as the covariance matrix of a multivariate Gaussian distribution with variables $\mathcal{X}_U = \{X_1,\ldots,X_n\}$. Introduce additional variables $\mathcal{X}_S = \{y_1,\ldots,y_n\}$ such that $y_i|X_i = x \sim \mathcal{N}(x,\sigma^2)$. Note that a subset

$\mathcal{A} \subseteq \mathcal{S}$, $|\mathcal{A}| = k$, has maximum entropy of all such subsets if and only if the parents $\Gamma_{\mathcal{A}} \subset \mathcal{U}$ of $\mathcal{A}$ have maximum entropy among all such subsets of $\mathcal{U}$. Now note that $I(\mathcal{A}; (\mathcal{U} \cup \mathcal{S}) \setminus \mathcal{A}) = H(\mathcal{A}) - H(\mathcal{A} \mid (\mathcal{U} \cup \mathcal{S}) \setminus \mathcal{A}) = H(\mathcal{A}) - H(\mathcal{A} \mid \mathcal{U})$, because y_i and y_j are conditionally independent given $\mathcal{U}$. Furthermore, again because of independence, $H(\mathcal{A} \mid \mathcal{U})$ is a constant only depending on the cardinality of $\mathcal{A}$. Assume we could decide efficiently whether there is a subset $\mathcal{A} \subset \mathcal{S}$ such that $I(\mathcal{A}; \mathcal{V} \setminus \mathcal{A}) \geq M'$. If we choose σ^2 small enough, then this would allow us to decide whether G has a clique of size k, utilizing the gap ε. ■

Proof [Lemma 5] Define $\hat{\mathcal{K}}(x,y) = \mathcal{K}(x,y)$ for $x \neq y$ and $\hat{\mathcal{K}}(x,x) = \mathcal{K}(x,x) + \sigma^2$ to include the sensor noise σ^2. Since $\mathcal{C}$ is compact and $\mathcal{K}$ continuous, $\mathcal{K}$ is uniformly continuous over $\mathcal{C}$. Hence, for any ε_1, there exists a δ_1 such that for all $x, x', y, y', \|x - x'\|_2 \leq \delta_1, \|y - y'\|_2 \leq \delta_1$ it holds that $|\mathcal{K}(x,y) - \mathcal{K}(x',y')| \leq \varepsilon_1$. Assume $\mathcal{C}_1 \subset \mathcal{C}$ is a finite mesh grid with mesh width $2\delta_1$. We allow sensor placement only on grid $\mathcal{C}_1$. Let $\mathcal{C}_2 \subset \mathcal{C}$ be a mesh grid of mesh width $2\delta_1$, which is derived by translating $\mathcal{C}_1$ by δ_1 in Euclidean norm, and let G_1, G_2 denote the restriction of the GP G to $\mathcal{C}_1, \mathcal{C}_2$. We assume $\mathcal{C}_1, \mathcal{C}_2$ cover $\mathcal{C}$ in the sense of compactness. We use the notation $\tilde{\cdot}$ to refer to the translated version in G_2 of the random variable $\cdot$ in G_1. $\hat{\mathcal{K}}$ is a symmetric strictly positive definite covariance function and $|\hat{\mathcal{K}}(X,y) - \hat{\mathcal{K}}(\tilde{X}, \tilde{y})| \leq \varepsilon_1$ for all $X, y \in G_1$. Moreover, since $\mathcal{K}$ is positive semidefinite, the smallest eigenvalue of any covariance matrix derived from $\hat{\mathcal{K}}$ is at least σ^2.

Let $\mathcal{A}$ be a subset of $\mathcal{C}_1$ and $X \in \mathcal{C}_1 \setminus \mathcal{A}$. Using (5), we first consider the conditional variance $\sigma^2_{X|\mathcal{A}}$. By definition, $\|y - \tilde{y}\|_2 \leq \delta_1$, and hence $|\hat{\mathcal{K}}(X,y) - \hat{\mathcal{K}}(X,\tilde{y})| \leq \varepsilon_1$ for all $y \in \mathcal{A}$. Hence we know that $\|\Sigma_{\mathcal{A}\mathcal{A}} - \Sigma_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\|_2 \leq \|\Sigma_{\mathcal{A}\mathcal{A}} - \Sigma_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\|_F \leq k^2\varepsilon_1$. We furthermore note that $\|\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\|_2 = \lambda^{max}(\Sigma^{-1}_{\mathcal{A}\mathcal{A}}) = \lambda^{min}(\Sigma_{\mathcal{A}\mathcal{A}})^{-1} \leq \sigma^{-2}$, and hence

$$\begin{aligned}\|\Sigma^{-1}_{\mathcal{A}\mathcal{A}} - \Sigma^{-1}_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\|_2 &= \|\Sigma^{-1}_{\mathcal{A}\mathcal{A}}(\Sigma_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}} - \Sigma_{\mathcal{A}\mathcal{A}})\Sigma^{-1}_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\|_2 \\ &\leq \|\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\|_2 \|\Sigma_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}} - \Sigma_{\mathcal{A}\mathcal{A}}\|_2 \|\Sigma^{-1}_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\|_2 \leq \sigma^{-4}k^2\varepsilon_1.\end{aligned}$$

We derive $\|\Sigma_{X\tilde{\mathcal{A}}} - \Sigma_{X\mathcal{A}}\|_2 \leq \|\varepsilon_1 \mathbf{1}^T\|_2 = \varepsilon_1\sqrt{k}$, hence

$$\begin{aligned}|\sigma^2_{X|A} - \sigma^2_{X|\tilde{\mathcal{A}}}| &= |\Sigma_{X\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\Sigma_{\mathcal{A}X} - \Sigma_{X\tilde{\mathcal{A}}}\Sigma^{-1}_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\Sigma_{\tilde{\mathcal{A}}X}| \\ &\leq 2\|\Sigma_{X\mathcal{A}} - \Sigma_{X\tilde{\mathcal{A}}}\|_2 \|\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\|_2 \|\Sigma_{X\mathcal{A}}\|_2 + \|\Sigma^{-1}_{\mathcal{A}\mathcal{A}} - \Sigma^{-1}_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\|_2 \|\Sigma_{X\mathcal{A}}\|_2^2 + O(\varepsilon_1^2) \\ &\leq 2\varepsilon_1\sqrt{k}\sigma^{-2}M\sqrt{k} + \sigma^{-4}k^2\varepsilon_1 M^2 k + O(\varepsilon_1^2) \\ &\leq \varepsilon_1 k\sigma^{-2}M\left(2 + \sigma^{-2}k^2M\right) + O(\varepsilon_1^2),\end{aligned}$$

where $M = \max_{x \in C} \mathcal{K}(x,x)$. We choose δ such that the above difference is bounded by $\sigma^2\varepsilon$. We note that (assuming w.l.o.g. $H(X \mid \mathcal{A}) \geq H(X \mid \tilde{\mathcal{A}})$)

$$H(X \mid \mathcal{A}) - H(X \mid \tilde{\mathcal{A}}) = \frac{1}{2}\log\frac{\sigma^2_{X|\mathcal{A}}}{\sigma^2_{X|\tilde{\mathcal{A}}}} \leq \frac{\log(1+\varepsilon)}{2} \leq \frac{\varepsilon}{2}.$$

which concludes the argument. ■

Proof [Corollary 6] The higher order terms $O(\varepsilon_1^2)$ can be worked out as $k\sigma^{-2}\varepsilon^2(1 + Mk^2\sigma^{-2} + \varepsilon k^2\sigma^{-2})$. Assuming that $\varepsilon < \min(M, 1)$, this is bounded by $3k^3M\sigma^{-4}\varepsilon$. Using the Lipschitz assumption, we can directly compute δ_1 from ε_1 in the above proof, by letting $\delta = \varepsilon_1/L$. Let $R =$

$k\sigma^{-2}M\left(2+\sigma^{-2}k^2M\right)+3k^3M\sigma^{-4}$. We want to choose δ such that $\varepsilon_1 R \leq \sigma^2\varepsilon$. Hence if we choose $\delta \leq \frac{\sigma^2\varepsilon}{LR}$, then $|H(X \mid \mathcal{A}) - H(X \mid \tilde{\mathcal{A}})| \leq \varepsilon$ uniformly as required. Note that in order to apply the result from Nemhauser et al. (1978), the approximate monotonicity has to be guaranteed for subsets of size $2k$, which results in the stated bound. ■

Proof [Theorem 7] The following proof is an extension of the proof by Nemhauser et al. (1978), using some simplifications by Jon Kleinberg.

Let $s_1,\ldots,s_k$ be the locations selected by the greedy algorithm. Let $\mathcal{A}_i = \{s_1,\ldots,s_i\}$, $\mathcal{A}^*$ be the optimal solution, and $\delta_i = \mathrm{MI}(\mathcal{A}_i) - \mathrm{MI}(\mathcal{A}_{i-1})$. By *Lemma* 5, we have, for all $1 \leq i \leq k$,

$$\mathrm{MI}(\mathcal{A}_i \cup \mathcal{A}^*) \geq \mathrm{MI}(\mathcal{A}^*) - k\varepsilon.$$

We also have, for $0 \leq i < k$,

$$\mathrm{MI}(\mathcal{A}_i \cup \mathcal{A}^*) \leq \mathrm{MI}(\mathcal{A}_i) + k\delta_{i+1} = \sum_{j=1}^{i} \delta_j + k\delta_{i+1}.$$

Hence we have the following sequence of inequalities:

$$\begin{aligned}
\mathrm{MI}(\mathcal{A}^*) - k\varepsilon &\leq k\delta_1 \\
\mathrm{MI}(\mathcal{A}^*) - k\varepsilon &\leq \delta_1 + k\delta_2 \\
&\vdots \\
\mathrm{MI}(\mathcal{A}^*) - k\varepsilon &\leq \sum_{j=1}^{k-1} \delta_j + k\delta_k.
\end{aligned}$$

Now we multiply both sides of the i-th inequality by $\left(1-\frac{1}{k}\right)^{k-1}$, and add all inequalities up. After cancellation, we get

$$\left(\sum_{i=0}^{k-1} (1-1/k)^i\right)(\mathrm{MI}(\mathcal{A}^*) - k\varepsilon) \leq k\sum_{i=1}^{k} \delta_i = k\,\mathrm{MI}(\mathcal{A}_k).$$

Hence, as claimed, with $\mathcal{A}_G = \mathcal{A}_k$ (i.e., $\mathcal{A}_G$ is the k-element greedy solution)

$$\mathrm{MI}(\mathcal{A}_G) \geq \left(1-(1-1/k)^k\right)(\mathrm{MI}(\mathcal{A}^*) - k\varepsilon) \geq (1-1/e)(\mathrm{MI}(\mathcal{A}^*) - k\varepsilon).$$

■

Proof [Remark 12] We have that $H(y \mid Z) < 0 \Leftrightarrow \mathcal{K}(y,y) + \sigma^2 - \frac{\mathcal{K}(Z,y)^2}{\mathcal{K}(Z,Z)+\sigma^2} < \frac{1}{2\pi e}$. Using a similar argument as the proof of Lemma 5, for very fine discretizations, there exists a y arbitrarily close to Z, such that for any $\alpha > 0$, $|\mathcal{K}(Z,Z) - \mathcal{K}(y,y)| \leq \alpha$ and $|\mathcal{K}(Z,Z) - \mathcal{K}(Z,y)| \leq \alpha$. Plugging these bounds into the definition of $H(y \mid Z)$ and some algebraic manipulation proves the claim. ■

References

A. C. Atkinson. Recent developments in the methods of optimum and related experimental designs. *International Statistical Review / Revue Internationale de Statistique*, 56(2):99–115, Aug. 1988.

A. C. Atkinson. The usefulness of optimum experimental designs. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):59–76, 1996.

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.

X. Bai, S. Kumar, Z. Yun, D. Xuan, and T. H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Florence, Italy, 2006.

J. M. Bernardo. Expected information as expected utility. *Annals of Statistics*, 7(3):686–690, May 1979.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge UP, March 2004.

W. F. Caselton and T. Hussain. Hydrologic networks: Information transmission. *Journal of Water Resources Planning and Management*, WR2:503–520, 1980.

W. F. Caselton and J. V. Zidek. Optimal monitoring network designs. *Statistics and Probability Letters*, 2(4):223–227, 1984.

W. F. Caselton, L. Kan, and J. V. Zidek. *Statistics in the Environmental and Earth Sciences*, chapter Quality data networks that minimize entropy, pages 10–38. Halsted Press, 1992.

K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3): 273–304, Aug. 1995. ISSN 08834237.

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.

R. D. Cook and C. J. Nachtsheim. A comparison of algorithms for constructing exact D-optimal designs. *Technometrics*, 22(3):315–324, Aug. 1980. ISSN 00401706.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley Interscience, 1991.

N. A. C. Cressie. *Statistics for Spatial Data*. Wiley, 1991.

C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416):953–963, 1991.

A. Das and D. Kempe. Algorithms for subset selection in linear regression. In *Symposium on the Theory of Computing*, 2008.

A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB)*, 2004.

V. Federov and W. Mueller. Comparison of two approaches in the optimal design of an observation network. *Statistics*, 20:339–351, 1989.

V. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, 1972. Trans. W. J. Studden and E. M. Klimko.

P. Flaherty, M. Jordan, and A. Arkin. Robust design of biological experiments. In *Advances in Neural Information Processing Systems (NIPS) 19*, 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.

G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins, 1989.

H. H. Gonzalez-Banos and J. Latombe. A randomized art-gallery algorithm for sensor placement. In *Proc. 17th ACM Symposium on Computational Geometry*, pages 232–240, 2001.

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.

P. Guttorp, N. D. Le, P. D. Sampson, and J. V. Zidek. Using entropy in the redesign of an environmental monitoring network. Technical report, Department of Statistics. University of British Columbia., 1992. Tech. Rep. 116.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2003.

N. Heo and P. K. Varshney. Energy-efficient deployment of intelligent mobile sensor networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35(1):78–92, 2005.

D. S. Hochbaum and W. Maas. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32:130–136, 1985.

H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

A. Howard, M. Mataric, and G. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem, 2002.

R. Kershner. The number of circles covering a set. *American Journal of Mathematics*, 61:665–671, 1939.

C. Ko, J. Lee, and M. Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995.

A. Krause and C. Guestrin. A note on the budgeted maximization of submodular functions. Technical report, CMU-CALD-05-103, 2005.

A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: An exploration–exploitation approach. In *International Conference on Machine Learning*, 2007.

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.

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.

U. Lerner and R. Parr. Inference in hybrid networks: Theoretical limits and practical algorithms. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI)*, 2001.

D. V. Lindley. On a measure of the information provided by an experiment. *Annals of Mathematical Statistics*, 27:986–1005, 1956.

D. V. Lindley and A. F. M. Smith. Bayes' estimates for the linear model. *Journal of the Royal Statistical Society, Ser. B*, 34:1–18, 1972.

S. P. Luttrell. The use of transinformation in the design of data sampling schemes for inverse problems. *Inverse Problems*, 1:199–218, 1985.

D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge UP, 2003.

G. P. McCabe. Principal variables. *Technometrics*, 26(2):137–144, 1984.

M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2): 239–245, May 1979.

R. K. Meyer and C. J. Nachtsheim. Constructing exact D-optimal experimental designs by simulated annealing. *American Journal of Mathematical and Management Sciences*, 8(3-4):329–359, 1988.

T. J. Mitchell. An algorithm for the construction of "D-optimal" experimental designs. *Technometrics*, 16(2):203–210, May 1974a. ISSN 00401706.

T.J. Mitchell. Computer construction of "D-optimal" first-order designs. *Technometrics*, 16(2): 211–220, May 1974b. ISSN 00401706.

Avidan Moghaddam, Weiss. Fast pixel/part selection with sparse eigenvectors. In *International Conference on Computer Vision*, 2007.

B. Moghaddam, Y. Weiss, and S. Avidan. Spectral bounds for sparse PCA: Exact and greedy algorithms. In *Advances in Neural Information Processing Systems (NIPS) 18*, 2005.

B. Moghaddam, Y. Weiss, and S. Avidan. Generalized spectral bounds for sparse LDA. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML)*, 2006.

M. Narasimhan and J. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Advances in Neural Information Processing Systems (NIPS) 19*, 2006.

G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

G. L. Nemhauser and L. A. Wolsey. *Studies on Graphs and Discrete Programming*, chapter Maximizing submodular set functions: Formulations and analysis of algorithms, pages 279–301. North-Holland, 1981.

N-.K Nguyen and A. J. Miller. A review of some exchange algorithms for constructing discrete D-optimal designs. *Computational Statistics and Data Analysis*, 14:489–498, 1992.

D. J. Nott and W. T. M. Dunsmuir. Estimation of nonstationary spatial covariance structure. *Biometrika*, 89:819–829, 2002.

A. O'Hagan. Curve fitting and optimal design for prediction (with discussion). *Journal of the Royal Statistical Society, Ser. B*, 40:1–42, 1978.

C. J. Paciorek. *Nonstationary Gaussian Processes for Regression and Spatial Modelling*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, May 2003.

L. Paninski. Asymptotic theory of information-theoretic experimental design. *Neural Computation*, 17(7):1480–1507, 2005.

F. Pukelsheim. Information increasing orderings in experimental design theory. *International Statistical Review / Revue Internationale de Statistique*, 55(2):203–219, Aug. 1987. ISSN 03067734.

N. Ramakrishnan, C. Bailey-Kellogg, S. Tadepalli, and V. N. Pandey. Gaussian processes for active data mining of spatial aggregates. In *SIAM Data Mining*, 2005.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Process for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2006.

P. J. Ribeiro Jr. and P. J. Diggle. geoR: A package for geostatistical analysis. R-NEWS Vol 1, No 2. ISSN 1609-3631, 2001.

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.

J. Sacks, S. B. Schiller, and W. J. Welch. Designs for computer experiments. *Technometrics*, 31(1): 41–47, Feb. 1989. ISSN 00401706.

P. Sebastiani and H. P. Wynn. Maximum entropy sampling and optimal bayesian experimental design. *Journal of the Royal Statistical Society, Series B*, 62(1):145–157, 2000.

M. Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14 (2):69–106, 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.

M. C. Shewry and H. P. Wynn. Maximum entropy sampling. *Journal of Applied Statistics*, 14: 165–170, 1987.

A. Singh, A. Krause, C. Guestrin, W. Kaiser, and M. Batalin. Efficient planning of informative paths for multiple robots. In *IJCAI*, 2007.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS) 18*, 2005.

P. Sollich. Learning from minimum entropy queries in a large committee machine. *Physical Review E*, 53:R2060–R2063, 1996.

A. J. Storkey. Truncated covariance matrices and Toeplitz methods in Gaussian processes. In *Artificial Neural Networks - ICANN 1999*, 1999.

J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software, special issue on interior-point methods*, 11(12):625–653, 1999.

M. Sviridenko. A note on maximizing a submodular set function subject to knapsack constraint. *Operations Research Letters*, 32:41–43, 2004.

S. Toumpis and G. A. Gupta. Optimal placement of nodes in large sensor networks under a general physical layer model. In *Proc. IEEE Communications Society Conference on Sensor and Ad Hoc Communications (SECON)*, 2005.

W. J. Welch. Branch-and-bound search for experimental design based on D-optimality and other criteria. *Technometrics*, 24(1):41–48, 1982.

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.

S. Wu and J. V. Zidek. An entropy based review of selected NADP/NTN network sites for 1983–86. *Atmospheric Environment*, 26A:2089–2103, 1992.

D. Ylvisaker. *A Survey of Statistical Design and Linear Models*, chapter Design on random fields. North-Holland, 1975.

K. Yu, J. Bi, and V. Tresp. Active learning via transductive experimental design. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML)*, 2006.

Z. Zhu and M. L. Stein. Spatial sampling design for prediction with estimated parameters. *Journal of Agricultural, Biological and Environmental Statistics*, 11:24–49, 2006.

J. V. Zidek, W. Sun, and N. D. Le. Designing and integrating composite networks for monitoring multivariate gaussian pollution fields. *Applied Statistics*, 49:63–79, 2000.

D. L. Zimmerman. Optimal network design for spatial prediction, covariance parameter estimation, and empirical prediction. *Environmetrics*, 17(6):635–652, 2006.

Journal of Machine Learning Research 9 (2008) 285-312 Submitted 2/07; Revised 9/07; Published 2/08

Support Vector Machinery for Infinite Ensemble Learning

Hsuan-Tien Lin HTLIN@CALTECH.EDU
Ling Li LING@CALTECH.EDU
Department of Computer Science
California Institute of Technology
Pasadena, CA 91125, USA

Editor: Peter L. Bartlett

Abstract

Ensemble learning algorithms such as boosting can achieve better performance by averaging over the predictions of some base hypotheses. Nevertheless, most existing algorithms are limited to combining only a finite number of hypotheses, and the generated ensemble is usually sparse. Thus, it is not clear whether we should construct an ensemble classifier with a larger or even an infinite number of hypotheses. In addition, constructing an infinite ensemble itself is a challenging task. In this paper, we formulate an infinite ensemble learning framework based on the support vector machine (SVM). The framework can output an infinite and nonsparse ensemble through embedding infinitely many hypotheses into an SVM kernel. We use the framework to derive two novel kernels, the stump kernel and the perceptron kernel. The stump kernel embodies infinitely many decision stumps, and the perceptron kernel embodies infinitely many perceptrons. We also show that the Laplacian radial basis function kernel embodies infinitely many decision trees, and can thus be explained through infinite ensemble learning. Experimental results show that SVM with these kernels is superior to boosting with the same base hypothesis set. In addition, SVM with the stump kernel or the perceptron kernel performs similarly to SVM with the Gaussian radial basis function kernel, but enjoys the benefit of faster parameter selection. These properties make the novel kernels favorable choices in practice.

Keywords: ensemble learning, boosting, support vector machine, kernel

1. Introduction

Ensemble learning algorithms, such as boosting (Freund and Schapire, 1996), are successful in practice (Meir and Rätsch, 2003). They construct a classifier that averages over some base hypotheses in a set $\mathcal{H}$. While the size of $\mathcal{H}$ can be infinite, most existing algorithms use only a finite subset of $\mathcal{H}$, and the classifier is effectively a finite ensemble of hypotheses. Some theories show that the finiteness places a restriction on the capacity of the ensemble (Freund and Schapire, 1997), and some theories suggest that the performance of boosting can be linked to its asymptotic behavior when the ensemble is allowed to be of an infinite size (Rätsch et al., 2001). Thus, it is possible that an infinite ensemble is superior for learning. Nevertheless, the possibility has not been fully explored because constructing such an ensemble is a challenging task (Vapnik, 1998).

In this paper, we conquer the task of infinite ensemble learning, and demonstrate that better performance can be achieved by going from finite ensembles to infinite ones. We formulate a framework for infinite ensemble learning based on the support vector machine (SVM) (Vapnik, 1998). The key of the framework is to embed an infinite number of hypotheses into an SVM kernel.

Such a framework can be applied both to construct new kernels for SVM, and to interpret some existing ones (Lin, 2005). Furthermore, the framework allows us to compare SVM and ensemble learning algorithms in a fair manner using the same base hypothesis set.

Based on the framework, we derive two novel SVM kernels, the stump kernel and the perceptron kernel, from an ensemble learning perspective (Lin and Li, 2005a). The stump kernel embodies infinitely many decision stumps, and as a consequence measures the similarity between examples by the ℓ_1-norm distance. The perceptron kernel embodies infinitely many perceptrons, and works with the ℓ_2-norm distance. While there exist similar kernels in literature, our derivation from an ensemble learning perspective is nevertheless original. Our work not only provides a feature-space view of their theoretical properties, but also broadens their use in practice. Experimental results show that SVM with these kernels is superior to successful ensemble learning algorithms with the same base hypothesis set. These results reveal some weakness in traditional ensemble learning algorithms, and help understand both SVM and ensemble learning better. In addition, SVM with these kernels shares similar performance to SVM with the popular Gaussian radial basis function (Gaussian-RBF) kernel, but enjoys the benefit of faster parameter selection. These properties make the two kernels favorable choices in practice.

We also show that the Laplacian-RBF kernel embodies infinitely many decision trees, and hence can be viewed as an instance of the framework. Experimentally, SVM with the Laplacian-RBF kernel performs better than ensemble learning algorithms with decision trees. In addition, our derivation from an ensemble learning perspective helps to explain the success of the kernel on some specific applications (Chapelle et al., 1999).

The paper is organized as follows. In Section 2, we review the connections between SVM and ensemble learning. Next in Section 3, we propose the framework for embedding an infinite number of hypotheses into a kernel. We then derive the stump kernel in Section 4, the perceptron kernel in Section 5, and the Laplacian-RBF kernel in Section 6. Finally, we show the experimental results in Section 7, and conclude in Section 8.

2. Support Vector Machine and Ensemble Learning

In this section, we first introduce the basics of SVM and ensemble learning. Then, we review some established connections between the two in literature.

2.1 Support Vector Machine

Given a training set $\{(x_i,y_i)\}_{i=1}^N$, which contains input vectors $x_i \in \mathcal{X} \subseteq \mathbb{R}^D$ and their corresponding labels $y_i \in \{-1,+1\}$, the soft-margin SVM (Vapnik, 1998) constructs a classifier

$$g(x) = \text{sign}\big(\langle w, \phi_x \rangle + b\big)$$

from the optimal solution to the following problem:[1]

$$\begin{aligned} (P_1) \quad \min_{w\in\mathcal{F},b\in\mathbb{R},\xi\in\mathbb{R}^N} \quad & \frac{1}{2}\langle w,w\rangle + C\sum_{i=1}^N \xi_i & \\ \text{s.t.} \quad & y_i\big(\langle w,\phi_{x_i}\rangle + b\big) \geq 1-\xi_i, & \text{for } i=1,2,\ldots,N, \\ & \xi_i \geq 0, & \text{for } i=1,2,\ldots,N. \end{aligned}$$

1. When η is nonzero, $\text{sign}(\eta) \equiv \frac{\eta}{|\eta|}$. We shall let $\text{sign}(0) \equiv 0$ to make some mathematical setup cleaner.

Here $C > 0$ is the regularization parameter, and $\phi_x = \Phi(x)$ is obtained from the feature mapping $\Phi\colon \mathcal{X} \to \mathcal{F}$. We assume the feature space $\mathcal{F}$ to be a Hilbert space equipped with the inner product $\langle \cdot, \cdot \rangle$ (Schölkopf and Smola, 2002). Because $\mathcal{F}$ can be of an infinite number of dimensions, SVM solvers usually work on the dual problem:

$$\begin{aligned} (P_2) \quad \min_{\lambda \in \mathbb{R}^N} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j \mathcal{K}(x_i,x_j) - \sum_{i=1}^{N}\lambda_i \\ \text{s.t.} \quad & 0 \le \lambda_i \le C, \qquad \text{for } i = 1,2,\ldots,N, \\ & \sum_{i=1}^{N} y_i\lambda_i = 0. \end{aligned}$$

Here $\mathcal{K}$ is the kernel function defined as $\mathcal{K}(x,x') = \langle \phi_x, \phi_{x'} \rangle$. Then, the optimal classifier becomes

$$g(x) = \operatorname{sign}\left(\sum_{i=1}^{N} y_i\lambda_i \mathcal{K}(x_i,x) + b\right), \tag{1}$$

where b can be computed through the primal-dual relationship (Vapnik, 1998; Schölkopf and Smola, 2002).

The use of a kernel function $\mathcal{K}$ instead of computing the inner product directly in $\mathcal{F}$ is called the kernel trick, which works when $\mathcal{K}(\cdot,\cdot)$ can be computed efficiently (Schölkopf and Smola, 2002). Alternatively, we can begin with an arbitrary $\mathcal{K}$, and check whether there exists a space-mapping pair $(\mathcal{F},\Phi)$ such that $\mathcal{K}(\cdot,\cdot)$ is a valid inner product in $\mathcal{F}$. A key tool here is the *Mercer's condition*, which states that a symmetric $\mathcal{K}(\cdot,\cdot)$ is a valid inner product if and only if its Gram matrix K, defined by $K_{i,j} = \mathcal{K}(x_i,x_j)$, is always positive semi-definite (PSD) (Vapnik, 1998; Schölkopf and Smola, 2002).

The soft-margin SVM originates from the hard-margin SVM, which forces the margin violations ξ_i to be zero. When such a solution is feasible for (P_1), the corresponding dual solution can be obtained by setting C to ∞ in (P_2).

2.2 Adaptive Boosting and Linear Programming Boosting

The adaptive boosting (AdaBoost) algorithm (Freund and Schapire, 1996) is perhaps the most popular and successful approach for ensemble learning. For a given integer T and a hypothesis set $\mathcal{H}$, AdaBoost iteratively selects T hypotheses $h_t \in \mathcal{H}$ and weights $w_t \ge 0$ to construct an ensemble classifier

$$g_T(x) = \operatorname{sign}\left(\sum_{t=1}^{T} w_t h_t(x)\right).$$

The underlying algorithm for selecting $h_t \in \mathcal{H}$ is called a base learner. Under some assumptions (Rätsch et al., 2001), it is shown that when $T \to \infty$, AdaBoost asymptotically approximates an infinite ensemble classifier

$$g_\infty(x) = \operatorname{sign}\left(\sum_{t=1}^{\infty} w_t h_t(x)\right), \tag{2}$$

such that (w,h) is an optimal solution to

$$
\begin{aligned}
(P_3) \quad \min_{w_t\in\mathbb{R},h_t\in\mathcal{H}} \quad & \sum_{t=1}^{\infty} w_t \\
\text{s.t.} \quad & y_i\left(\sum_{t=1}^{\infty} w_t h_t(x_i)\right) \geq 1, && \text{for } i=1,2,\ldots,N, \\
& w_t \geq 0, && \text{for } t=1,2,\ldots,\infty.
\end{aligned}
$$

Note that there are infinitely many variables in (P_3). In order to approximate the optimal solution well with a fixed and finite T, AdaBoost resorts to two related properties of some of the optimal solutions for (P_3): finiteness and sparsity.

- **Finiteness:** When two hypotheses have the same prediction patterns on the training input vectors, they can be used interchangeably during the training time, and are thus *ambiguous*. Since there are at most 2^N prediction patterns on N training input vectors, we can partition $\mathcal{H}$ into at most 2^N groups, each of which contains mutually ambiguous hypotheses. Some optimal solutions of (P_3) only assign one or a few nonzero weights within each group (Demiriz et al., 2002). Thus, it is possible to work on a finite data-dependent subset of $\mathcal{H}$ instead of $\mathcal{H}$ itself without losing optimality.
- **Sparsity:** Minimizing the ℓ_1-norm $\|w\|_1 = \sum_{t=1}^{\infty} |w_t|$ often leads to sparse solutions (Meir and Rätsch, 2003; Rosset et al., 2007). That is, for hypotheses in the finite (but possibly still large) subset of $\mathcal{H}$, only a small number of weights needs to be nonzero. AdaBoost can be viewed as a stepwise greedy search algorithm that approximates such a finite and sparse ensemble (Rosset et al., 2004).

Another boosting approach, called the linear programming boosting (LPBoost), can solve (P_3) exactly. We will introduce the soft-margin LPBoost, which constructs an ensemble classifier like (2) with the optimal solution to

$$
\begin{aligned}
(P_4) \quad \min_{w_t\in\mathbb{R},h_t\in\mathcal{H}} \quad & \sum_{t=1}^{\infty} w_t + C\sum_{i=1}^{N} \xi_i \\
\text{s.t.} \quad & y_i\left(\sum_{t=1}^{\infty} w_t h_t(x_i)\right) \geq 1-\xi_i, && \text{for } i=1,2,\ldots,N, \\
& \xi_i \geq 0, && \text{for } i=1,2,\ldots,N, \\
& w_t \geq 0, && \text{for } t=1,2,\ldots,\infty.
\end{aligned}
$$

Demiriz et al. (2002) proposed to solve (P_4) with the column generating technique.[2] The algorithm works by adding one unambiguous h_t to the ensemble in each iteration. Because of the finiteness property, the algorithm is guaranteed to terminate within $T \leq 2^N$ iterations. The sparsity property can sometimes help speed up the convergence of the algorithm.

Rätsch et al. (2002) worked on a variant of (P_4) for regression problems, and discussed optimality conditions when $\mathcal{H}$ is of infinite size. Their results can be applied to (P_4) as well. In particular,

2. Demiriz et al. (2002) actually worked on an equivalent but slightly different formulation.

they showed that even without the finiteness property (e.g., when h_t outputs real values rather than binary values), (P_4) can still be solved using a finite subset of $\mathcal{H}$ that is associated with nonzero weights. The results justify the use of the column generating technique above, as well as a barrier, AdaBoost-like, approach that they proposed.

Recently, Rosset et al. (2007) studied the existence of a sparse solution when solving a generalized form of (P_4) with some $\mathcal{H}$ of infinite and possibly uncountable size. They showed that under some assumptions, there exists an optimal solution of (P_4) such that at most $N+1$ weights are nonzero. Thus, iterative algorithms that keep adding necessary hypotheses h_t to the ensemble, such as the proposed path-following approach (Rosset et al., 2007) or the column generating technique (Demiriz et al., 2002; Rätsch et al., 2002), could work by aiming towards such a sparse solution.

Note that even though the findings above indicate that it is possible to design good algorithms to return an optimal solution when $\mathcal{H}$ is infinitely large, the resulting ensemble relies on the sparsity property, and is effectively of only finite size. Nevertheless, it is not clear whether the performance could be improved if either or both the finiteness and the sparsity restrictions are removed.

2.3 Connecting Support Vector Machine to Ensemble Learning

The connection between AdaBoost, LPBoost, and SVM is well-known in literature (Freund and Schapire, 1999; Rätsch et al., 2001; Rätsch et al., 2002; Demiriz et al., 2002). Consider the feature transform

$$\Phi(x) = \big(h_1(x), h_2(x), \ldots\big). \tag{3}$$

We can see that the problem (P_1) with this feature transform is similar to (P_4). The elements of ϕ_x in SVM are similar to the hypotheses $h_t(x)$ in AdaBoost and LPBoost. They all work on linear combinations of these elements, though SVM deals with an additional intercept term b. SVM minimizes the ℓ_2-norm of the weights while AdaBoost and LPBoost work on the ℓ_1-norm. SVM and LPBoost introduce slack variables ξ_i and use the parameter C for regularization, while AdaBoost relies on the choice of the parameter T (Rosset et al., 2004). Note that AdaBoost and LPBoost require $w_t \geq 0$ for ensemble learning.

Several researchers developed interesting results based on the connection. For example, Rätsch et al. (2001) proposed to select the hypotheses h_t by AdaBoost and to obtain the weights w_t by solving an optimization problem similar to (P_1) in order to improve the robustness of AdaBoost. Another work by Rätsch et al. (2002) introduced a new density estimation algorithm based on the connection. Rosset et al. (2004) applied the similarity to compare SVM with boosting algorithms. Nevertheless, as limited as AdaBoost and LPBoost, their results could use only a finite subset of $\mathcal{H}$ when constructing the feature mapping (3). One reason is that the infinite number of variables w_t and constraints $w_t \geq 0$ are difficult to handle. We will show the remedies for these difficulties in the next section.

3. SVM-Based Framework for Infinite Ensemble Learning

Vapnik (1998) proposed a challenging task of designing an algorithm that actually generates an infinite ensemble classifier, that is, an ensemble classifier with infinitely many nonzero w_t. Traditional algorithms like AdaBoost or LPBoost cannot be directly generalized to solve the task, because they select the hypotheses in an iterative manner, and only run for a finite number of iterations.

We solved the challenge via another route: the connection between SVM and ensemble learning. The connection allows us to formulate a kernel that embodies all the hypotheses in $\mathcal{H}$. Then, the classifier (1) obtained from SVM with the kernel is a linear combination over $\mathcal{H}$ (with an intercept term). Nevertheless, there are still two main obstacles. One is to actually derive the kernel, and the other is to handle the constraints $w_t \geq 0$ to make (1) an ensemble classifier. In this section, we combine several ideas to deal with these obstacles, and conquer Vapnik's task with a novel SVM-based framework for infinite ensemble learning.

3.1 Embedding Hypotheses into the Kernel

We start by embedding the infinite number of hypotheses in $\mathcal{H}$ into an SVM kernel. We have shown in (3) that we could construct a feature mapping from $\mathcal{H}$. The idea is extended to a more general form for deriving a kernel in Definition 1.

Definition 1 *Assume that $\mathcal{H} = \{h_\alpha : \alpha \in \mathcal{C}\}$, where $\mathcal{C}$ is a measure space. The kernel that embodies $\mathcal{H}$ is defined as*

$$\mathcal{K}_{\mathcal{H},r}(x,x') = \int_{\mathcal{C}} \phi_x(\alpha)\phi_{x'}(\alpha)\, d\alpha, \tag{4}$$

where $\phi_x(\alpha) = r(\alpha)h_\alpha(x)$, and $r : \mathcal{C} \to \mathbb{R}^+$ is chosen such that the integral exists for all $x, x' \in \mathcal{X}$.

Here α is the parameter of the hypothesis h_α. Although two hypotheses with different α values may have the same input-output relation, we would treat them as different objects in our framework. We shall denote $\mathcal{K}_{\mathcal{H},r}$ by $\mathcal{K}_{\mathcal{H}}$ when r is clear from the context. The validity of the definition is formalized in the following theorem.

Theorem 2 *Consider the kernel $\mathcal{K}_{\mathcal{H}}$ in Definition 1.*

1. *The kernel is an inner product for ϕ_x and $\phi_{x'}$ in the Hilbert space $\mathcal{F} = \mathcal{L}_2(\mathcal{C})$, which contains functions $\varphi(\cdot) : \mathcal{C} \to \mathbb{R}$ that are square integrable.*

2. *For a set of input vectors $\{x_i\}_{i=1}^N \in \mathcal{X}^N$, the Gram matrix of $\mathcal{K}_{\mathcal{H}}$ is PSD.*

Proof The first part is known in mathematical analysis (Reed and Simon, 1980), and the second part follows Mercer's condition. ■

Constructing kernels from an integral inner product is a known technique in literature (Schölkopf and Smola, 2002). The framework adopts this technique for embedding the hypotheses, and thus could handle the situation even when $\mathcal{H}$ is uncountable. Note that when $r^2(\alpha)\, d\alpha$ is a "prior" on h_α, the kernel $\mathcal{K}_{\mathcal{H},r}(x,x')$ can be interpreted as a covariance function commonly used in Gaussian process (GP) models (Williams, 1998; Rasmussen and Williams, 2006). Some Bayesian explanations can then be derived from the connection between SVM and GP, but are beyond the scope of this paper.

3.2 Negation Completeness and Constant Hypotheses

When we use $\mathcal{K}_{\mathcal{H}}$ in (P_2), the primal problem (P_1) becomes

$$(P_5) \quad \min_{w\in\mathcal{L}_2(C),b\in\mathbb{R},\xi\in\mathbb{R}^N} \quad \frac{1}{2}\int_C w^2(\alpha)\,d\alpha + C\sum_{i=1}^{N}\xi_i$$
$$\text{s.t.} \quad y_i\left(\int_C w(\alpha)r(\alpha)h_\alpha(x_i)\,d\alpha + b\right) \geq 1-\xi_i, \qquad \text{for } i=1,2,\ldots,N,$$
$$\xi_i \geq 0, \qquad \text{for } i=1,2,\ldots,N.$$

In particular, the classifier obtained after solving (P_2) with $\mathcal{K}_{\mathcal{H}}$ is the same as the classifier obtained after solving (P_5):

$$g(x) = \text{sign}\left(\int_C w(\alpha)r(\alpha)h_\alpha(x)\,d\alpha + b\right). \tag{5}$$

When C is uncountable, it is possible that each hypothesis h_α only takes an infinitesimal weight $\big(w(\alpha)r(\alpha)\,d\alpha\big)$ in the ensemble. Thus, the classifier (5) is very different from those obtained with traditional ensemble learning, and will be discussed further in Subsection 4.2.

Note that the classifier (5) is not an ensemble classifier yet, because we do not have the constraints $w(\alpha)\geq 0$, and we have an additional term b. Next, we would explain that such a classifier is equivalent to an ensemble classifier under some reasonable assumptions.

We start from the constraints $w(\alpha)\geq 0$, which cannot be directly considered in (P_1). Vapnik (1998) showed that even if we add a countably infinite number of constraints to (P_1), infinitely many variables and constraints would be introduced to (P_2). Then, the latter problem would still be difficult to solve.

One remedy is to assume that $\mathcal{H}$ is negation complete, that is,[3]

$$h\in\mathcal{H} \Leftrightarrow (-h)\in\mathcal{H}.$$

Then, every linear combination over $\mathcal{H}$ has an equivalent linear combination with only nonnegative weights. Negation completeness is usually a mild assumption for a reasonable $\mathcal{H}$ (Rätsch et al., 2002). Following this assumption, the classifier (5) can be interpreted as an ensemble classifier over $\mathcal{H}$ with an intercept term b. Somehow b can be viewed as the weight on a constant hypothesis c, which always predicts $c(x)=1$ for all $x\in\mathcal{X}$. We shall further add a mild assumption that $\mathcal{H}$ contains both c and $(-c)$. Then, the classifier (5) or (1) is indeed equivalent to an ensemble classifier.

We summarize our framework in Algorithm 1. The framework shall generally inherit the profound performance of SVM. Most of the steps in the framework can be done by existing SVM implementations, and the hard part is mostly in obtaining the kernel $\mathcal{K}_{\mathcal{H}}$. In the next sections, we derive some concrete instances using different base hypothesis sets.

4. Stump Kernel

In this section, we present the stump kernel, which embodies infinitely many decision stumps. The decision stump $s_{q,d,\alpha}(x) = q\cdot\text{sign}\big((x)_d-\alpha\big)$ works on the d-th element of x, and classifies x according to $q\in\{-1,+1\}$ and the threshold α (Holte, 1993). It is widely used for ensemble learning because of its simplicity (Freund and Schapire, 1996).

3. We use $(-h)$ to denote the function $(-h)(\cdot) = -(h(\cdot))$.

1. Consider a training set $\{(x_i,y_i)\}_{i=1}^N$ and the hypothesis set $\mathcal{H}$, which is assumed to be negation complete and to contain a constant hypothesis.
2. Construct a kernel $\mathcal{K}_{\mathcal{H}}$ according to Definition 1 with a proper embedding function r.
3. Choose proper parameters, such as the soft-margin parameter C.
4. Solve (P_2) with $\mathcal{K}_{\mathcal{H}}$ and obtain Lagrange multipliers λ_i and the intercept term b.
5. Output the classifier
$$g(x) = \operatorname{sign}\left(\sum_{i=1}^N y_i\lambda_i \mathcal{K}_{\mathcal{H}}(x_i,x) + b\right),$$
which is equivalent to some ensemble classifier over $\mathcal{H}$.

Algorithm 1: SVM-based framework for infinite ensemble learning

4.1 Formulation and Properties

To construct the stump kernel, we consider the following set of decision stumps

$$\mathcal{S} = \left\{s_{q,d,\alpha_d} : q \in \{-1,+1\}, d \in \{1,\ldots,D\}, \alpha_d \in [L_d,R_d]\right\}.$$

We also assume $\mathcal{X} \subseteq (L_1,R_1)\times(L_2,R_2)\times\cdots\times(L_D,R_D)$. Thus, the set $\mathcal{S}$ is negation complete and contains $s_{+1,1,L_1}$ as a constant hypothesis. The stump kernel $\mathcal{K}_{\mathcal{S}}$ defined below can then be used in Algorithm 1 to obtain an infinite ensemble of decision stumps.

Definition 3 *The stump kernel is $\mathcal{K}_{\mathcal{S}}$ with $r(q,d,\alpha_d) = r_{\mathcal{S}} = \frac{1}{2}$,*

$$\mathcal{K}_{\mathcal{S}}(x,x') = \Delta_{\mathcal{S}} - \sum_{d=1}^D \left|(x)_d - (x')_d\right| = \Delta_{\mathcal{S}} - \left\|x-x'\right\|_1,$$

where $\Delta_{\mathcal{S}} = \frac{1}{2}\sum_{d=1}^D (R_d - L_d)$ is a constant.

Definition 3 is a concrete instance that follows Definition 1. The details of the derivation are shown in Appendix A. As we shall see further in Section 5, scaling $r_{\mathcal{S}}$ is equivalent to scaling the parameter C in SVM. Thus, without loss of generality, we use $r_{\mathcal{S}} = \frac{1}{2}$ to obtain a cosmetically cleaner kernel function.

The validity of the stump kernel follows directly from Theorem 2 of the general framework. That is, the stump kernel is an inner product in a Hilbert space of some square integrable functions $\varphi(q,d,\alpha_d)$, and it produces a PSD Gram matrix for any set of input vectors $\{x_i\}_{i=1}^N \in \mathcal{X}^N$. Given the ranges (L_d,R_d), the stump kernel is very simple to compute. Furthermore, the ranges are not even necessary in general, because dropping the constant $\Delta_{\mathcal{S}}$ does not affect the classifier obtained from SVM.

Theorem 4 *Solving (P_2) with the stump kernel $\mathcal{K}_{\mathcal{S}}$ is the same as solving (P_2) with the simplified stump kernel $\tilde{\mathcal{K}}_{\mathcal{S}}(x,x') = -\left\|x-x'\right\|_1$. That is, equivalent classifiers can be obtained from* (1)*.*

Proof We extend from the results of Berg et al. (1984) to show that $\tilde{\mathcal{K}}_{\mathcal{S}}(x,x')$ is conditionally PSD (CPSD). In addition, because of the constraint $\sum_{i=1}^{N} y_i\lambda_i = 0$, a CPSD kernel $\tilde{\mathcal{K}}(x,x')$ works exactly the same for (P_2) as any PSD kernel of the form $\tilde{\mathcal{K}}(x,x')+\Delta$, where Δ is a constant (Schölkopf and Smola, 2002). The proof follows with $\Delta = \Delta_{\mathcal{S}}$. ■

In fact, a kernel $\hat{\mathcal{K}}(x,x') = \tilde{\mathcal{K}}(x,x') + f(x) + f(x')$ with any mapping f is equivalent to $\tilde{\mathcal{K}}(x,x')$ for (P_2) because of the constraint $\sum_{i=1}^{N} y_i\lambda_i = 0$. Now consider another kernel

$$\hat{\mathcal{K}}_{\mathcal{S}}(x,x') = \tilde{\mathcal{K}}_{\mathcal{S}}(x,x') + \sum_{d=1}^{D}(x)_d + \sum_{d=1}^{D}(x')_d = 2\sum_{d=1}^{D}\min((x)_d,(x')_d).$$

We see that $\hat{\mathcal{K}}_{\mathcal{S}}$, $\tilde{\mathcal{K}}_{\mathcal{S}}$, and $\mathcal{K}_{\mathcal{S}}$ are equivalent for (P_2). The former is called the *histogram intersection kernel* (up to a scale of 2) when the elements $(x)_d$ represent generalized histogram counts, and has been successfully used in image recognition applications (Barla et al., 2003; Boughorbel et al., 2005; Grauman and Darrell, 2005). The equivalence demonstrates the usefulness of the stump kernel on histogram-based features, which would be further discussed in Subsection 6.4. A remark here is that our proof for the PSD-ness of $\mathcal{K}_{\mathcal{S}}$ comes directly from the framework, and hence is simpler and more straightforward than the proof of Boughorbel et al. (2005) for the PSD-ness of $\hat{\mathcal{K}}_{\mathcal{S}}$.

The simplified stump kernel is simple to compute, yet useful in the sense of dichotomizing the training set, which comes from the following positive definite (PD) property.

Theorem 5 *(Lin, 2005) Consider training input vectors $\{x_i\}_{i=1}^{N} \in \mathcal{X}^N$. If there exists a dimension d such that $(x_i)_d \neq (x_j)_d$ for all $i \neq j$, the Gram matrix of $\mathcal{K}_{\mathcal{S}}$ is PD.*

The PD-ness of the Gram matrix is directly connected to the classification capacity of the SVM classifiers. Chang and Lin (2001b) showed that when the Gram matrix of the kernel is PD, a hard-margin SVM with such a kernel can always dichotomize the training set perfectly. Keerthi and Lin (2003) then applied the result to show that SVM with the popular Gaussian-RBF kernel $\mathcal{K}(x,x') = \exp\left(-\gamma\|x-x'\|_2^2\right)$ can always dichotomize the training set when $C \to \infty$. We obtain a similar theorem for the stump kernel.

Theorem 6 *Under the assumption of Theorem 5, there exists some $C^* > 0$ such that for all $C \geq C^*$, SVM with $\mathcal{K}_{\mathcal{S}}$ can always dichotomize the training set $\{(x_i,y_i)\}_{i=1}^{N}$.*

We make two remarks here. First, although the assumption of Theorem 6 is mild in practice, there are still some data sets that do not have this property. An example is the famous XOR data set (Figure 1). We can see that every possible decision stump makes 50% of errors on the training input vectors. Thus, AdaBoost and LPBoost would terminate with one bad decision stump in the ensemble. Similarly, SVM with the stump kernel cannot dichotomize this training set perfectly, regardless of the choice of C. Such a problem is inherent in any ensemble model that combines decision stumps, because the model belongs to the family of generalized additive models (Hastie and Tibshirani, 1990; Hastie et al., 2001), and hence cannot approximate non-additive target functions well.

Second, although Theorem 6 indicates how the stump kernel can be used to dichotomize the training set perfectly, the classifier obtained usually overfits to noise (Keerthi and Lin, 2003). For

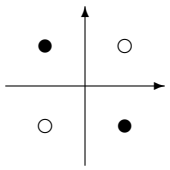


Figure 1: The XOR data set

the Gaussian-RBF kernel, it has been known that SVM with reasonable parameter selection provides suitable regularization and achieves good generalization performance even in the presence of noise (Keerthi and Lin, 2003; Hsu et al., 2003). We observe similar experimental results for the stump kernel (see Section 7).

4.2 Averaging Ambiguous Stumps

We have discussed in Subsection 2.2 that the set of hypotheses can be partitioned into groups and traditional ensemble learning algorithms can only pick a few representatives within each group. Our framework acts in a different way: the ℓ_2-norm objective function of SVM leads to an optimal solution that combines all the predictions within each group. This property is formalized in the following theorem.

Theorem 7 *Consider two ambiguous $h_\alpha, h_\beta \in \mathcal{H}$. If the kernel $\mathcal{K}_\mathcal{H}$ is used in Algorithm 1, the optimal w of (P_5) satisfies $\frac{w(\alpha)}{r(\alpha)} = \frac{w(\beta)}{r(\beta)}$.*

Proof The optimality condition between (P_1) and (P_2) leads to

$$\frac{w(\alpha)}{r(\alpha)} = \sum_{i=1}^{N} \lambda_i h_\alpha(x_i) = \sum_{i=1}^{N} \lambda_i h_\beta(x_i) = \frac{w(\beta)}{r(\beta)}. \qquad \blacksquare$$

If $w(\alpha)$ is nonzero, $w(\beta)$ would also be nonzero, which means both h_α and h_β are included in the ensemble. As a consequence, for each group of mutually ambiguous hypotheses, our framework considers the average prediction of all hypotheses as the consensus output.

The averaging process constructs a smooth representative for each group. In the following theorem, we demonstrate this behavior with the stump kernel, and show how the decision stumps group together in the final ensemble classifier.

Theorem 8 *Define $(\tilde{x})_{d,a}$ as the a-th smallest value in $\{(x_i)_d\}_{i=1}^N$, and A_d as the number of different $(\tilde{x})_{d,a}$. Let $(\tilde{x})_{d,0} = L_d$, $(\tilde{x})_{d,(A_d+1)} = R_d$, and*

$$\hat{s}_{q,d,a}(x) = q \cdot \begin{cases} +1, & \text{when } (x)_d \geq (\tilde{x})_{d,a+1}; \\ -1, & \text{when } (x)_d \leq (\tilde{x})_{d,a}; \\ \frac{2(x)_d - (\tilde{x})_{d,a} - (\tilde{x})_{d,a+1}}{(\tilde{x})_{d,a+1} - (\tilde{x})_{d,a}}, & \text{otherwise.} \end{cases}$$

Then, for $\hat{r}(q,d,a) = \frac{1}{2}\sqrt{(\tilde{x})_{d,a+1} - (\tilde{x})_{d,a}}$,

$$\mathcal{K}_\mathcal{S}(x_i, x) = \sum_{q \in \{-1,+1\}} \sum_{d=1}^{D} \sum_{a=0}^{A_d} \hat{r}^2(q,d,a) \hat{s}_{q,d,a}(x_i) \hat{s}_{q,d,a}(x).$$

Proof First, for any fixed q and d, a simple integration shows that

$$\int_{(\tilde{x})_{d,a}}^{(\tilde{x})_{d,a+1}} s_{q,d,\alpha}(x)\, d\alpha \quad = \quad \Big((\tilde{x})_{d,a+1} - (\tilde{x})_{d,a}\Big)\hat{s}_{q,d,a}(x).$$

In addition, note that for all $\alpha \in \Big((\tilde{x})_{d,a}, (\tilde{x})_{d,a+1}\Big)$, $\hat{s}_{q,d,a}(x_i) = s_{q,d,\alpha}(x_i)$. Thus,

$$\begin{aligned}
& \int_{L_d}^{R_d} \Big(r(q,d,\alpha) s_{q,d,\alpha}(x_i)\Big)\Big(r(q,d,\alpha) s_{q,d,\alpha}(x)\Big)\, d\alpha \\
= & \sum_{a=0}^{A_d} \int_{(\tilde{x})_{d,a}}^{(\tilde{x})_{d,a+1}} \left(\frac{1}{2} s_{q,d,\alpha}(x_i)\right)\left(\frac{1}{2} s_{q,d,\alpha}(x)\right) d\alpha \\
= & \sum_{a=0}^{A_d} \frac{1}{4} \hat{s}_{q,d,a}(x_i) \int_{(\tilde{x})_{d,a}}^{(\tilde{x})_{d,a+1}} s_{q,d,\alpha}(x)\, d\alpha \\
= & \sum_{a=0}^{A_d} \frac{1}{4} \Big((\tilde{x})_{d,a+1} - (\tilde{x})_{d,a}\Big) \hat{s}_{q,d,a}(x_i) \hat{s}_{q,d,a}(x).
\end{aligned}$$

The theorem can be proved by summing over all q and d. ■

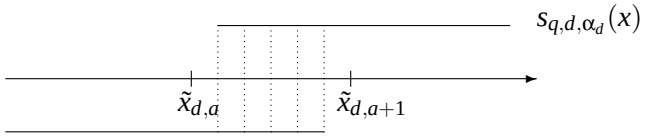
As shown in Figure 2, the function $\hat{s}_{q,d,a}$ is a smoother variant of the decision stump. Theorem 8 indicates that the infinite ensemble of decision stumps produced by our framework is equivalent to a finite ensemble of data-dependent and smoother variants. Another view of $\hat{s}_{q,d,a}$ is that they are continuous piecewise linear functions (order-2 splines) with knots defined on the training features (Hastie et al., 2001). Then, Theorem 8 indicates that an infinite ensemble of decision stumps can be obtained by fitting an additive model of finite size using these special splines as the bases. Note that although the fitting problem is of finite size, the number of possible splines can grow as large as $O(ND)$, which can sometimes be too large for iterative algorithms such as backfitting (Hastie et al., 2001). On the other hand, our SVM-based framework with the stump kernel can be thought as a route to solve this special spline fitting problem efficiently via the kernel trick.

As shown in the proof of Theorem 8, the averaged stump $\hat{s}_{q,d,a}$ represents the group of ambiguous decision stumps with $\alpha_d \in \Big((\tilde{x})_{d,a}, (\tilde{x})_{d,a+1}\Big)$. When the group is larger, $\hat{s}_{q,d,a}$ becomes smoother. Traditional ensemble learning algorithms like AdaBoost or LPBoost rely on a base learner to choose one decision stump as the only representative within each group, and the base learner usually returns the middle stump $m_{q,d,a}$. As shown in Figure 2, the threshold of the middle stump is at the mean of $(\tilde{x})_{d,a}$ and $(\tilde{x})_{d,a+1}$. Our framework, on the other hand, enjoys a smoother decision by averaging over more decision stumps. Even though each decision stump only has an infinitesimal hypothesis weight, the averaged stump $\hat{s}_{q,d,a}$ has a concrete weight in the ensemble.

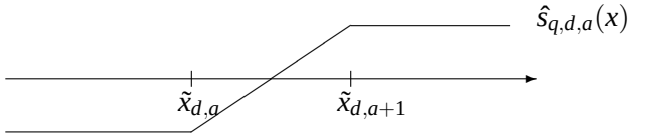
5. Perceptron Kernel

In this section, we extend the stump kernel to the perceptron kernel, which embodies infinitely many perceptrons. A perceptron is a linear threshold classifier of the form

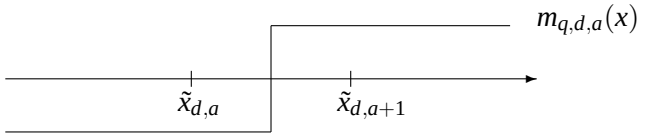
$$p_{\theta,\alpha}(x) = \mathrm{sign}\left(\theta^T x - \alpha\right).$$



(a) a group of ambiguous decision stumps s_{q,d,α_d} with $\alpha_d \in \left((\tilde{x})_{d,a}, (\tilde{x})_{d,a+1}\right)$



(b) SVM-based infinite ensemble learning uses the consensus: the averaged stump $\hat{s}_{q,d,a}$



(c) Base learners for AdaBoost and LPBoost usually only consider the middle stump $m_{q,d,a}$

Figure 2: The averaged stump and the middle stump

It is a basic theoretical model for a neuron, and is very important for building neural networks (Haykin, 1999).

To construct the perceptron kernel, we consider the following set of perceptrons

$$\mathcal{P} = \left\{ p_{\theta,\alpha} \colon \theta \in \mathbb{R}^D, \|\theta\|_2 = 1, \alpha \in [-R, R] \right\}.$$

We assume that $\mathcal{X}$ is within the interior of $\mathcal{B}(R)$, where $\mathcal{B}(R)$ is a ball of radius R centered at the origin in $\mathbb{R}^D$. Then, the set $\mathcal{P}$ is negation complete, and contains a constant hypothesis $p_{e_1,-R}$ where $e_1 = (1,0,\ldots,0)^T$. Thus, the perceptron kernel $\mathcal{K}_{\mathcal{P}}$ defined below can be used in Algorithm 1 to obtain an infinite ensemble of perceptrons.

Definition 9 *Let*

$$\Theta_D = \int_{\|\theta\|_2=1} d\theta, \quad \Xi_D = \int_{\|\theta\|_2=1} \left|\cos\left(\text{angle}\langle\theta, e_1\rangle\right)\right| d\theta,$$

where the operator angle$\langle\cdot,\cdot\rangle$ *is the angle between two vectors, and the integrals are calculated with uniform measure on the surface* $\|\theta\|_2 = 1$. *The perceptron kernel is* $\mathcal{K}_{\mathcal{P}}$ *with* $r(\theta,\alpha) = r_{\mathcal{P}}$,

$$\mathcal{K}_{\mathcal{P}}(x,x') = \Delta_{\mathcal{P}} - \left\|x - x'\right\|_2,$$

where the constants $r_{\mathcal{P}} = (2\Xi_D)^{-\frac{1}{2}}$ *and* $\Delta_{\mathcal{P}} = \Theta_D \Xi_D^{-1} R$.

The details are shown in Appendix A. With the perceptron kernel, we can construct an infinite ensemble of perceptrons. Such an ensemble is equivalent to a neural network with one hidden layer, infinitely many hidden neurons, and the hard-threshold activation functions. Williams (1998) built an infinite neural network with either the sigmoidal or the Gaussian activation function through computing the corresponding covariance function for GP models. Analogously, our approach returns an infinite neural network with hard-threshold activation functions (ensemble of perceptrons) through

computing the perceptron kernel for SVM. Williams (1998) mentioned that "*Paradoxically, it may be easier to carry out Bayesian prediction with infinite networks rather than finite ones.*" Similar claims can be made with ensemble learning.

The perceptron kernel shares many similar properties to the stump kernel. First, the constant $\Delta_{\mathcal{P}}$ can also be dropped, as formalized below.

Theorem 10 *Solving (P_2) with the simplified perceptron kernel $\tilde{\mathcal{K}}_{\mathcal{P}}(x,x') = -\|x-x'\|_2$ is the same as solving (P_2) with $\mathcal{K}_{\mathcal{P}}(x,x')$.*

Second, SVM with the perceptron kernel can also dichotomize the training set perfectly, which comes from the usefulness of the simplified perceptron kernel $\tilde{\mathcal{K}}_{\mathcal{P}}$ in interpolation.

Theorem 11 *(Micchelli, 1986) Consider input vectors $\{x_i\}_{i=1}^N \in \mathcal{X}^N$, and the perceptron kernel $\mathcal{K}_{\mathcal{P}}$ in Definition 9. If $x_i \neq x_j$ for all $i \neq j$, then the Gram matrix of $\mathcal{K}_{\mathcal{P}}$ is PD.*

Then, similar to Theorem 6, we get the following result.

Theorem 12 *If $x_i \neq x_j$ for all $i \neq j$, there exists some $C^* > 0$ such that for all $C \geq C^*$, SVM with $\mathcal{K}_{\mathcal{P}}$ can always dichotomize the training set $\{(x_i,y_i)\}_{i=1}^N$.*

Another important property, called *scale-invariance*, accompanies the simplified perceptron kernel, which was also named the *triangular kernel* by Fleuret and Sahbi (2003). They proved that when the kernel is used in the hard-margin SVM, scaling all training input vectors x_i by some positive γ does not change the optimal solution.

In fact, in the soft-margin SVM, a well-known result is that scaling the Gram matrix K by some $\gamma > 0$ is equivalent to scaling C by γ in (P_2). Because the simplified perceptron kernel $\tilde{\mathcal{K}}_{\mathcal{P}}$ satisfies $\gamma\tilde{\mathcal{K}}_{\mathcal{P}}(x,x') = \tilde{\mathcal{K}}_{\mathcal{P}}(\gamma x,\gamma x')$, the effect of scaling training examples can be equivalently performed with the parameter selection step on C. That is, when C is selected reasonably, there is no need to explicitly have a scaling parameter γ.

Recall that we construct the perceptron kernel (and the stump kernel) with an embedding constant $r_{\mathcal{P}}$ (and $r_{\mathcal{S}}$), and from Definition 1, multiplying the constant by $\sqrt{\gamma} > 0$ is equivalent to scaling the Gram matrix K by γ. Thus, when C is selected reasonably, there is also no need to explicitly try different $r_{\mathcal{P}}$ or $r_{\mathcal{S}}$ for these two kernels. We will further discuss the benefits of this property in Subsection 6.4.

6. Laplacian-RBF Kernel

In the previous sections, we applied Definition 1 on some simple base hypothesis sets. Next, we show how complex hypothesis sets can also be embedded in a kernel by suitably combining the kernels that embody simpler sets. We will introduce two useful tools: summation and multiplication. The tools would eventually allow us to embed infinitely many decision trees in a kernel. Interestingly, the kernel obtained is equivalent to the well-known Laplacian-RBF kernel in some parameters.

6.1 Summation: Embedding Multiple Sets of Hypotheses

Summation can be used to embed multiple sets of hypotheses altogether. For example, given kernels $\mathcal{K}_{\mathcal{H}_1}$ and $\mathcal{K}_{\mathcal{H}_2}$, their summation

$$\mathcal{K}(x,x') = \mathcal{K}_{\mathcal{H}_1}(x,x') + \mathcal{K}_{\mathcal{H}_2}(x,x')$$

embodies both $\mathcal{H}_1$ and $\mathcal{H}_2$. In other words, if we use $\mathcal{K}(x,x')$ in Algorithm 1, we could obtain an ensemble classifier over $\mathcal{H}_1 \cup \mathcal{H}_2$ when the union is negation complete and contains a constant hypothesis.

In traditional ensemble learning, when multiple sets of hypotheses are considered altogether, it is usually necessary to call a base learner for each set. On the other hand, our framework only requires a simple summation on the kernel evaluations. In fact, as shown in the next theorem, our framework can be applied to work with any countable sets of hypotheses, which may not be an easy task for traditional ensemble learning algorithms.

Theorem 13 *Assume that the kernels $\mathcal{K}_{\mathcal{H}_1}, \ldots, \mathcal{K}_{\mathcal{H}_J}$ are defined for some $J \in \mathbb{N} \bigcup \{\infty\}$ with sets of hypotheses $\mathcal{H}_1, \ldots, \mathcal{H}_J$, respectively. Then, let*

$$\mathcal{K}(x,x') = \sum_{j=1}^{J} \mathcal{K}_{\mathcal{H}_j}(x,x').$$

If $\mathcal{K}(x,x')$ exists for all $x,x' \in \mathcal{X}$, and $\mathcal{H} = \bigcup_{j=1}^{J} \mathcal{H}_j$ is negation complete and contains a constant hypothesis, Algorithm 1 using $\mathcal{K}(x,x')$ outputs an ensemble classifier over $\mathcal{H}$.

Proof The theorem comes from the following result in mathematical analysis: any countable direct sum over Hilbert spaces is a Hilbert space (Reed and Simon, 1980, Example 5). Lin (2005, Theorem 6) showed the details of the proof. ■

A remark on Theorem 13 is that we do not intend to define a kernel with $\mathcal{H}$ directly. Otherwise we need to choose suitable $\mathcal{C}$ and r first, which may not be an easy task for such a complex hypothesis set. Using the summation of the kernels, on the other hand, allow us to obtain an ensemble classifier over the full union with less efforts.

6.2 Multiplication: Performing Logical Combination of Hypotheses

It is known that we can combine two kernels by point-wise multiplication to form a new kernel (Schölkopf and Smola, 2002). When the two kernels are associated with base hypothesis sets, a natural question is: what hypothesis set is embedded in the new kernel?

Next, let output $+1$ represent logic TRUE and -1 represent logic FALSE. We show that multiplication can be used to perform common logical combinations on the hypotheses.

Theorem 14 *For two sets of hypotheses $\mathcal{H}_1 = \{h_\alpha : \alpha \in \mathcal{C}_1\}$ and $\mathcal{H}_2 = \{h_\beta : \beta \in \mathcal{C}_2\}$, define*

$$\mathcal{H} = \left\{h_{\alpha,\beta} : h_{\alpha,\beta}(x) = -h_\alpha(x) \cdot h_\beta(x), \alpha \in \mathcal{C}_1, \beta \in \mathcal{C}_2\right\}.$$

In addition, let $r(\alpha,\beta) = r_1(\alpha) r_2(\beta)$. Then,

$$\mathcal{K}_{\mathcal{H},r}(x,x') = \mathcal{K}_{\mathcal{H}_1,r_1}(x,x') \cdot \mathcal{K}_{\mathcal{H}_2,r_2}(x,x')$$

for all $x,x' \in \mathcal{X}$.

The proof simply follows from Definition 1. Note that when representing logic, the combined hypothesis $h_{\alpha,\beta}$ is the XOR operation on h_α and h_β. More complicated results about other operations can be introduced under a mild assumption called *neutrality*.

Definition 15 *A set of hypothesis $\mathcal{H} = \{h_\alpha : \alpha \in \mathcal{C}\}$ is neutral to $\mathcal{X}$ with a given r if and only if for all $x \in \mathcal{X}$, $\int_{\alpha\in\mathcal{C}} h_\alpha(x) r^2(\alpha)\, d\alpha = 0$.*

Note that for a negation complete set $\mathcal{H}$, neutrality is usually a mild assumption (e.g., by assigning the same r for h_α and $-h_\alpha$). We can easily verify that the set of decision stumps in Definition 3 and the set of perceptrons in Definition 9 are both neutral.

Theorem 16 *For two sets of hypotheses $\mathcal{H}_1 = \{h_\alpha : \alpha \in \mathcal{C}_1\}$ and $\mathcal{H}_2 = \{h_\beta : \beta \in \mathcal{C}_2\}$, define*

$$\mathcal{H} \quad = \quad \left\{h_{q,\alpha,\beta} : h_{q,\alpha,\beta}(x) = q \cdot \min\left(h_\alpha(x), h_\beta(x)\right), \alpha \in \mathcal{C}_1, \beta \in \mathcal{C}_2, q \in \{-1,+1\}\right\}.$$

Assume that $\mathcal{H}_1$ and $\mathcal{H}_2$ are neutral with r_1 and r_2, respectively, and both integrals

$$\Delta_1 = \int_{\alpha\in\mathcal{C}_1} r_1^2(\alpha)\, d\alpha,\ \Delta_2 = \int_{\beta\in\mathcal{C}_2} r_2^2(\beta)\, d\beta$$

are finite. In addition, let $r(q,\alpha,\beta) = \sqrt{2} r_1(\alpha) r_2(\beta)$. Then,

$$\mathcal{K}_{\mathcal{H},r}(x,x') = \left(\mathcal{K}_{\mathcal{H}_1,r_1}(x,x') + \Delta_1\right) \cdot \left(\mathcal{K}_{\mathcal{H}_2,r_2}(x,x') + \Delta_2\right)$$

for all $x, x' \in \mathcal{X}$. Furthermore, $\mathcal{H}$ is neutral to $\mathcal{X}$ with r.

Proof Because $h_\alpha(x), h_\beta(x) \in \{-1,+1\}$,

$$h_{+1,\alpha,\beta}(x) = \frac{1}{2}\left(h_\alpha(x)h_\beta(x) + h_\alpha(x) + h_\beta(x) - 1\right).$$

Then,

$$\begin{aligned}
& \mathcal{K}_{\mathcal{H},r}(x,x') \\
= \ & 2\int h_{+1,\alpha,\beta}(x) h_{+1,\alpha,\beta}(x') r^2(\alpha,\beta)\, d\beta\, d\alpha \\
= \ & \frac{1}{2}\int \left(h_\alpha(x)h_\beta(x) + h_\alpha(x) + h_\beta(x) - 1\right)\left(h_\alpha(x')h_\beta(x') + h_\alpha(x') + h_\beta(x') - 1\right) r^2(\alpha,\beta)\, d\beta\, d\alpha \\
= \ & \int \left(h_\alpha(x)h_\beta(x)h_\alpha(x')h_\beta(x') + h_\alpha(x)h_\alpha(x') + h_\beta(x)h_\beta(x') + 1\right) r_1^2(\alpha) r_2^2(\beta)\, d\beta\, d\alpha \qquad (6) \\
= \ & \left(\mathcal{K}_{\mathcal{H}_1,r_1}(x,x') + \Delta_1\right) \cdot \left(\mathcal{K}_{\mathcal{H}_2,r_2}(x,x') + \Delta_2\right).
\end{aligned}$$

Note that (6) comes from the neutrality assumption, which implies that during integration, the cross-terms like

$$\int h_\alpha(x) h_\beta(x') r_1^2(\alpha) r_2^2(\beta)\, d\alpha\, d\beta$$

are all 0. Neutrality of $\mathcal{H}$ follows from the symmetry in q. ■

The arithmetic operation $(+1 \cdot \min)$ is equivalent to the AND operation when the outputs represent logic, and hence $(-1 \cdot \min)$ represents the NAND operation. If $\mathcal{H}_1$ and $\mathcal{H}_2$ are negation complete, the NOT operation is implicit in the original sets, and hence OR can be equivalently performed through $\mathrm{OR}(a,b) = \mathrm{NAND}(\mathrm{NOT}(a), \mathrm{NOT}(b))$.

6.3 Stump Region Kernel, Decision Tree Kernel, and Laplacian-RBF Kernel

Next, we use the stump kernel to demonstrate the usefulness of summation and multiplication. When $\mathcal{H}_1 = \mathcal{H}_2 = \mathcal{S}$, the resulting $\mathcal{K}_{\mathcal{H}}$ from Theorem 16 embodies AND/OR combinations of two decision stumps in $\mathcal{S}$. Extending this concept, we get the following new kernels.

Definition 17 *The L-level stump region kernel $\mathcal{K}_{\mathcal{I}_L}$ is recursively defined by*

$$\begin{aligned} \mathcal{K}_{\mathcal{I}_1}(x,x') &= \mathcal{K}_{\mathcal{S}}(x,x') + \Delta_{\mathcal{S}},\ \Delta_1 = 2\Delta_{\mathcal{S}}, \\ \mathcal{K}_{\mathcal{I}_{L+1}}(x,x') &= \left(\mathcal{K}_{\mathcal{I}_L}(x,x') + \Delta_L\right)\left(\mathcal{K}_{\mathcal{S}}(x,x') + \Delta_{\mathcal{S}}\right),\ \Delta_{L+1} = 2\Delta_L\Delta_{\mathcal{S}} \text{ for } L \in \mathbb{N}. \end{aligned}$$

If we construct a kernel from $\{c,-c\}$ with $r = \sqrt{\frac{1}{2}\Delta_{\mathcal{S}}}$ on each hypothesis, we can see that the constant $\Delta_{\mathcal{S}}$ is also a neutral kernel. Since neutrality is preserved by summation, the kernel $\mathcal{K}_{\mathcal{I}_1}$ is neutral as well. By repeatedly applying Theorem 16 and maintaining Δ_L as the constant associated with $\mathcal{I}_L$, we see that $\mathcal{K}_{\mathcal{I}_L}$ embodies all possible AND/OR combinations of L decision stumps in $\mathcal{S}$. We call these hypotheses the *L-level stump regions*.

Note that we can solve the recurrence and get

$$\mathcal{K}_{\mathcal{I}_L}(x,x') = 2^L\Delta_{\mathcal{S}}^L \sum_{\ell=1}^{L} \left(\frac{\mathcal{K}_{\mathcal{S}}(x,x') + \Delta_{\mathcal{S}}}{2\Delta_{\mathcal{S}}}\right)^{\ell}, \text{ for } L \in \mathbb{N}.$$

Then, by applying Theorem 13, we obtain an ensemble classifier over stump regions of any level.

Theorem 18 *For $0 < \gamma < \frac{1}{\Delta_{\mathcal{S}}}$, the infinite stump region (decision tree) kernel*

$$\mathcal{K}_{\mathcal{I}}(x,x') = \exp\left(\gamma \cdot \left(\mathcal{K}_{\mathcal{S}}(x,x') + \Delta_{\mathcal{S}}\right)\right) - 1$$

can be applied to Algorithm 1 to obtain an ensemble classifier over $\mathcal{I} = \bigcup_{L=1}^{\infty} \mathcal{I}_L$.

Proof By Taylor's series expansion of $\exp(\varepsilon)$ near $\varepsilon = 0$, we get

$$\begin{aligned} \mathcal{K}_{\mathcal{I}}(x,x') &= \sum_{L=1}^{\infty} \frac{\gamma^L}{L!}\left(\mathcal{K}_{\mathcal{S}}(x,x') + \Delta_{\mathcal{S}}\right)^L \\ &= \gamma\mathcal{K}_{\mathcal{I}_1}(x,x') + \sum_{L=2}^{\infty} \frac{\gamma^L}{L!}\left(\mathcal{K}_{\mathcal{I}_L}(x,x') - 2\Delta_{\mathcal{S}}\,\mathcal{K}_{\mathcal{I}_{L-1}}(x,x')\right) \\ &= \sum_{L=1}^{\infty} \frac{\gamma^L}{L!}\mathcal{K}_{\mathcal{I}_L}(x,x') - \sum_{L=1}^{\infty} \frac{\gamma^{L+1}}{(L+1)!} 2\Delta_{\mathcal{S}}\,\mathcal{K}_{\mathcal{I}_L}(x,x') \\ &= \sum_{L=1}^{\infty} \left(\frac{\gamma^L}{L!} - \frac{\gamma^{L+1}2\Delta_{\mathcal{S}}}{(L+1)!}\right)\mathcal{K}_{\mathcal{I}_L}(x,x'). \end{aligned}$$

Note that $\tau_L = \frac{\gamma^L}{L!} - \frac{\gamma^{L+1}2\Delta_{\mathcal{S}}}{(L+1)!} > 0$ for all $L \geq 1$ if and only if $0 < \gamma < \frac{1}{\Delta_{\mathcal{S}}}$. The desired result simply follows Theorem 13 by scaling the r functions of each $\mathcal{K}_{\mathcal{I}_L}$ by $\sqrt{\tau_L}$. ■

The set of stump regions of any level contains all AND/OR combinations of decision stumps. It is not hard to see that every stump region can be represented by recursive axis-parallel partitions

that output $\{-1,+1\}$, that is, a decision tree (Quinlan, 1986; Hastie et al., 2001). In addition, we can view the nodes of a decision tree as logic operations:

$$\begin{aligned} & \mathsf{tree} \\ = \; & \mathrm{OR}(\mathrm{AND}(\text{root node condition}, \mathsf{left}), \mathrm{AND}(\mathrm{NOT}(\text{root node condition}), \mathsf{right})). \end{aligned}$$

By recursively replacing each root node condition with a decision stump, we see that every decision tree can be represented as a stump region hypothesis. Thus, the set $\mathcal{T}$ that contains stump regions of any level is the same as the set of all possible decision trees, which leads to the name *decision tree kernel*.[4]

Decision trees are popular for ensemble learning, but traditional algorithms can only deal with trees of finite levels (Breiman, 1999; Dietterich, 2000). On the other hand, when the decision tree kernel $\mathcal{K}_{\mathcal{T}}$ is plugged into our framework, it allows us to actually build an infinite ensemble over decision trees of arbitrary levels.

Note that the decision tree kernel $\mathcal{K}_{\mathcal{T}}(x,x')$ is of the form

$$\kappa_1 \exp\left(-\kappa_2 \left\|x-x'\right\|_1\right) + \kappa_3$$

where $\kappa_1, \kappa_2, \kappa_3$ are constants and κ_1, κ_2 are positive. We mentioned in Section 4 that scaling the kernel with κ_1 is equivalent to scaling the soft-margin parameter C in SVM, and in Theorem 4 that dropping κ_3 does not affect the solution obtained from SVM. Then, the kernel $\mathcal{K}_{\mathcal{T}}(x,x')$ is similar to the Laplacian-RBF kernel $\mathcal{K}_{\mathcal{L}}(x,x') = \exp(-\gamma\|x-x'\|_1)$. This result is a novel interpretation of the Laplacian-RBF kernel: under suitable parameters, SVM with the Laplacian-RBF kernel allows us to obtain an infinite ensemble classifier over decision trees of any level.[5]

Not surprisingly, when all training input vectors x_i are distinct (Micchelli, 1986; Baxter, 1991), the Gram matrix of $\mathcal{K}_{\mathcal{L}}$ (and hence $\mathcal{K}_{\mathcal{T}}$) is PD. Then, the Laplacian-RBF kernel and the decision tree kernel could be used to dichotomize the training set perfectly.

6.4 Discussion on Radial Basis Function Kernels

Note that the stump kernel, the perceptron kernel, the Laplacian-RBF kernel, and the Gaussian-RBF kernel are all radial basis functions. They can all be used to dichotomize the training set perfectly under mild conditions, while the first three connect to explanations from an ensemble perspective. Next, we compare two properties of these kernels, and discuss their use in SVM applications.

First, we can group these kernels by the distance metrics they use. The stump kernel and the Laplacian-RBF kernel deal with the ℓ_1-norm distance between input vectors, while the others work on the ℓ_2-norm distance. An interesting property of using the ℓ_2-norm distance is the invariance to rotations. From the construction of the perceptron kernel, we can see how the rotation invariance is obtained from an ensemble point-of-view. The transformation vectors θ in perceptrons represent the rotation, and rotation invariance comes from embedding all possible θ uniformly in the kernel.

4. We use the name decision tree kernel for $\mathcal{K}_{\mathcal{T}}$ in Theorem 18 because the kernel embodies an infinite number of decision tree "hypotheses" and can be used in our framework to construct an infinite ensemble of decision trees. As pointed out by a reviewer, however, the kernel is derived in a particular way, which makes the metric of the underlying feature space different from the metrics associated with common decision tree "algorithms."

5. Note that the techniques in Theorem 18 can be coupled with Theorem 14 to show that Laplacian-RBF kernel with any $\gamma > 0$ embodies XOR stump regions (a special type of decision tree) of any level. We emphasize on the AND-OR stump regions here to connect better to general decision trees.

Some applications, however, may not desire rotation invariance. For example, when representing an image with color histograms, rotation could mix up the information in each color component. Chapelle et al. (1999) showed some successful results with the Laplacian-RBF kernel on this application. In Subsection 4.1, we have also discussed some image recognition applications using the histogram intersection kernel, which is equivalent to the stump kernel, on histogram-based features. Gene expression analysis, as demonstrated by Lin and Li (2005b), is another area that the stump kernel could be helpful.

Second, we can group kernels by whether they are scale-invariant (see also Section 5). The simplified stump kernel and the simplified perceptron kernel are scale-invariant, which means that C is the only parameter that needs to be determined. On the other hand, different combinations of (γ, C) need to be considered for the Gaussian-RBF kernel or the Laplacian-RBF kernel during parameter selection (Keerthi and Lin, 2003). Thus, SVM with the simplified stump kernel or the simplified perceptron kernel enjoys an advantage on speed during parameter selection. As we will see in Section 7.2, experimentally they perform similarly to the Gaussian-RBF kernel on many data sets. Thus, SVM applications that consider speed as an important factor may benefit from using the simplified stump kernel or the simplified perceptron kernel.

7. Experiments

We first compare our SVM-based infinite ensemble learning framework with AdaBoost and LPBoost using decision stumps, perceptrons, or decision trees as the base hypothesis set. The simplified stump kernel (SVM-Stump), the simplified perceptron kernel (SVM-Perc), and the Laplacian-RBF kernel (SVM-Dec) are plugged into Algorithm 1 respectively. We also compare SVM-Stump, SVM-Perc, and SVM-Dec with SVM-Gauss, which is SVM with the Gaussian-RBF kernel.

The deterministic decision stump algorithm (Holte, 1993), the random coordinate descent perceptron algorithm (Li and Lin, 2007), and the C4.5 decision tree algorithm (Quinlan, 1986) are taken as base learners in AdaBoost and LPBoost for the corresponding base hypothesis set. For perceptrons, we use the RCD-bias setting with 200 epochs of training; for decision trees, we take the pruned tree with the default settings of C4.5. All base learners above have been shown to work reasonably well with boosting in literature (Freund and Schapire, 1996; Li and Lin, 2007).

We discussed in Subsection 4.2 that a common implementation of AdaBoost-Stump and LPBoost-Stump only chooses the middle stumps. For further comparison, we include all the middle stumps in a set $\mathcal{M}$, and construct a kernel $\mathcal{K}_{\mathcal{M}}$ with $r = \frac{1}{2}$ according to Definition 1. Because $\mathcal{M}$ is a finite set, the integral in (4) becomes a summation when computed with the counting measure. We test our framework with this kernel, and call it SVM-Mid.

LIBSVM 2.8 (Chang and Lin, 2001a) is adopted as the soft-margin SVM solver, with a suggested procedure that selects a suitable parameter with a five-fold cross validation on the training set (Hsu et al., 2003). For SVM-Stump, SVM-Mid, and SVM-Perc, the parameter $\log_2 C$ is searched within $\{-17, -15, \ldots, 3\}$, and for SVM-Dec and SVM-Gauss, the parameters $(\log_2 \gamma, \log_2 C)$ are searched within $\{-15, -13, \ldots, 3\} \times \{-5, -3, \ldots, 15\}$. We use different search ranges for $\log_2 C$ because the numerical ranges of the kernels could be quite different. After the parameter selection procedure, a new model is trained using the whole training set, and the generalization ability is evaluated on an unseen test set.

For boosting algorithms, we conduct the parameter selection procedure similarly. The parameter $\log_2 C$ of LPBoost is also searched within $\{-17, -15, \ldots, 3\}$. For AdaBoost, the parameter T

data set	number of training examples	number of test examples	number of features
twonorm	300	3000	20
twonorm-n	300	3000	20
threenorm	300	3000	20
threenorm-n	300	3000	20
ringnnorm	300	3000	20
ringnorm-n	300	3000	20
australian	414	276	14
breast	409	274	10
german	600	400	24
heart	162	108	13
ionosphere	210	141	34
pima	460	308	8
sonar	124	84	60
votes84	261	174	16
a1a	1605	30956	123
splice	1000	2175	60
svmguide1	3089	4000	4
w1a	2477	47272	300

Table 1: Summarized information of the data sets used

is searched within $\{10, 20, \ldots, 1500\}$. Note that because LPBoost can be slow when the ensemble size is too large (Demiriz et al., 2002), we set a stopping criterion to generate at most 1000 columns (hypotheses) in order to obtain an ensemble within a reasonable amount of time.

The three artificial data sets from Breiman (1999) (twonorm, threenorm, and ringnorm) are generated with training set size 300 and test set size 3000. We create three more data sets (twonorm-n, threenorm-n, ringnorm-n), which contain mislabeling noise on 10% of the training examples, to test the performance of the algorithms on noisy data. We also use eight real-world data sets from the UCI repository (Hettich et al., 1998): australian, breast, german, heart, ionosphere, pima, sonar, and votes84. Their feature elements are scaled to $[-1, 1]$. We randomly pick 60% of the examples for training, and the rest for testing. For the data sets above, we compute the means and the standard errors of the results over 100 runs. In addition, four larger real-world data sets are used to test the validity of the framework for large-scale learning. They are a1a (Hettich et al., 1998; Platt, 1999), splice (Hettich et al., 1998), svmguide1 (Hsu et al., 2003), and w1a (Platt, 1999).[6] Each of them comes with a benchmark test set, on which we report the results. Some information of the data sets used is summarized in Table 1.

6. These data sets are downloadable on tools page of LIBSVM (Chang and Lin, 2001a).

data set	SVM-Stump	SVM-Mid	AdaBoost-Stump	LPBoost-Stump
twonorm	$\mathbf{2.86 \pm 0.04}$	3.10 ± 0.04	5.02 ± 0.06	5.58 ± 0.07
twonorm-n	$\mathbf{3.08 \pm 0.06}$	3.29 ± 0.05	12.7 ± 0.17	17.9 ± 0.19
threenorm	$\mathbf{17.7 \pm 0.10}$	18.6 ± 0.12	22.1 ± 0.12	24.1 ± 0.15
threenorm-n	$\mathbf{19.0 \pm 0.14}$	19.6 ± 0.13	26.1 ± 0.17	30.3 ± 0.16
ringnorm	$\mathbf{3.97 \pm 0.07}$	5.30 ± 0.07	10.1 ± 0.14	10.3 ± 0.14
ringnorm-n	$\mathbf{5.56 \pm 0.11}$	7.03 ± 0.14	19.6 ± 0.20	22.4 ± 0.21
australian	$\mathbf{14.4 \pm 0.21}$	15.9 ± 0.18	$\mathbf{14.2 \pm 0.18}$	19.8 ± 0.24
breast	3.11 ± 0.08	$\mathbf{2.77 \pm 0.08}$	4.41 ± 0.10	4.79 ± 0.12
german	$\mathbf{24.7 \pm 0.18}$	$\mathbf{24.9 \pm 0.17}$	25.4 ± 0.19	31.6 ± 0.20
heart	$\mathbf{16.4 \pm 0.27}$	19.1 ± 0.35	19.2 ± 0.35	24.4 ± 0.39
ionosphere	$\mathbf{8.13 \pm 0.17}$	$\mathbf{8.37 \pm 0.20}$	11.3 ± 0.25	11.5 ± 0.24
pima	$\mathbf{24.1 \pm 0.23}$	$\mathbf{24.4 \pm 0.23}$	24.8 ± 0.23	31.0 ± 0.24
sonar	$\mathbf{16.6 \pm 0.42}$	18.0 ± 0.37	19.4 ± 0.38	19.8 ± 0.37
votes84	4.76 ± 0.14	4.76 ± 0.14	$\mathbf{4.27 \pm 0.15}$	5.87 ± 0.16
a1a	16.2	16.3	**16.0**	16.3
splice	6.21	6.71	**5.75**	8.78
svmguide1	**2.92**	3.20	3.35	4.50
w1a	**2.09**	2.26	2.18	2.79

Table 2: Test error (%) of several ensemble learning algorithms using decision stumps

data set	SVM-Perc	AdaBoost-Perc	LPBoost-Perc
twonorm	$\mathbf{2.55 \pm 0.03}$	3.11 ± 0.04	3.52 ± 0.05
twonorm-n	$\mathbf{2.75 \pm 0.05}$	4.53 ± 0.10	6.89 ± 0.11
threenorm	$\mathbf{14.6 \pm 0.08}$	17.3 ± 0.11	18.2 ± 0.11
threenorm-n	$\mathbf{16.3 \pm 0.10}$	20.0 ± 0.18	22.1 ± 0.13
ringnorm	$\mathbf{2.46 \pm 0.04}$	36.3 ± 0.14	37.4 ± 0.13
ringnorm-n	$\mathbf{3.50 \pm 0.09}$	37.8 ± 0.20	39.1 ± 0.15
australian	$\mathbf{14.5 \pm 0.17}$	15.7 ± 0.16	16.4 ± 0.17
breast	$\mathbf{3.23 \pm 0.08}$	3.49 ± 0.10	3.80 ± 0.10
german	$\mathbf{24.6 \pm 0.20}$	25.0 ± 0.18	26.4 ± 0.21
heart	$\mathbf{17.6 \pm 0.31}$	$\mathbf{18.2 \pm 0.32}$	19.8 ± 0.32
ionosphere	$\mathbf{6.40 \pm 0.20}$	11.4 ± 0.23	12.1 ± 0.25
pima	$\mathbf{23.5 \pm 0.21}$	24.8 ± 0.20	26.4 ± 0.19
sonar	$\mathbf{15.6 \pm 0.40}$	19.8 ± 0.43	22.5 ± 0.47
votes84	$\mathbf{4.43 \pm 0.14}$	$\mathbf{4.37 \pm 0.16}$	4.92 ± 0.16
a1a	**15.7**	20.0	18.6
splice	**10.4**	13.7	14.7
svmguide1	**3.10**	3.28	3.62
w1a	**1.91**	2.35	2.13

Table 3: Test error (%) of several ensemble learning algorithms using perceptrons

data set	SVM-Dec	AdaBoost-Dec	LPBoost-Dec
twonorm	$\mathbf{2.87 \pm 0.04}$	3.74 ± 0.05	4.80 ± 0.06
twonorm-n	$\mathbf{3.10 \pm 0.05}$	6.46 ± 0.09	7.89 ± 0.10
threenorm	$\mathbf{15.0 \pm 0.11}$	16.8 ± 0.09	18.3 ± 0.10
threenorm-n	$\mathbf{16.8 \pm 0.15}$	20.2 ± 0.16	22.0 ± 0.12
ringnorm	$\mathbf{2.25 \pm 0.05}$	4.33 ± 0.06	6.00 ± 0.10
ringnorm-n	$\mathbf{2.67 \pm 0.06}$	7.32 ± 0.12	8.76 ± 0.12
australian	14.3 ± 0.18	$\mathbf{13.7 \pm 0.16}$	$\mathbf{13.8 \pm 0.17}$
breast	3.18 ± 0.08	$\mathbf{2.92 \pm 0.09}$	3.94 ± 0.16
german	24.9 ± 0.20	$\mathbf{24.5 \pm 0.17}$	24.9 ± 0.19
heart	$\mathbf{16.8 \pm 0.31}$	19.5 ± 0.33	20.4 ± 0.35
ionosphere	$\mathbf{6.48 \pm 0.19}$	$\mathbf{6.59 \pm 0.19}$	$\mathbf{6.81 \pm 0.22}$
pima	$\mathbf{24.0 \pm 0.24}$	26.1 ± 0.21	26.4 ± 0.20
sonar	$\mathbf{14.7 \pm 0.42}$	19.6 ± 0.45	21.3 ± 0.42
votes84	$\mathbf{4.59 \pm 0.15}$	5.04 ± 0.14	5.95 ± 0.17
a1a	**15.7**	18.8	20.3
splice	3.77	**2.94**	3.77
svmguide1	3.28	3.22	**3.17**
w1a	**2.37**	2.53	3.01

Table 4: Test error (%) of several ensemble learning algorithms using decision trees

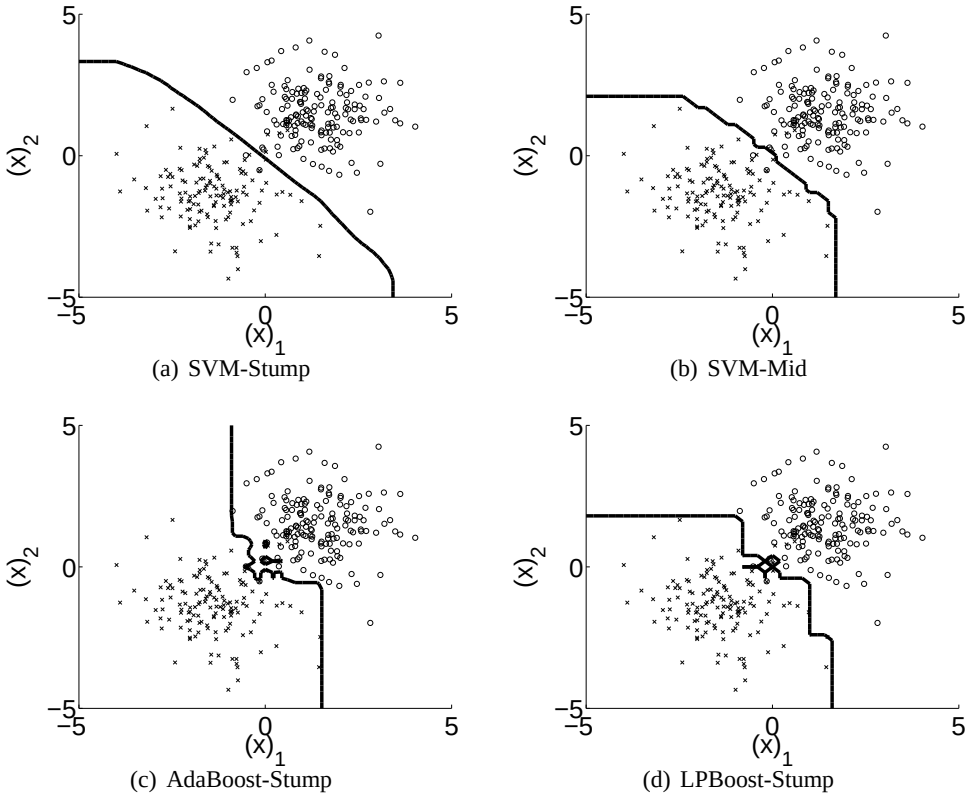
7.1 Comparison of Ensemble Learning Algorithms

Tables 2, 3, and 4 show the test performance of several ensemble learning algorithms on different base hypothesis sets.[7] We can see that SVM-Stump, SVM-Perc, and SVM-Dec are usually better than AdaBoost and LPBoost with the same base hypothesis set, especially for the cases of decision stumps and perceptrons. In noisy data sets, SVM-based infinite ensemble learning always significantly outperforms AdaBoost and LPBoost. These results demonstrate that it is beneficial to go from a finite ensemble to an infinite one with suitable regularization. When comparing the two boosting approaches, LPBoost is at best comparable to AdaBoost on a small number of the data sets, which suggests that the success of AdaBoost may not be fully attributed to its connection to (P_3) or (P_4).

Note that SVM-Stump, SVM-Mid, AdaBoost-Stump, and LPBoost-Stump usually generate different kinds of ensembles: SVM-Stump produces infinite and nonsparse ones; SVM-Mid produces finite and nonsparse ones; AdaBoost-Stump produces finite and sparse ones; LPBoost-Stump produces finite and even sparser ones (since some of the selected h_t may end up having $w_t = 0$). In Table 2, we see that SVM-Stump often outperforms SVM-Mid, which is another evidence that an infinite ensemble could help. Interestingly, SVM-Mid often performs better than AdaBoost-Stump, which means that a nonsparse ensemble introduced by minimizing the ℓ_2-norm of w is better than a sparse one.

In Figure 3, we further illustrate the difference between the finite and infinite ensemble learning algorithms by a simplified experiment. We show the decision boundaries generated by the four algorithms on 300 training examples from the 2-D version of the twonorm data set. The Bayes-

7. For the first 14 rows of Tables 2, 3, 4, and 5, results that are as significant as the best ones are marked in bold; for the last 4 rows, the best results are marked in bold.



(a) SVM-Stump (b) SVM-Mid

(c) AdaBoost-Stump (d) LPBoost-Stump

Figure 3: Decision boundaries of ensemble learning algorithms on a 2-D twonorm data set

optimal decision boundary is the line $(x)_1 + (x)_2 = 0$. We can see that SVM-Stump produces a decision boundary close to the optimal, SVM-Mid is slightly worse, while AdaBoost-Stump and LPBoost-Stump fail to generate a decent boundary. SVM-Stump obtains the smooth boundary by averaging over infinitely many decision stumps; SVM-Mid can also generate a smooth boundary by constructing a nonsparse ensemble over a finite number of decision stumps. Nevertheless, both LPBoost-Stump and AdaBoost-Stump, for which sparsity could be observed from the axis-parallel decision boundaries, do not have the ability to approximate the Bayes optimal boundary well. In addition, as can be seen near the origin point of Figure 3(c), AdaBoost-Stump could suffer from overfitting the noise.

Note that traditional ensemble learning and our SVM-based framework differ in the concept of sparsity. As illustrated in Subsection 2.2, traditional ensemble learning prefers sparse ensemble classifiers, that is, ensembles that include a small number of hypotheses. Our framework works with an infinite number of hypotheses, but results in a sparse classifier in the support vector domain. Both concepts can be justified with various generalization bounds (Freund and Schapire, 1997; Graepel et al., 2005). Nevertheless, our experimental results indicate that sparse ensemble classifiers are sometimes not sophisticated enough in practice, especially when the base hypothesis set is simple. For example, when using the decision stumps, a general data set may require many of them to describe a suitable decision boundary. Thus, AdaBoost-Stump and LPBoost-Stump could be limited by the finiteness and sparsity restrictions. The comparison between AdaBoost-Stump and SVM-Mid

data set	SVM-Stump	SVM-Perc	SVM-Dec	SVM-Gauss
twonorm	2.86 ± 0.04	$\mathbf{2.55 \pm 0.03}$	2.87 ± 0.04	2.64 ± 0.05
twonorm-n	3.08 ± 0.06	$\mathbf{2.75 \pm 0.05}$	3.10 ± 0.05	$\mathbf{2.86 \pm 0.07}$
threenorm	17.7 ± 0.10	$\mathbf{14.6 \pm 0.08}$	15.0 ± 0.11	$\mathbf{14.6 \pm 0.11}$
threenorm-n	19.0 ± 0.14	16.3 ± 0.10	16.8 ± 0.15	$\mathbf{15.6 \pm 0.15}$
ringnorm	3.97 ± 0.07	2.46 ± 0.04	2.25 ± 0.05	$\mathbf{1.77 \pm 0.04}$
ringnorm-n	5.56 ± 0.11	3.50 ± 0.09	2.67 ± 0.06	$\mathbf{2.05 \pm 0.07}$
australian	$\mathbf{14.4 \pm 0.21}$	$\mathbf{14.5 \pm 0.17}$	$\mathbf{14.3 \pm 0.18}$	14.7 ± 0.18
breast	$\mathbf{3.11 \pm 0.08}$	$\mathbf{3.23 \pm 0.08}$	$\mathbf{3.18 \pm 0.08}$	3.53 ± 0.10
german	$\mathbf{24.7 \pm 0.18}$	$\mathbf{24.6 \pm 0.20}$	$\mathbf{24.9 \pm 0.20}$	$\mathbf{24.5 \pm 0.21}$
heart	$\mathbf{16.4 \pm 0.27}$	17.6 ± 0.31	$\mathbf{16.8 \pm 0.31}$	17.5 ± 0.31
ionosphere	8.13 ± 0.17	$\mathbf{6.40 \pm 0.20}$	$\mathbf{6.48 \pm 0.19}$	$\mathbf{6.54 \pm 0.19}$
pima	24.1 ± 0.23	$\mathbf{23.5 \pm 0.21}$	24.0 ± 0.24	$\mathbf{23.5 \pm 0.20}$
sonar	16.6 ± 0.42	15.6 ± 0.40	$\mathbf{14.7 \pm 0.42}$	$\mathbf{15.5 \pm 0.50}$
votes84	4.76 ± 0.14	$\mathbf{4.43 \pm 0.14}$	$\mathbf{4.59 \pm 0.15}$	$\mathbf{4.62 \pm 0.14}$
a1a	16.2	**15.7**	**15.7**	16.2
splice	6.21	10.4	**3.77**	9.56
svmguide1	**2.92**	3.10	3.28	3.12
w1a	2.09	**1.92**	2.37	2.03

Table 5: Test error (%) of SVM with radial basis function kernels

indicates that the second restriction could be crucial. On the other hand, our framework (SVM-Stump), which suffers from neither restrictions, can perform better by averaging over an infinite number of hypotheses.

7.2 Comparison to Gaussian Kernel

In Table 5, we compare all the radial basis function kernels. We can see that SVM-Gauss, being the state-of-the-art setting (Hsu et al., 2003), usually performs well. Its superior performance on the artificial data sets is because they are generated from certain Gaussian distributions. Nevertheless, all SVM-Stump, SVM-Perc, and SVM-Dec outperform SVM-Gauss on some data sets, which demonstrates that they could achieve decent test performances as well.

Furthermore, SVM-Perc and SVM-Gauss share almost indistinguishable performance on the real-world data sets, which is possibly because they both use the ℓ_2-norm distance for measuring similarity. In addition, as discussed in Subsection 6.4, SVM-Perc enjoys the benefit of faster parameter selection. For example, in our experiments, SVM-Gauss involves solving 550 optimization problems, but SVM-Perc deals with only 55 problems. Table 6 shows the speed comparison.[8] We can qualitatively see that SVM-Stump and SVM-Perc are much faster than SVM-Dec and SVM-Gauss.

8. Solving each SVM optimization problem heavily depends on the condition number of the Gram matrix (which depends on the kernel) and the soft-margin parameter C. Thus, it is not easy to compare the training time between different kernels and the numbers here are meant to be interpreted qualitatively. Similar results are observed when using real-world data sets as well (Lin, 2005).

data set	SVM-Stump	SVM-Perc	SVM-Dec	SVM-Gauss
twonorm	1.34	1.44	19.5	23.1
threenorm	1.69	1.69	23.1	31.1
ringnorm	1.50	1.60	23.7	27.9

Table 6: Parameter selection time (sec.) on a dual 1.7 GHz Intel Xeon CPU machine

Since SVM-Perc and SVM-Gauss perform similarly on real-world data sets, the benefit of faster parameter selection makes SVM-Perc a favorable choice in practice. Furthermore, SVM-Stump, albeit slightly worse than SVM-Perc or SVM-Gauss, could still be a useful alternative in some applications where decision stump ensemble models are preferred, such as those described in Subsection 6.4.

8. Conclusion

We derived two novel kernels based on the infinite ensemble learning framework. The stump kernel embodies infinitely many decision stumps, and the perceptron kernel embodies infinitely many perceptrons. These kernels can be simply evaluated by the ℓ_1- or ℓ_2-norm distance between examples. We also explained that the Laplacian-RBF kernel embodies infinitely many decision trees. SVM equipped with the kernels can generate infinite and nonsparse ensembles, which are usually more robust than finite and sparse ones.

Experimental comparisons with AdaBoost and LPBoost showed that SVM with the kernels usually performs much better than boosting approaches with the same base hypothesis set, especially in the cases of decision stumps or perceptrons. Therefore, existing applications that use boosting with decision stumps, perceptrons, or decision trees may be improved by switching to SVM with the corresponding kernel.

In addition, we showed that the perceptron kernel shares similar performance to the Gaussian-RBF kernel, while the former benefits from faster parameter selection. This property makes the perceptron kernel favorable to the Gaussian-RBF kernel in practice.

Acknowledgments

We thank Yaser Abu-Mostafa, Amrit Pratap, Kai-Min Chung, and the anonymous reviewers for valuable suggestions. Most of the work was done in 2005, in which Hsuan-Tien Lin was supported by the Caltech Center for Neuromorphic Systems Engineering under the US NSF Cooperative Agreement EEC-9402726, and Ling Li was sponsored by the Caltech SISL Graduate Fellowship.

Appendix A. Derivation of Kernels

$$\begin{aligned}
\mathcal{K}_{\mathcal{S}}(x,x') &= \sum_{q\in\{-1,+1\}}\sum_{d=1}^{D}\int_{L_d}^{R_d} r_{\mathcal{S}}^2 s_{q,d,\alpha}(x)s_{q,d,\alpha}(x')\,d\alpha \\
&= 2r_{\mathcal{S}}^2\sum_{d=1}^{D}\int_{L_d}^{R_d}\operatorname{sign}\big((x)_d-\alpha\big)\operatorname{sign}\big((x')_d-\alpha\big)\,d\alpha \\
&= 2r_{\mathcal{S}}^2\sum_{d=1}^{D}\left((R_d-L_d)-2\int_{\min((x)_d,(x')_d)}^{\max((x)_d,(x')_d)} d\alpha\right) \\
&= 2r_{\mathcal{S}}^2\sum_{d=1}^{D}(R_d-L_d)-4r_{\mathcal{S}}^2\sum_{d=1}^{D}\left|(x)_d-(x')_d\right| \\
&= 2r_{\mathcal{S}}^2\sum_{d=1}^{D}(R_d-L_d)-4r_{\mathcal{S}}^2\left\|x-x'\right\|_1.
\end{aligned}$$

$$\begin{aligned}
\mathcal{K}_{\mathcal{P}}(x,x') &= r_{\mathcal{P}}^2\int_{\|\theta\|_2=1}\left[\int_{-R}^{R} p_{\theta,\alpha}(x)p_{\theta,\alpha}(x')\,d\alpha\right]d\theta \\
&= r_{\mathcal{P}}^2\int_{\|\theta\|_2=1}\left[\int_{-R}^{R} s_{+1,1,\alpha}(\theta^T x)s_{+1,1,\alpha}(\theta^T x')\,d\alpha\right]d\theta \\
&= 2r_{\mathcal{P}}^2\int_{\|\theta\|_2=1}\left(R-\left|\theta^T x-\theta^T x'\right|\right)d\theta \\
&= 2r_{\mathcal{P}}^2\int_{\|\theta\|_2=1}\left(R-\left\|x-x'\right\|_2\left|\cos\left(\operatorname{angle}\left\langle\theta,x-x'\right\rangle\right)\right|\right)d\theta \\
&= 2r_{\mathcal{P}}^2\Theta_D R-2r_{\mathcal{P}}^2\Xi_D\left\|x-x'\right\|_2.
\end{aligned}$$

The last equality comes from the symmetry when integrating over every possible θ.

References

Annalisa Barla, Francesca Odone, and Alessandro Verri. Histogram intersection kernel for image classification. In *Proceedings of the 2003 Conference on Image Processing*, volume 3, pages 513–516, 2003.

Brad J. C. Baxter. Conditionally positive functions and *p*-norm distance matrices. *Constructive Approximation*, 7(1):427–440, 1991.

Christian Berg, Jens P. R. Christensen, and Paul Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*. Springer-Verlag, 1984.

Sabri Boughorbel, Jean-Philippe Tarel, and Nozha Boujemaa. Generalized histogram intersection kernel for image recognition. In *Proceedings of the 2005 Conference on Image Processing*, volume 3, pages 161–164, 2005.

Leo Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1517, 1999.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A library for support vector machines*, 2001a. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chih-Chung Chang and Chih-Jen Lin. Training ν-support vector classifiers: Theory and algorithms. *Neural Computation*, 13(9):2119–2147, 2001b.

Olivier Chapelle, Patrick Haffner, and Vladimir N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999.

Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1–3):225–254, 2002.

Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

François Fleuret and Hichem Sahbi. Scale-invariance of support vector machines based on the triangular kernel. In *Proceedings of the Third International Workshop on Statistical and Computational Theories of Vision*, 2003.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

Yoav Freund and Robert 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, 1997.

Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

Thore Graepel, Ralf Herbrich, and John Shawe-Taylor. PAC-Bayesian compression bounds on the prediction error of learning algorithms for classification. *Machine Learning*, 59(1–2):55–76, 2005.

Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1458–1465, 2005.

Trevor Hastie and Robert Tibshirani. *Generalized Additive Models*. Chapman and Hall, 1990.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, second edition, 1999.

Seth Hettich, Catherine L. Blake, and Christopher J. Merz. UCI repository of machine learning databases, 1998. Downloadable at http://www.ics.uci.edu/~mlearn/MLRepository.html.

Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–91, 1993.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, National Taiwan University, 2003.

S. Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

Ling Li and Hsuan-Tien Lin. Optimizing 0/1 loss for perceptrons by random coordinate descent. In *Proceedings of the 2007 International Joint Conference on Neural Networks*, pages 749–754, 2007.

Hsuan-Tien Lin. Infinite ensemble learning with support vector machines. Master's thesis, California Institute of Technology, 2005.

Hsuan-Tien Lin and Ling Li. Novel distance-based SVM kernels for infinite ensemble learning. In *Proceedings of the 12th International Conference on Neural Information Processing*, pages 761–766, 2005a.

Hsuan-Tien Lin and Ling Li. Analysis of SAGE results with combined learning techniques. In P. Berka and B. Crémilleux, editors, *Proceedings of the ECML/PKDD 2005 Discovery Challenge*, pages 102–113, 2005b.

Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. J. Smola, editors, *Advanced Lectures on Machine Learning*, pages 118–183. Springer-Verlag, 2003.

Charles A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2(1):11–22, 1986.

John C. 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 185–208. MIT Press, 1999.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Gunnar Rätsch, Takashi Onoda, and Klaus-Robert Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

Gunnar Rätsch, Ayhan Demiriz, and Kristin P. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48(1–3):189–218, 2002.

Gunnar Rätsch, Sebastian Mika, Bernhard Schölkopf, and Klaus-Robert Müller. Constructing boosting algorithms from SVMs: an application to one-class classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1184–1199, 2002.

Michael Reed and Barry Simon. *Functional Analysis*. Academic Press, revised and enlarged edition, 1980.

Saharon Rosset, Ji Zhu, and Trevor Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.

Saharon Rosset, Grzegorz Swirszcz, Nathan Srebro, and Ji Zhu. ℓ_1 regularization in infinite dimensional feature spaces. In N. H. Bshouty and C. Gentile, editors, *Learning Theory: 20th Annual Conference on Learning Theory*, volume 4539 of *Lecture Notes in Computer Science*, pages 544–558. Springer-Verlag, 2007.

Bernhard Schölkopf and Alex J. Smola. *Learning with Kernels*. MIT Press, 2002.

Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

Christopher K. I. Williams. Computation with infinite neural networks. *Neural Computation*, 10 (5):1203–1216, 1998.

Journal of Machine Learning Research 9 (2008) 313-337 Submitted 1/06; Revised 10/07; Published 2/08

Algorithms for Sparse Linear Classifiers in the Massive Data Setting

Suhrid Balakrishnan SUHRID@CS.RUTGERS.EDU
Department of Computer Science
Rutgers University
Piscataway, NJ 08854, USA

David Madigan MADIGAN@STAT.COLUMBIA.EDU
Department of Statistics
Columbia University
New York, NY 10027, USA

Editor: Peter Bartlett

Abstract

Classifiers favoring sparse solutions, such as support vector machines, relevance vector machines, LASSO-regression based classifiers, etc., provide competitive methods for classification problems in high dimensions. However, current algorithms for training sparse classifiers typically scale quite unfavorably with respect to the number of training examples. This paper proposes online and multi-pass algorithms for training sparse linear classifiers for high dimensional data. These algorithms have computational complexity and memory requirements that make learning on massive data sets feasible. The central idea that makes this possible is a straightforward quadratic approximation to the likelihood function.

Keywords: Laplace approximation, expectation propagation, LASSO

1. Introduction

We consider the problem of learning high-dimensional sparse linear classifiers from large numbers of training examples. A number of different applications from finance, text mining, and bioinformatics motivate this work. We concern ourselves specifically with binary classification and consider L_1-regularized logistic and probit regression models. Such models have provided excellent predictive accuracy in many applications (see, for example, Genkin et al., 2007; Figueiredo and Jain, 2001; Shevade and Keerthi, 2003) and attack overfitting and variable selection in a unified manner. L_1-regularization and a maximum *a posteriori* (MAP) Bayesian analysis with so-called Laplacian priors yield identical results (Tibshirani, 1996) and in order to streamline our presentation, we adopt the Bayesian approach. Many training algorithms now exist for L_1-logistic regression that can handle high-dimensional input vectors (Hastie et al., 2004; Shevade and Keerthi, 2003; Koh et al., 2007). However, these algorithms generally begin with a "load data into memory" step that precludes applications with large numbers of training examples. More precisely, consider a training data set that comprises t examples each of dimension d. Due to matrix multiplications on $t \times t$ or $d \times d$ matrices, typical computational time requirements are $O(t^3 + d^3)$, with memory requirements that are $O(td + d^2)$. In our target applications, both t and d can exceed 10^6 so standard algorithms become impractical.

This paper presents two basic algorithms for learning L_1-logistic and/or probit regression models. Both operate in the data streaming model, by which we mean that they scan the data sequentially, and never require storing processed observations. The first algorithm we present is an online algorithm which sequentially processes each observation only once. This algorithm is provably non-divergent and uses in the worst case $O(d^2)$ time and $O(d^2)$ space to assimilate each new training example (note that both costs are constant with respect to the number of observations, t). Further, if the input data are sparse, the practical computational cost can be significantly lower.

For massive data sets where t is constant, that is, when given a fixed training data set, we present a second algorithm that allows practitioners to trade-off computational time for improved accuracy. This multi-pass algorithm (the MP algorithm) also processes data sequentially but makes a small constant number of extra passes over the data set. Hence, this sequential algorithm provides results similar to those of batch algorithms for this problem. The MP algorithm's computational cost is a constant factor higher and memory costs are essentially the same as those of the online algorithm. Finally, we propose the RMMP (Reduced Memory MP) algorithm that has significantly lower worst case memory costs, $O(d+k^2)$ (where $k \ll d$) and the same computational costs as the MP algorithm (thus both computational and memory costs are essentially linear in t and d). We will comment on the similarities and differences of our technique to other learning algorithms, in particular other online algorithms, in the following sections.

2. Background and Notation

Throughout this manuscript, we concern ourselves with the task of binary classification, with class labels $y \in \{0,1\}$. The training data comprise t labeled training examples, that is, $D_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^t$, with input vectors $\mathbf{x}_i = [x_{i1}, \ldots, x_{id}]^T$ in $\mathbb{R}^d$ and corresponding labels y_i, $i = 1, \ldots, t$. We consider probabilistic classifiers of the form:

$$p(y = 1|\mathbf{x}) = \Phi(\beta^T \mathbf{x})$$

where $\beta \in \mathbb{R}^d$ is a vector of regression parameters and $\Phi(\cdot)$ is a link function. We restrict our analytical results to the two most commonly used link functions, the probit $\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$ and logistic $\Phi(z) = \frac{e^z}{1+e^z}$ link functions.

The machine learning problem is thus to estimate the parameters β, in the light of the training data D_t. We tailor our results towards high input dimension, that is, large d, and large numbers of training vectors, large t. Viewing the learning problem as one of Bayesian inference, we work with the posterior distribution of the parameters β conditioned on a labeled training data set D_t, given a prior distribution on the parameters β:

$$p(\beta|D_t) \propto \left(\prod_{i=1}^{t} p(y_i|\beta) \right) p(\beta). \tag{1}$$

The quantity on the left hand side of (1) is the required posterior distribution of β given the data set D_t, while the second term on the right hand side is the prior distribution on β, which we will specify momentarily. The first term on the right hand side is the likelihood:

$$\prod_{i=1}^{t} p(y_i|\beta) = \prod_{i=1}^{t} \left(y_i \Phi(\beta^T \mathbf{x}_i) + (1-y_i)(1-\Phi(\beta^T \mathbf{x}_i)) \right).$$

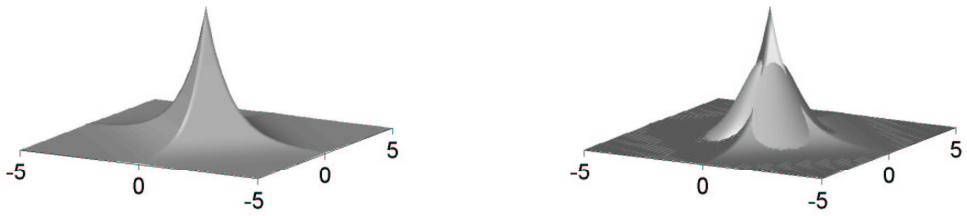
Finding the MAP β leads to the optimization problem we wish to solve (now on the log scale):

$$\begin{aligned} & \max_{\beta} \left(\log p(\beta|D_t)\right) \\ \equiv \quad & \max_{\beta} \left(\sum_{i=1}^{t} \log \left(y_i \Phi(\beta^T \mathbf{x}_i) + (1-y_i)(1-\Phi(\beta^T \mathbf{x}_i)\right) - \log p(\beta) \right). \end{aligned} \tag{2}$$

The prior distribution $p(\beta)$ we pick for the parameters is the LASSO prior (Tibshirani, 1996), a product of independent Laplacian or double-exponential prior distributions on each component β_j (with mean 0):

$$p(\beta_j|\gamma) = \frac{\gamma}{2} e^{-\gamma|\beta_j|}, \gamma > 0, j = 1, \ldots, d.$$

A prior of this form places high probability mass near zero and along individual component axes. It also has heavier tails than a Gaussian distribution—see Figure 1 for plots of the 2-dimensional distributions. It thus favors locations in parameter space with component magnitudes either exactly



(a) (b)

Figure 1: (a) A standard Laplacian distribution, $\gamma = 1$ (b) A superposition of standard (zero mean, unit variance) Gaussian distribution, and the Laplacian distribution showing both the higher probability mass the Laplacian assigns along the axes and at zero as well as its heavier tails.

zero, and hence pruned from our predictive model, or shrunk towards zero. With this prior distribution, (2) presents a convex optimization problem and yields the same solutions as the LASSO (Tibshirani, 1996) and Basis Pursuit (Chen et al., 1999):

$$\begin{aligned} & \max_{\beta} \left(\log p(\beta|D_t)\right) \\ \equiv \quad & \max_{\beta} \left(\sum_{i=1}^{t} \log \left(y_i \Phi(\beta^T \mathbf{x}_i) + (1-y_i)(1-\Phi(\beta^T \mathbf{x}_i)\right) - \gamma \|\beta\|_1 \right). \end{aligned} \tag{3}$$

The parameter γ in the above problem controls the amount of regularization. Figure 2 shows a 2-dimensional visualization of how the objective function of the optimization problem changes

as γ is varied. The choice of the regularization parameter is an important but separate question in itself (Efron et al., 2004; Hastie et al., 2004). Methods such as cross validation can be used to pick its value and algorithms also exist to find solutions for all values of the regularization parameter (commonly called regularization path algorithms). However, we do not address such issues in this manuscript, and we simply assume γ is some fixed, user-specified constant.

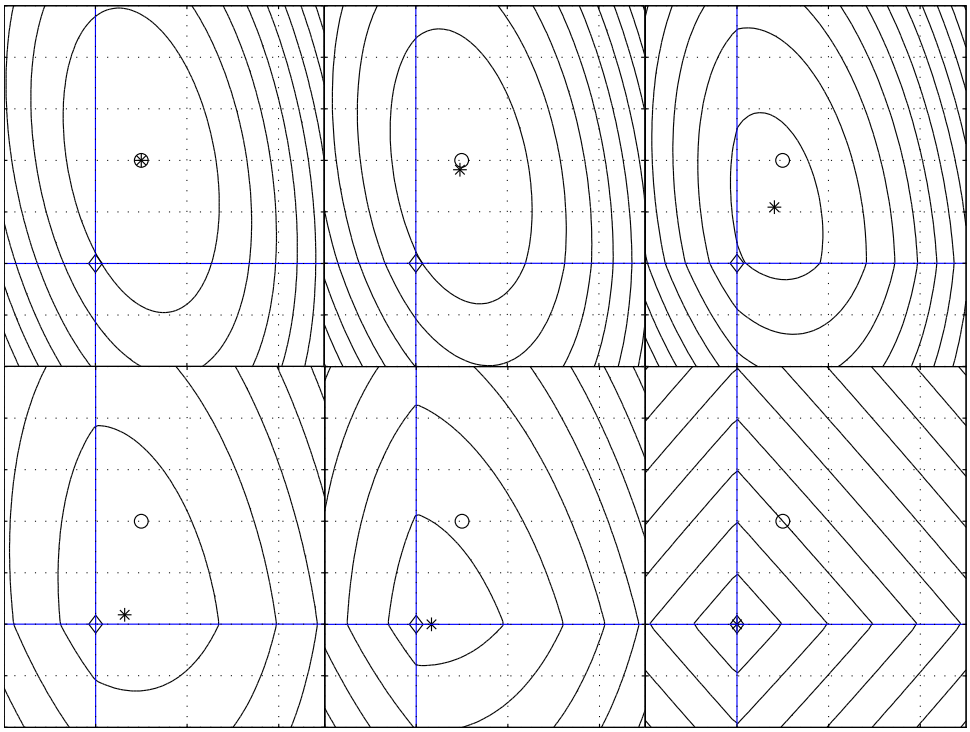


Figure 2: L_1-regularization in two dimensions (i.e., $d = 2$). The axes are the solid lines, the horizontal axis representing β_1 and the vertical axis representing β_2. The diamond represents the origin and the open circle represents the (non-regularized) maximum likelihood solution. The figure shows contours of the function in (3), the objective function, for increasing amounts of regularization (right to left and then top to bottom). The star shows the MAP location. The top row, left figure, shows negligible regularization; the MAP and maximum likelihood estimates coincide and the contours show no L_1-induced discontinuities. The top row, right figure, shows noticeable L_1 effects and the MAP and maximum likelihood solutions differ. The bottom row, middle panel shows enough L_1-regularization to set β_2 to zero (i.e., variable selection has occurred). The bottom row, right panel, shows extreme regularization, where both β_1 and β_2 are zero.

To the best of our knowledge, all existing algorithms solve the above convex optimization problem in the batch setting, that is, by storing the data set D_t in memory and iterating over it (Fu, 1998; Osborne et al., 2000; Zhang, 2002; Shevade and Keerthi, 2003; Genkin et al., 2007; Koh et al., 2007). Consequently, these algorithms cannot be used in the massive data/online scenario, where memory costs dependent on t represent a significant practical impediment. The approach we present

now attempts to overcome this limitation and thereby provide algorithms for training sparse linear classifiers without loading the entire data set into memory.

3. Approximating the Likelihood for Online Learning

The Bayesian paradigm supports online learning in a natural fashion; starting from the prior, the first training example produces a posterior distribution incorporating the evidence from the first example. This then becomes the prior distribution awaiting the arrival of the second example, and so on. In practice, however, except in those cases where the posterior distribution has the same mathematical form as the prior distribution, some form of approximation is required to carry out the sequential updating.

We want to avoid algorithms that begin with a "load data into memory" step and also avoid memory costs that increase with increasing amounts of data. In other words, we want memory costs independent of t. This requirement in turn, necessitates that we "forget" examples after processing them. We achieve this by maintaining the sufficient statistics of a quadratic approximation in β to the log-likelihood of the parameters after incorporating each observation.

We approximate the log-likelihood as:

$$\begin{aligned}\sum_{i=1}^{t}\log(p(y_i|\beta)) &= \sum_{i=1}^{t}\log\left(y_i\Phi(\beta^T\mathbf{x}_i)+(1-y_i)(1-\Phi(\beta^T\mathbf{x}_i)\right)\\ &\approx \sum_{i=1}^{t}\left(a_i(\beta^T\mathbf{x}_i)^2+b_i(\beta^T\mathbf{x}_i)+c_i\right),\end{aligned}$$

where $a_i(\beta^T\mathbf{x}_i)^2+b_i(\beta^T\mathbf{x}_i)+c_i$ approximates $\log\Phi(\beta^T\mathbf{x}_i)$ when $y_i=1$ and approximates $\log(1-\Phi(\beta^T\mathbf{x}_i))$ when $y_i=0$, $i=1,\ldots,t$. In either case the approximation uses a simple Taylor expansion around $\beta_{i-1}^T\mathbf{x}_i$, where β_{i-1} estimates the posterior mode given the first $i-1$ examples, D_{i-1} (Appendix A provides expressions for a_i, b_i for the probit and logistic link functions). We then have:

$$\begin{aligned}\sum_{i=1}^{t}\log(p(y_i|\beta)) &\approx \sum_{i=1}^{t}\left(a_i(\beta^T\mathbf{x}_i)^2+b_i(\beta^T\mathbf{x}_i)+c_i\right)\\ &= \sum_{i=1}^{t}a_i(\beta^T\mathbf{x}_i)(\mathbf{x}_i^T\beta)+\sum_{i=1}^{t}b_i(\beta^T\mathbf{x}_i)+\sum_{i=1}^{t}c_i\\ &= \beta^T\Psi_t\beta+\beta^T\theta_t+\sum_{i=1}^{t}c_i\end{aligned}$$

where:

$$\Psi_t=\sum_{i=1}^{t}a_i\mathbf{x}_i\mathbf{x}_i^T,\text{ and }\theta_t=\sum_{i=1}^{t}b_i\mathbf{x}_i.$$

We now substitute this approximation of the log-likelihood function into Equation (3) to obtain the modified (approximate) optimization problem:

$$\max_{\beta}\left(\log p(\beta|D_t)\right)\approx\max_{\beta}\left(\beta^T\Psi_t\beta+\beta^T\theta_t-\gamma\|\beta\|_1\right). \tag{4}$$

Note that we can ignore the term involving the c_i's, as it is not a function of β. Further, the fixed size $d \times d$ matrix Ψ and the $d \times 1$ vector θ can be updated in an online fashion as data accumulate:

$$\Psi_{t+1} = \Psi_t + a_{t+1}\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T, \text{ and } \theta_{t+1} = \theta_t + b_{t+1}\mathbf{x}_{t+1}. \tag{5}$$

The size of the optimization problem in (4) doesn't depend on t, the size of the data set seen so far. Thus, solving a fixed (with respect to t) size optimization problem allows one to sequentially process labeled data items and march through the data set. In data streaming terminology, the matrix Ψ and the vector θ provide a constant size sketch or summary of the labeled observations seen so far.

A number of questions now present themselves: how good is this approximation? How do we solve the approximate optimization problem efficiently? How does this approach differ from other likelihood approximation schemes (some of which are also quadratic)? Also, the scheme as set up requires $O(d^2)$ memory in the worst case. Since we would like to use this approach for high dimensional data sets, can we reduce the memory requirements?

The remainder of this manuscript addresses these and other questions. First, we consider how to efficiently obtain the MAP solution of (4), the approximate optimization problem.

3.1 The Modified Shooting Algorithm

Recall that we need to find β that solves:

$$\max_{\beta} \left(\beta^T \Psi \beta + \beta^T \theta - \gamma \|\beta\|_1 \right). \tag{6}$$

In the above equation and following discussion, we drop the subscript t from Ψ, θ for notational convenience. This is a convex optimization problem and a number of efficient techniques exist to solve it. Newton's method and other Hessian-based algorithms may be prohibitively expensive as they need $O(d^3)$ computational time in order to construct the Hessian/invert $d \times d$ matrices. Other authors have described good results on the arguably tougher (non-approximate) optimization problem for logistic regression (essentially the terms in Equation 3, but with L_2 regularization of β) with techniques such as fixed memory BFGS (Minka, 2000), modified conjugate gradient (Komarek and Moore, 2005) and cyclic coordinate descent (Zhang and Oles, 2001; Genkin et al., 2007).

In this paper, we employ instead a slight modification of the Shooting algorithm (Fu, 1998), see Algorithm 1. Shooting is essentially a coordinate-wise gradient ascent algorithm, explicitly tailored for convex L_1-constrained regression problems (squared loss). Since our approximate optimization problem is also quadratic, the resulting modifications required are straightforward. The vector Ω in the algorithm is defined as $\Omega = 2\Psi'\beta + \theta$, where Ψ' is the matrix Ψ with its diagonal entries set to zero (see Appendix B for details). This vector is related to the gradient of the differentiable part of the objective function and consequently can be used for optimality checking. Minor variants of this algorithm have been independently proposed by Shevade and Keerthi (2003) and Krishnapuram et al. (2005). Although Fu originally derived the algorithm by taking the limit of a modified Newton-Raphson method, it can also be obtained by a subgradient analysis of the system (subgradients are necessary due to the non-differentiability that the L_1 constraints on β result in, see Appendix B for the derivation).

While one can think of numerous stopping criteria for the algorithm, in this paper we stop when successive iterates are sufficiently close to each other (relatively, and with respect to the L_2

Algorithm 1: The modified Shooting algorithm.

Data: $\Psi, \theta, \beta_0, \gamma$.
β_0 is initial β vector.
Ω_j refers to the j'th component of Ω.
Ψ_{jj} refers to the (j,j)'th element of matrix Ψ.
Result: β satisfying (6).
while *not converged* **do**
  for $j \leftarrow 1$ **to** d **do**

$$\beta_j = \begin{cases} 0, & \text{if } |\Omega_j| \leq \gamma \\ \frac{\gamma - \Omega_j}{2\Psi_{jj}}, & \text{if } \Omega_j > \gamma \\ \frac{-\gamma - \Omega_j}{2\Psi_{jj}}, & \text{if } \Omega_j < -\gamma \end{cases}$$

    Update Ω.
  end
end

norm). More precisely, we declare convergence whenever $\|\beta_i - \beta_{i-1}\|_2 / \|\beta_{i-1}\|_2$ is less than some user specified tolerance. Note that β_i is the parameter vector at iteration i, which is obtained after cycling through and updating all d components once.

In the worst case, each iteration of Shooting requires $O(d^2)$ computational time. However, for reasonable amounts of regularization, where the final set of non-zero β values is small, the time requirements are much smaller. Indeed, the practical computational cost is perhaps better reflected by bounds in terms of the sparsity of MAP β. Let m denote the maximum number of non-zero components of β along the solution path to MAP β (hence $m \leq d$). Implemented carefully, Shooting requires $O(md)$ time per iteration (see Appendix B for details). Shooting can be initialized with $\beta_0 = 0$ if no information about the optimal β is known or to an appropriate "warm" starting point.

While coordinate-wise approaches are commonly regarded as slow in the literature (for example, Minka, 2001a), for sparse classifiers, they are much faster (see for example, Shevade and Keerthi, 2003). In our experiments, the Shooting algorithm has proven to be practical even for d in the hundreds of thousands.

4. An Online Algorithm

The quadratic approximation and the Shooting algorithm lead straightforwardly to an online algorithm. After initializing the sketch parameters Ψ_0, θ_0 and the initial parameter vector β_0, process the data set one observation at a time. Calculate the quadratic Taylor series approximation to each observation's log-likelihood at the current estimate of the posterior mode, β_{i-1}, thus finding parameters a_i, b_i. Use these parameters and the observation to update the sketches, Ψ, θ. Now run the modified Shooting algorithm to update the posterior mode, producing β_i and repeat for the next labelled observation—see Algorithm 2.

We show the performance of the online algorithm on a low dimensional simulated data set in Figure 3 (the data generating mechanism is a logistic regression model with $d = 11$, and $t = 100,000$. For details see the Experiments section of the manuscript). As we process greater numbers of observations, the online estimates (the solid lines) improve, that is, get closer to the batch

Algorithm 2: The Online algorithm.

Data: D_t, γ.
Result: For each i, produces β_i, an approximation to the MAP estimate of β for observations $(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_i, y_i)$.
Initialize $\beta_0 = \theta_0 = 0$, $\Psi_0 = 0$, $i = 1$.
while $i < t$ **do**
  Get i'th observation $(\mathbf{x}_i, y_i)$.
  Obtain quadratic approximation to term likelihood at β_{i-1}, that is, obtain a_i, b_i.
  $\Psi_i \leftarrow \Psi_{i-1} + a_i \mathbf{x}_i \mathbf{x}_i^T$.
  $\theta_i \leftarrow \theta_{i-1} + b_i \mathbf{x}_i$.
  $\beta_i \leftarrow$ modified Shooting$(\Psi_i, \theta_i, \beta_{i-1}, \gamma)$
  $i \leftarrow i + 1$.
end

estimates (the dashed lines which we obtain using BBR, Genkin et al. 2007, publicly available software for batch L_1 penalized logistic regression). See Figure 3, where different colors represent different components of MAP β_i. Figure 4 shows individual plots of the online and batch estimates for four representative components of MAP β_i in blue. We also plot the absolute difference between the batch and online estimates in green (dotted line) on the same plot on the right (green) axis. As we expect, after the parameter estimates stabilize, this difference steadily tapers off with increasing amounts of data.

	$t = 2\text{x}10^4$		$t = 6\text{x}10^4$		$t = 10^5$	
β_{true}	**Batch**	**Online**	**Batch**	**Online**	**Batch**	**Online**
0.259	0.244	0.242	0.248	0.247	0.254	0.253
0.761	0.700	0.690	0.743	0.739	0.740	0.737
-0.360	-0.360	-0.356	-0.401	-0.399	-0.394	-0.393
0.876	0.980	0.966	0.918	0.913	0.922	0.919
0.913	0.920	0.907	0.920	0.916	0.931	0.929
-0.302	-0.275	-0.270	-0.327	-0.324	-0.317	-0.315
-0.820	-0.826	-0.814	-0.806	-0.802	-0.819	-0.816
0	0	0	-0.010	-0.010	-0.005	-0.005
0	0.050	0.049	0	0	0.013	0.013
0	0.038	0.037	0.014	0.014	0.013	0.013
-0.319	-0.298	-0.294	-0.318	-0.316	-0.320	-0.319
L_1 Norm	0.066		0.025		0.016	

Table 1: Table with columns showing values of β_{true}, and the MAP estimates of β obtained by the batch algorithm and the online algorithm, for increasing amounts of data on the simulated data set. To aid assessing convergence of the online to the batch estimates, we show the value of the L_1 norm of the adjacent vectors (batch vs. online estimates) in the last row. For this example, $\gamma = 10$ (logistic link function).

In the worst case, the online algorithm requires $O(d^2)$ space and $O(d^2)$ computational time to compute the MAP β for each new observation. Note however, that if the input data has sparsity,

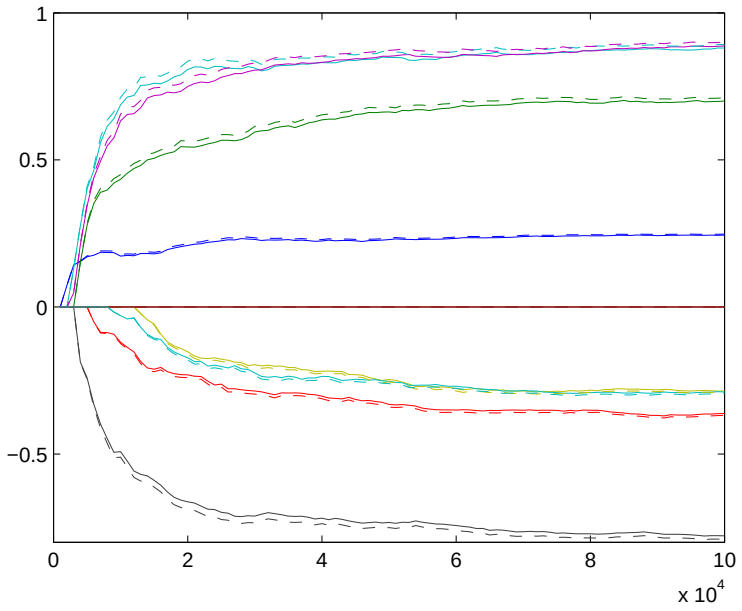


Figure 3: Performance of the online algorithm on a simulated data set, with regularization parameter $\gamma = 100$ (see text for details). The y-axis is the parameter value, the x-axis the number of observations processed, t.

which is true of text data for instance, the algorithm leverages this. Let the maximum number of non-zero components in any $\mathbf{x}$ be f and assume a constant number of iterations of the modified Shooting algorithm. In such case, the practical computational time requirement of the algorithm is $O(f^2 + md)$ per observation (we remind the reader that the md term, is for the cost of the Shooting algorithm—see 3.1). Although the practical memory costs of the algorithm will likely be less than $O(d^2)$, exactly how much less depends heavily on the data, since Ψ (the part of the sketch dominating the memory requirements) is a weighted sum of outer products of the $\mathbf{x}_i$'s. It is possible that even very sparse data may result in the full $O(d^2)$ memory requirement.

Here, we highlight the fact that the online algorithm is accurate and practical if the problem is of low to medium input dimension, but massive in terms of the number of observations. Appendix C proves non-divergence of the algorithm in the infinite data limit.

4.1 Heuristics for Improvement/Issues

While one can also obtain parameter estimates for fixed t (batch problems) using the online algorithm, multiple passes typically provide better estimates, albeit with increased computational cost. Denote by β_* the solution to the exact optimization problem (3) for some fixed t. Since the online algorithm typically initializes itself far from β_*, it is only after processing a sufficient number of examples that the online algorithm's term approximations will start being taken closer to β_*. The update formulae, (5), reveal that for values of $i < t$, both Ψ_i and θ_i are (comparatively) smaller in

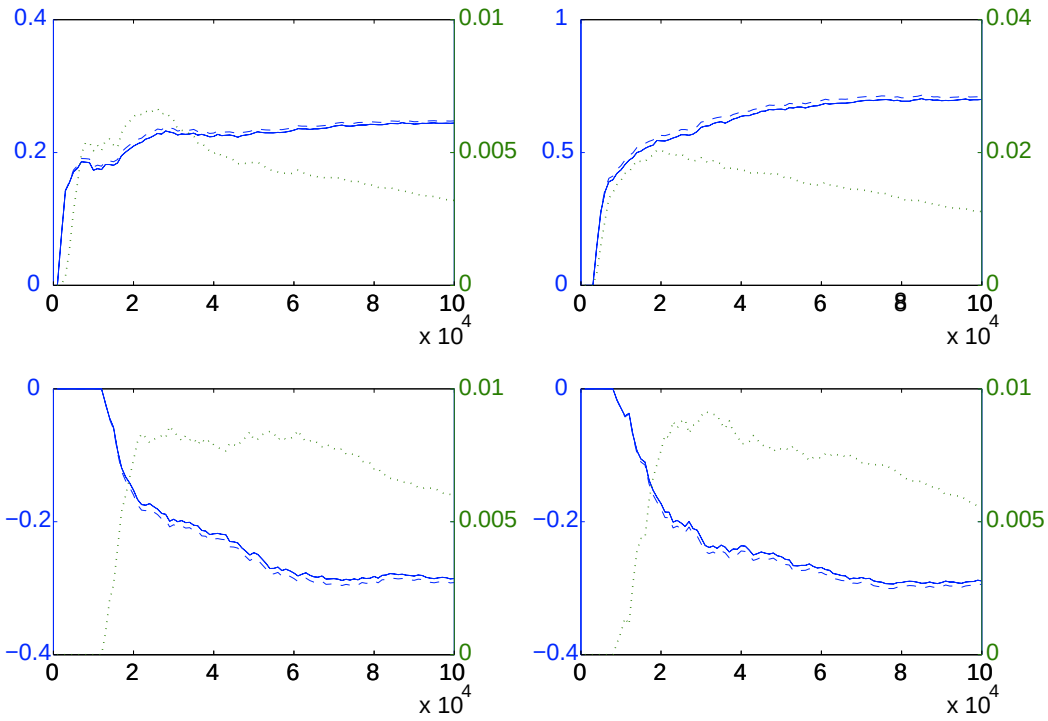


Figure 4: Slightly more detailed version of Figure 3. The panels show four representative parameters from that figure, also showing tapering L_1 loss (dotted green line) between the online and batch algorithm estimates on the right axis (in green). Simulated data set, $\gamma = 100$. Once again, the (left) y-axis is the parameter value and the x-axis the number of observations processed, t.

magnitude than their respective final values, Ψ_t, θ_t. However, the amount of regularization remains relatively fixed at $\gamma\|\beta\|_1$. Hence, if the online algorithm is initialized at $\beta_0 = 0$, for any $i < t$, the output MAP estimate β_i will be more shrunk towards zero than β_*. Figure 3 illustrates this for smaller values of t where the solid lines (approximate MAP estimates) are closer to zero than the dashed lines (exact batch estimates).

This suggests the following two heuristics to improve the quality of estimates from the online algorithm. The first is to increase the amount of regularization gradually as the algorithm processes observations sequentially (via a schedule, linearly say, $\propto t$ from zero initially to the specified value γ at the end of the data set[1]). Less regularization of the first few observations somewhat mitigates the effect of taking term approximations at shrunken parameter estimates.

The second heuristic is for the online algorithm to keep a block of observations in memory temporarily instead of immediately discarding each observation after processing it. The algorithm

1. While the choice of this regularization schedule in this setting is understudied in the literature, asymptotic consistency results for a slightly modified form of the problem may be of theoretical interest. We refer readers to Zou (2006), and the references therein.

then uses the value of the parameter estimates after having seen/processed all the observations in a block to update the sketches for the whole block. Note that this will involve keeping track of the corresponding updates to the sketches for the block (the block's contributions to Ψ and θ). In experiments not reported here, both of these heuristics improve the final online estimates somewhat.

One possibility for improving upon the $O(d^2)$ worst case computational requirement of the online algorithm is as follows. In the infinite data case, in order to obtain sparsity in parameter estimates, the amount of regularization must be allowed to increase as observations accumulate—an increasingly weighty likelihood term will inundate any fixed amount of regularization. In this setting (where we have the freedom to choose the amount of regularization), we can use exactly the same quadratic approximation machinery to pick the value of γ that maximizes the approximate one-step look ahead likelihood (although the expressions for this approximation would be slightly different). The resulting scheme has the flavor of predictive automatic relevance determination as presented in Qi et al. (2004).

The worst case $O(d^2)$ memory requirement of the online algorithm, however, presents a greater challenge. In the next section we outline a multi-pass algorithm based on the same sequential quadratic approximation that improves the accuracy of estimates when applied to finite data sets and also uses less memory than the online algorithm.

5. A Multi-pass Algorithm

The block heuristic of the previous section implies that taking *all* term approximations at the final online algorithm MAP β_t value would certainly produce better estimates of Ψ_t, θ_t. This in turn would lead to a better estimate of β_*.

Therefore, for fixed data sets where computational time restrictions still permit a few passes over the data set, this suggests the following algorithm, which we will refer to as the MP (Multi-Pass) algorithm: Initialize $\beta_0 = \theta_0 = 0$, $\Psi_0 = 0$, $z = 1$. The quantity z will count the number of passes through the data set. Compute Ψ_t, θ_t by the steps in Online Algorithm (Algorithm 2), except take *all* term approximations at the *fixed value* β_z. Note that consequently there is no need for the shooting algorithm during the pass through the data set. Once a pass through the data set is complete, compute a revised estimate of β_* by running modified Shooting, that is, set β_{z+1} =modified Shooting($\Psi_t, \theta_t, \beta_z, \gamma$). Iteratively loop over the data set, appropriately incrementing z.

For a constant number of passes, the MP algorithm has the worst case computational time requirement of $O(td^2)$ to do an equivalent batch MAP β estimation. Once again, if the data set is sparse, this cost is closer in practice to $O(tf^2 + md)$ (the first term is the cost of updating the sketches and the second md term is the cost of the Shooting algorithm).

The worst case memory requirement of the MP algorithm is $O(d^2)$, which is just a constant with respect to t. Expectation Propagation (Minka, 2001b) by contrast requires explicitly storing term approximations and thus has memory costs that scale linearly with t, that is, $O(t)$. The next subsection presents a modification of the MP algorithm that reduces this worst case memory requirement.

5.1 A Reduced Memory Multi-pass Algorithm

The key to reducing the memory requirements of the algorithm in the previous subsection is exploiting the sparsity of β_*. Towards this end, consider the modified Shooting algorithm upon convergence; say β_{MAP} is the sparse converged solution Shooting obtains with inputs Ψ, θ and γ. Now consider the smaller system obtained by only retaining those rows of the vectors, and also corre-

sponding columns for matrices, for which the components of β_{MAP} are nonzero (denoted with a $\tilde{}$). The important observation is that the solution to the reduced size system $\tilde{\beta}_{MAP}$, obtained using $\tilde{\Psi}, \tilde{\theta}$ and $\tilde{\Omega}$, has exactly the same nonzero components as β_{MAP} obtained for the full system.

We use this fact to derive the RMMP (Reduced Memory Multi-Pass) algorithm, Algorithm 3. The central idea is to use the optimality criteria for the Shooting algorithm to determine which components of β to keep track of. Call this set S, the active set, which is fixed during every iteration. Specifically, we set $S = \{j : |\Omega_j| \geq \gamma\}$. That is, the active set is the set of variables that are either nonzero and optimal or variables that violate optimality at the start of a pass (the corresponding nonzero elements of the vectors/matrices are denoted by their previous symbols but with a $\tilde{}$ above them). Now, during the pass we keep track of the much smaller matrix $\tilde{\Psi}$, while also keeping track of the unmodified/original full length vectors θ and Ω. The update for θ is unchanged and Appendix B shows how to perform the update for the full length vector Ω in small space. The algorithm continues by using $\tilde{\Psi}, \Omega$, and θ from the latest pass to re-estimate the active set, S and so on.

A desirable consequence of the setup is that no new approximation is introduced. The search for the optimal parameter values is slightly more involved though, now proceeding iteratively by first identifying candidate nonzero components of β_{MAP}, and then refining the estimates for these components. We can employ the same stopping criteria as for modified Shooting algorithm.

Algorithm 3: The RMMP algorithm.

Data: fixed data set D_t, γ.
Result: β_z, the MAP estimate of β that solves (3).
Initialize $\beta_0 = 0$, $S = \{\}, z = 1$.
while *not converged* **do**
  Set $\theta = 0$, $\tilde{\Psi} = 0$, $i = 1$.
  for $i = 1, 2, \ldots, t$ **do**
    Get i'th observation $(\mathbf{x}_i, y_i)$.
    Obtain quadratic approximation to term likelihood at β_{z-1}, that is, obtain a_i, b_i.
    $\tilde{\Psi} \leftarrow \tilde{\Psi} + a_i(\tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T)$.
    $\theta \leftarrow \theta + b_i \mathbf{x}_i$.
    Update Ω.
  end
  $\beta_z \leftarrow$ modified Shooting$(\tilde{\Psi}, \tilde{\theta}, \tilde{\beta}_{z-1}, \gamma)$.
  Obtain new active set $S = \{j : |\Omega_j| \geq \gamma\}$.
  $z \leftarrow z + 1$.
end

Note that memory requirements are now $O(d + k^2)$, where k is the number of variables in the largest active set. However, we can be even more stringent and set k to be a user specified constant provided k is bigger than the final number of nonzero components of β_*. Typically, setting k very close to this limit results in some loss of accuracy and the cost of a few more passes over the data for convergence. The worst case computational time requirements for a constant number of passes, are still $O(td^2)$ to do an equivalent batch MAP β estimation. Under the same sparsity assumptions as in previous sections, in practice this cost is better quantified as $O(t(k^2 + f^2) + kd)$ (again, the first term is the cost associated with updating the sketches and the second term is the cost of Shooting).

We now draw attention to a few practical considerations about the RMMP algorithm. The first is that although we consider initializing the parameter vector to zero, $\beta_0 = 0$, better guesses of β_0 (guesses closer to the MAP β) would likely result in fewer passes for convergence. Further, given we do initialize at zero, the first pass is completed very rapidly. This is because no outer products are computed, since the active set is initialized as the empty set; the first pass is used simply to determine the size and components of the active set and the parameter estimates for the next iteration are still zero, $\beta_1 = 0$. Typically, setting the reduced memory parameter k to be larger than this first active set size results in further RMMP iterations mimicking iterations of the MP algorithm. This is seen by observing two facts. One, for both algorithms, the only components that change in successive iterations are those in the active set (components that are either non-zero and optimal or not optimal). Two, in a typical search path for the MAP β, the size of the active set decreases (and finally stabilizes) as the MAP β is honed in on. Both of these observations together imply that if we start the RMMP algorithm with enough memory allotted to look at all possibly relevant β components, we will follow the MP search path (as a motivating example, consider that setting $k = d$ results in the MP algorithm exactly).

Another consideration is a very useful practical advantage of the proposed algorithm: knowledge of Ω implies the practitioner can confirm when convergence to β_* has/has not occurred. In practice, for numerical stability, slightly expanding the active set seems to be a good heuristic. In our experiments that follow, we do so only if we have extra space (if k is bigger than the number of variables in the current active set, for any iteration) in two ways: 1. We retain in the active set variables that were in the active set in the previous iteration and, 2. we add to the active set components that are *close to* violating optimality (close in terms of a threshold, $\tau < 1$. This amounts to replacing the rule in Algorithm 3 with $S = \{j : |\Omega_j| \geq \tau\gamma\}$).

In the next section, we place our work in the context of existing literature on similar problems.

6. Related Work

Although the Bayesian paradigm facilitates sequential updating of the posterior distribution (online learning) in a natural way, some form of approximation is almost always necessary for practical applications. Approximating the posterior distribution at every stage by a multivariate Gaussian distribution (which implies a quadratic approximation of the log posterior distribution) seems a natural first step backed by asymptotic Bayesian central limit results that imply this approximation will get better and better with the addition of data (Bernardo and Smith, 1994).

Indeed, approximating the log-likelihood function by a quadratic polynomial is a standard technique in Bayesian learning applications; see for example Laplace approximation (Kass and Raftery, 1995; MacKay, 1995), Assumed Density Filtering (ADF)/Expectation Propagation (EP) (Minka, 2001b), some variational approximation methods such as Jaakkola and Jordan (2000) and in Bayesian online learning (Opper, 1998). We would like to stress here that many of the above schemes are for the harder task of approximate inference—we are concerned only with the easier problem of approximate convex optimization. The similarities in the approaches are confined to the nature of the approximate (Gaussian) posterior.

The next sections describes results we obtained on some simulated as well as real examples using the proposed algorithms.

7. Experiments

We now present examples illustrating the application of the Online, MP and RMMP algorithms to simulated data sets, where we control the data generating mechanism, and some real data sets. We make logistic regression comparisons to results obtained using BBR (Genkin et al., 2007). BBR is publicly available software for Bayesian binary logistic regression that handles the Laplacian prior. We make probit regression comparisons to results obtained using a batch EM algorithm for Laplacian prior based probit regression (we implemented a slightly modified version of the algorithm in Figueiredo and Jain, 2001). We generally do not present prediction accuracy results here as our goal is to obtain accurate, that is, close to batch, parameter values. What we wish to accomplish with the experiments is demonstrate practical efficiency and applicability of the algorithms. In so doing and by obtaining essentially identical parameter estimates to batch algorithms, our predictive performance will mirror those of the batch algorithms. Several papers provide representative predictive performance results for L_1-regularized classifiers, for example, Genkin et al. (2007); Figueiredo and Jain (2001).

We carried out all the experiments on a standard Windows OS based 2Ghz processor machine with 1GB RAM. For all experiments we set the modified Shooting convergence tolerance to be 10^{-6}, and $\tau = 0.8$ (for experiments involving the RMMP algorithm).

We use the following data sets:

- Simulated data sets: d=11, t=10,000. The data generating mechanism is either a probit or logistic regression model with one intercept term and 10 model coefficients, for a total of 11 fixed parameters. Of the ten model variables, three are intentionally set as redundant variables (set with zero coefficients in the model). The data vectors $\mathbf{x}$, are draws from i.i.d. Gaussian distributions with mean zero and unit variance. For the experiments with the online algorithm (Figure 3, Table 1), we used the same model parameters as above, but with $t = 100,000$ and only a logistic regression model.
- ModApte training data set: $d = 21,989$, $t = 9,603$. This is a text data set, the ModApte split of Reuters-21578 (Lewis, 2004). We examine one particular category, "earn", to which we fit a logistic regression model.
- BIG-RCV data set: $d = 288,062$, $t = 421,816$, a data set constructed from the RCV1-v2 data set (Lewis et al., 2004). It consists of the training portion of the LYRL2004 split plus 2 parts of the test data (the test data is made publicly available in $4 \approx 350$ MB parts)—see Figure 5. We also use just the training portion of RCV1-v2 in some experiments. RCV1-v2 training data set : $d = 47,236$, $t = 23,149$ (the features in this data set are a particular subset of the features in BIG-RCV). Our results are for a single topic "ECAT", whether or not a document is related to economics.

7.1 Results

The low dimensional simulated data set highlights typical results we obtain with the Online algorithm and the MP algorithm (the RMMP algorithm is not of practical significance in this case). See Table 2. Each column in the table is an 11-dimensional vector which is the MAP β estimate of the parameter values (as a reminder, the true parameter values used to generate the data can be seen in Table 1). The parameter estimates from the Online algorithm are quite close to batch estimates, likely due to the relatively large data set size (t being large relative to d). Also, with very few passes over the data set, denoted as before by the variable z, we obtain parameter estimates practically identical to those obtained by the batch algorithm. The results in the table are typical for both link

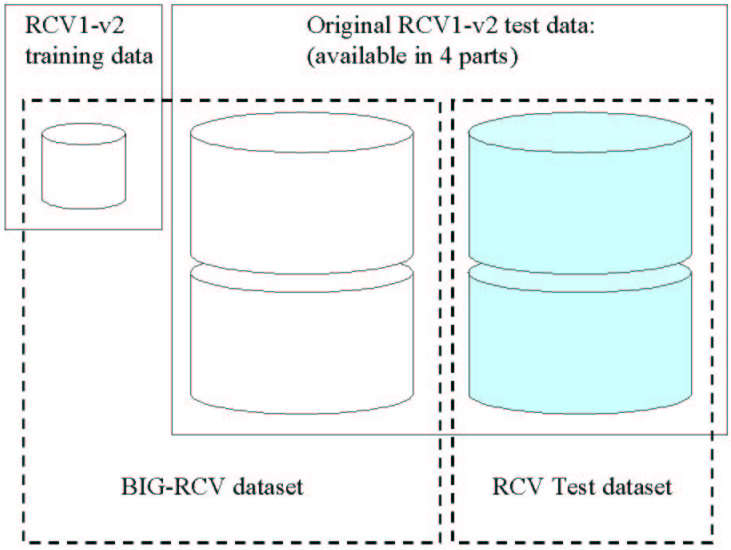


Figure 5: Schematic showing the construction of the various RCV1-v2 based data sets used in the experiments. The solid line bordered rectangles show the data as publicly available, the dashed-line bordered rectangles show the data sets we assembled. The shaded portion of the data is used only during testing.

functions and over a wide range of settings for the regularization parameter, γ. To show this, the tables report results for both too little regularization ($\gamma = 10$, probit link) and too much regularization ($\gamma = 100$, logistic link) for this particular data set. As a guide to assessing convergence in this and other tables that follow, we show the L_1 norm of the difference between the batch algorithm estimates (EM or BBR as appropriate) and the Online, MP or RMMP algorithm iterates (also as appropriate).

We next examine the first real data set, the training data for the ModApte split of Reuters-21578 (Lewis et al., 2004). This is a moderate dimensional ($d = 21989$ features) data set with $t = 9603$ labelled observations (we use the feature vectors that can be downloaded from the paper's appendix.). The features of this data set are weighted term occurrences and it is quite sparse, as is typical for text data. The batch EM algorithm for probit regression is prohibitively expensive on this data set as it involves inverting a high dimensional matrix, but we can run BBR to obtain batch logistic regression results. Hence we focus our results on logistic regression for this data set. We examine two reasonable settings for the regularization parameter, $\gamma = 10$ and $\gamma = 100$. For $\gamma = 10$, BBR returns 150 nonzero components and for $\gamma = 100$, the MAP β BBR returns has 31 non-zero components. Since the data set is sparse, and presents no memory limitations, we are able to apply the Online and MP algorithms in addition to the RMMP algorithm—see Tables 3 and 4.

For both amounts of regularization the Online parameter estimates aren't particularly good (although between the two settings, the parameter estimates with the higher amount of regularization are better). As discussed in Section 5, this is likely due to the relatively high dimensionality compared to the number of examples in the data set. The MP algorithm improves parameter estimates

Probit link function, $\gamma = 10$					Logistic link function, $\gamma = 100$				
EM	**Online**	**MP**			**BBR**	**Online**	**MP**		
		$z=1$	$z=2$	$z=3$			$z=1$	$z=2$	$z=3$
0.252	0.250	0.207	0.250	0.252	0.178	0.174	0.168	0.178	0.178
0.764	0.764	0.614	0.755	0.764	0.450	0.435	0.422	0.450	0.450
-0.318	-0.314	-0.263	-0.314	-0.318	-0.124	-0.120	-0.1161	-0.124	-0.124
0.834	0.821	0.667	0.824	0.834	0.713	0.689	0.666	0.712	0.713
0.894	0.880	0.719	0.884	0.894	0.656	0.634	0.613	0.655	0.656
-0.304	-0.297	-0.243	-0.301	-0.304	0	0	0	0	0
-0.782	-0.770	-0.627	-0.773	-0.782	-0.511	-0.493	-0.477	-0.510	-0.511
-0.039	-0.039	-0.037	-0.039	-0.039	0	0	0	0	0
-0.036	-0.036	-0.029	-0.036	-0.036	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-0.327	-0.322	-0.266	-0.324	-0.327	-0.030	-0.029	-0.028	-0.030	-0.030
L_1 Norm	0.074	0.878	0.050	0		0.0872	0.172	0.003	0

Table 2: Table with columns showing values of the MAP estimates of β obtained by the batch algorithms (EM on the left half, for probit regression and BBR on the right half for logistic regression), the Online algorithm and three successive iterates of the MP algorithm applied to the simulated data set. The final row displays the L_1 norm of the difference between the batch algorithm estimates (EM or BBR as appropriate) and the Online/MP algorithm estimates. The results shown here are representative of those obtained for other values of γ as well.

as expected. For $\gamma = 100$, the MP algorithm converges in about $z = 6$ iterations to parameter values indistinguishable from BBR—see the left three columns in Table 3. We next applied the RMMP algorithm to this data set. Examining the size of the first active set reveals setting $k \approx 3000$, would give exactly the same results as the MP algorithm—see typical effects of changing k in Table 4 for $\gamma = 10$. We point out that this is a huge reduction in the worst case memory required, an approximately 98% reduction ($k = 3000$ vs. $d = 21989$ originally). Note also that the size of k should be compared relative to the nonzero components for MAP β (150 and 31 for $\gamma = 10$ and $\gamma = 100$ respectively).

We further test the limits of the algorithm, by running it with $k = 300$ for $\gamma = 100$. The RMMP algorithm performs very well, requiring about $z = 7$ passes (only two more than the MP algorithm) to converge to correct parameter values. For $\gamma = 10$, where $k = 300$ is small (only twice the number of non-zero components in the MAP β), once again the same kind of results hold, with the MP algorithm needing about 7 passes over the data set and the RMMP algorithm needing about 15 passes to converge to the batch β.

Finally, we present results of application of the algorithms to the RCV1-v2 data sets. For the RCV1-v2 training data ($d = 47,236$, $t = 23,149$), sparsity again enables application of BBR to obtain the batch MAP β parameter values, as well as the Online and MP algorithms, although this is quite cumbersome. See Table 5. Again, as expected (examining d vs. t for this data set), the Online estimates are not very good. The multi-pass algorithms have improved parameter estimates. For $\gamma = 10$ (a fairly high amount of regularization), we find essentially the same qualitative results as the ModApte data set—it takes about $z = 6$ passes through the data set to obtain indistinguishable

j	**BBR**	**Online**	**MP**		**RMMP**, $k = 300$		
			$z = 3$	$z = 5$	$z = 3$	$z = 5$	$z = 7$
Intercept	-1.588	-1.404	-1.527	-1.586	-1.451	-1.573	-1.588
9 (bank)	1.188	0.697	0.957	1.185	0.688	1.143	1.188
13 (share)	0.847	0.609	0.793	0.846	0.678	0.839	0.847
147 (acquisit)	0.813	0.562	0.795	0.813	0.696	0.812	0.813
31 (offer)	0.801	0.337	0.618	0.800	0.356	0.772	0.801
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
3 (pct)	-2.264e-2	-2.259e-2	-2.127e-2	-2.240e-2	-3.247e-2	-2.062e-2	-2.264e-2
62 (plan)	-1.757e-2	-1.430e-2	-2.840e-2	-1.779e-2	-3.346e-2	-2.045e-2	-1.757e-2
2 (dlr)	1.552e-2	6.932e-3	1.542e-2	1.548e-2	1.610e-2	1.525e-2	1.552e-2
12 (net)	-1.467e-2	-6.671e-3	-1.956e-2	-1.480e-2	-1.415e-2	-1.643e-2	-1.467e-2
8 (ct)	1.277e-2	3.587e-2	2.870e-2	1.320e-2	2.915e-2	1.776e-2	1.278e-2
L_1 Norm		4.029	1.691	0.034	3.496	0.4027	3e-4

Table 3: Results obtained on the ModApte data set. The 5 highest and 5 lowest magnitude non-zero coefficients of MAP β for $\gamma = 100$ are shown. In table are the indices of β (and word stem features they correspond to in brackets), coefficients from BBR, and the Online algorithm, those obtained after a particular number of passes over the data using the MP algorithm (full memory) and parameters from the RMMP algorithm with $k = 300$.

j	**BBR**	**Online**	**RMMP**, $z = 8$				
			$k = 3120^*$	$k = 2000$	$k = 1000$	$k = 600$	$k = 300$
292 (banker)	2.695	1.523	2.695	2.695	2.695	2.695	2.699
20 (4)	2.268	0.617	2.268	2.260	2.260	2.259	2.273
Intercept	-2.010	-1.615	-2.010	-2.009	-2.009	-2.009	-2.005
341 (charg)	1.755	0.832	1.755	1.754	1.754	1.754	1.742
147 (acquisit)	1.572	0.862	1.572	1.572	1.572	1.572	1.568
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
66 (loan)	4.943e-3	9.106e-2	4.944e-3	4.849e-3	4.821e-3	4.849e-3	3.224e-3
134 (agre)	4.488e-3	4.836e-2	4.479e-3	4.720e-3	4.756e-3	4.712e-3	1.677e-2
267 (commerci)	-2.057e-3	0	-2.068e-3	-1.863e-3	-1.897e-3	-1.852e-3	-3.427e-3
28 (stock)	-1.652e-3	-3.879e-2	-1.644e-3	-1.542e-3	-1.560e-3	-1.537e-3	-1.991e-3
56 (interest)	-1.518e-4	-7.623e-2	-1.540e-4	-3.059e-4	-2.983e-4	-3.121e-4	-8.640e-4
L_1 Norm		28.290	1.4e-3	0.047	0.044	0.048	1.269

Table 4: Results for the ModApte data set: Illustrating the effect of changing k. The 5 highest and 5 lowest magnitude non-zero coefficients of MAP β for $\gamma = 10$ are shown. In table are the indices of β (and word stem features they correspond to in brackets), coefficients from BBR, the Online algorithm, and those obtained after 8 passes over the data using the RMMP algorithm. * For $k = 3120$, RMMP behaves the same as the MP algorithm.

$\gamma = 10$						$\gamma = 100$, $k = 2500$	
β index	**BBR**	**Online**	**RMMP,** $k = 1500$			β index	**RMMP**
			$z = 2$	$z = 5$	$z = 10$		$z = 10$
12220 (econom)	18.065	16.145	0	18.084	18.065	12220 (econom)	17.234
27407 (moody)	9.909	9.982	7.988	9.904	9.909	37665 (shar)	-11.901
37665 (shar)	-8.201	-3.918	-2.255	-8.118	-8.201	43626 (union)	8.654
46160 (work)	7.144	6.339	4.061	7.133	7.144	27407 (moody)	8.308
5946 (budget)	6.453	6.327	5.142	6.436	6.453	5946 (budget)	8.215
33192 (profit)	-6.211	-3.840	-2.066	-6.159	-6.211	19647 (inflat)	6.326
43626 (union)	6.164	5.789	4.430	6.157	6.164	39539 (statist)	5.782
21160 (july)	5.661	5.093	3.498	5.644	5.661	29641 (obligat)	4.728
19647 (inflat)	5.573	5.437	6.587	5.539	5.573	37471 (sery)	4.621
29641 (obligat)	5.472	6.250	4.810	5.473	5.472	41148 (tax)	4.507
L_1 Norm		24.798	87.940	0.480	0.001		

Table 5: RCV1-v2 results. Left portion RCV1-v2 training data set, right BIG-RCV data set.

parameter values as BBR (not shown in the table). The RMMP algorithm also gives excellent results in about 10 passes, see the left portion of Table 5 with $k = 1500$.

For the BIG-RCV data set ($d = 288,062$, $t = 421,816$) however, computational and memory limitations made it impossible to run the batch algorithms on this data set (also the Online and MP algorithm). It is precisely for cases like this that the RMMP algorithm is useful, and we were able to obtain parameter estimates for reasonable settings of regularization—see for example, the right portion of Table 5.

Does training on the entire BIG-RCV data set actually result in improved predictive performance? To address this, we conducted the following experiment. We obtained the best possible predictive parameters using 10-fold cross-validation on the RCV1-v2 training data set with a batch algorithm. This is an expensive computation, involving many repeated BBR runs for different values of the regularization parameter (we searched over $\gamma = 0.01, 0.1, 1, 10, 100$). The final cross-validation chosen β has 1010 non-zero parameters.

We then trained a separate sparse logistic classifier on the BIG-RCV data set using the RMMP algorithm with $k = 3000$ and $\gamma = 40$. Setting $\gamma = 40$ results in 1015 non-zero MAP β coefficients which is approximately the same number of non-zero coefficients as the cross-validation chosen β. Finally, we compare the predictive accuracy of both classifiers on the unused RCV test set (comprising the unused two portions of the original RCV1-v2 test data).

The results, shown in Table 6, demonstrate that using the information in extra examples, the "unsophisticated" classifier trained on the much larger data set outperforms the "optimized" classifier trained on a smaller data set.

8. Conclusions

In this paper we presented an asymptotically convergent online algorithm that builds sparse generalized linear models for massive data sets. We also presented efficient multi-pass algorithms that examine observations sequentially and thus enable learning on massive data sets. Both algorithms exploit sparsity of input data. We applied the algorithms to large, sparse data sets, for which state-

	"Optimized" β trained on RCV1-v2 training data		"Naive" β trained on BIG-RCV	
	Relevant	Not Relevant	Relevant	Not Relevant
Retrieved	38,821	7,415 (**83.96**%)	40,655	6,017 (**87.11**%)
Not Retrieved	16,368 (**70.34**%)	319,994	14,534 (**73.67**%)	321,392

Table 6: This table shows confusion matrices for prediction results on the RCV Test data set. The CV β (trained on the RCV1-v2 training data set) results are on the left and the MAP β (trained on the BIG-RCV data set, with $\gamma = 30$, $k = 3000$) results are on the right. Also shown are recall and precision percentages in bold and brackets. There are approximately 383,000 examples in the test data set.

of-the-art batch algorithms are impractical/cumbersome, and our results show that examining such data sets in their entirety can lead to better classifier performance.

Some areas of further research that this work opens up are: extension of the algorithms for a hierarchical prior model so that the choice of regularization is less important, the possible application of our methods to kernel classifiers, and applications to multi-class classification problems.

Acknowledgments

National Science Foundation grants IIS-9988642 and DMS-0505599 and the Multidisciplinary Research Program of the Department of Defense (MURI N00014-00-1-0637) supported this work. We are very grateful to David D. Lewis for detailed and insightful comments on an earlier draft of this paper.

Appendix A.

Here we show the Taylor expansions for the quadratic approximations to the log-likelihood function. To simplify notation, let $c(\beta) = \beta^T \mathbf{x}_i$ and $\hat{c} = \beta_{i-1}^T \mathbf{x}_i$. The link function (we will restrict analytical results to the logistic and probit link functions) is $\Phi(z)$ as before and we denote its first and second derivative, with respect to z, by $\Phi'(z)$ and $\Phi''(z)$ respectively.

Consider the case where $y_i = 1$:

$$\begin{aligned}
\log\Phi(c) &\approx \log\Phi(\hat{c}) + (c-\hat{c})\frac{\Phi'(\hat{c})}{\Phi(\hat{c})} + \frac{(c-\hat{c})^2}{2}\left(\frac{\Phi''(\hat{c})}{\Phi(\hat{c})} - \left(\frac{\Phi'(\hat{c})}{\Phi(\hat{c})}\right)^2\right) \\
&\propto \frac{\Phi'(\hat{c})}{\Phi(\hat{c})}c + \frac{1}{2}\left(\frac{\Phi''(\hat{c})}{\Phi(\hat{c})} - \left(\frac{\Phi'(\hat{c})}{\Phi(\hat{c})}\right)^2\right)c^2 - \hat{c}\left(\frac{\Phi''(\hat{c})}{\Phi(\hat{c})} - \left(\frac{\Phi'(\hat{c})}{\Phi(\hat{c})}\right)^2\right)c
\end{aligned}$$

so that:

$$a_i = \frac{1}{2}\left(\frac{\Phi''(\hat{c})}{\Phi(\hat{c})} - \left(\frac{\Phi'(\hat{c})}{\Phi(\hat{c})}\right)^2\right)$$

and

$$b_i = \frac{\Phi'(\hat{c})}{\Phi(\hat{c})} - \hat{c}\left(\frac{\Phi''(\hat{c})}{\Phi(\hat{c})} - \left(\frac{\Phi'(\hat{c})}{\Phi(\hat{c})}\right)^2\right).$$

Analogously, when $y_i = 0$:

$$\log(1-\Phi(c)) \approx \log(1-\Phi(\hat{c})) - (c-\hat{c})\frac{\Phi'(\hat{c})}{1-\Phi(\hat{c})} - \frac{(c-\hat{c})^2}{2}\left(\frac{\Phi''(\hat{c})}{1-\Phi(\hat{c})} + \left(\frac{\Phi'(\hat{c})}{1-\Phi(\hat{c})}\right)^2\right)$$

so that:

$$a_i = -\frac{1}{2}\left(\frac{\Phi''(\hat{c})}{1-\Phi(\hat{c})} + \left(\frac{\Phi'(\hat{c})}{1-\Phi(\hat{c})}\right)^2\right)$$

and

$$b_i = -\frac{\Phi'(\hat{c})}{1-\Phi(\hat{c})} + \hat{c}\left(\frac{\Phi''(\hat{c})}{1-\Phi(\hat{c})} + \left(\frac{\Phi'(\hat{c})}{1-\Phi(\hat{c})}\right)^2\right).$$

For the probit link function:

$$\begin{aligned}
\Phi(z) &= \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx \\
\Phi'(z) &= \frac{1}{\sqrt{2\pi}} e^{-z^2/2} \\
\Phi''(z) &= \frac{-z}{\sqrt{2\pi}} e^{-z^2/2},
\end{aligned}$$

whereas for the logistic link function:

$$\begin{aligned}
\Phi(z) &= \frac{e^z}{1+e^z} \\
\Phi'(z) &= \frac{e^z}{(1+e^z)^2} \\
\Phi''(z) &= \frac{(e^z)(1-e^z)}{(1+e^z)^3}.
\end{aligned}$$

These expressions then allow us to compute the a_i, b_i in the cases needed.

Appendix B.

In this appendix we derive the modified Shooting algorithm, Algorithm 1 and discuss its efficient implementation. We derive Shooting by analyzing the subdifferential of the system (Rockafellar, 1970). We need convex non-smooth analysis results because the regularization term is non-differentiable at zero. Reviewing concepts very briefly, the subgradient $\xi \in \mathbb{R}^{|x|}$, of a convex function f at x_0 is defined to be any vector satisfying:

$$f(x) \geq f(x_0) + \xi^T (x - x_0).$$

In words, any vector ξ, such that a plane through $(x, f(x))$ with slope ξ contains f in its upper half-space qualifies as a subgradient (equivalently, a tangent plane supporting the convex function

f). The subdifferential, ∂f, is just the set of all subgradients, ξ, at a particular point. This is a generalization of the gradient which collapses to the gradient, whenever f is differentiable. As a simple example, the subdifferential of $f(\beta) = |\beta|$, the absolute value function (which is non-differentiable at $\beta = 0$) is:

$$\partial f = \begin{cases} \{-1\}, & \beta < 0 \\ [-1,1], & \beta = 0 \\ \{1\}, & \beta > 0. \end{cases}$$

As one expects, analogous to optimality conditions resulting from setting the gradient of a differentiable function to zero, optimality conditions for non-differentiable functions result from restrictions on the subdifferential. In particular we appeal to the following result from non-smooth analysis (Rockafellar, 1970):

Theorem $\hat{\beta}$ *is a global minimizer of a convex function* $f(\beta)$ *if and only if* $0 \in \partial f(\hat{\beta})$.

Now to our particular problem. We need to find β that is a solution to:

$$\max_{\beta} \left(\beta^T \Psi \beta + \beta^T \theta - \gamma \|\beta\|_1 \right).$$

The convexity of the problem allows us to make incremental progress towards the maxima coordinate-wise. Starting from some parameter vector, we compute the jth component of the subdifferential of the function (keeping all other components fixed):

$$\begin{aligned} & \tfrac{\partial}{\partial \beta_j}(\beta^T \Psi \beta) + \tfrac{\partial}{\partial \beta_j}(\beta^T \theta) - \gamma \partial(\textstyle\sum_{j=1}^{d}(|\beta_j|) \\ = & \quad 2(\Psi\beta)_j + \theta_j - \gamma\partial(|\beta_j|) \\ = & \quad 2\Psi_{jj}\beta_j + 2(\Psi'\beta)_j + \theta_j - \gamma\partial(|\beta_j|) \end{aligned}$$

where $(\Psi'\beta)_j$ is the j'th component of the vector $\Psi'\beta$ and Ψ_{jj} refers to the (j,j)'th element of the matrix Ψ (Recall that Ψ' is defined to be the matrix Ψ with diagonal entries set to zero). The second equation follows from the first as the subdifferential of a univariate differentiable function is just its derivative and since matrix Ψ is symmetric (it is just a weighted sum of outer products). Now if we plug in the subdifferential of the non-differentiable absolute value function, and set $\Omega_j = 2(\Psi'\beta)_j + \theta_j$ (and thus define the vector Ω to be the gradient of the purely differentiable part of the objective function), we obtain the subdifferential of the objective function, whose j'th component we denote by ∂_{β_j} as:

$$\partial_{\beta_j} = \begin{cases} \{2\Psi_{jj}\beta_j + \Omega_j + \gamma\}, & \beta_j < 0 \\ [\Omega_j - \gamma, \Omega_j + \gamma], & \beta_j = 0 \\ \{2\Psi_{jj}\beta_j + \Omega_j - \gamma\}, & \beta_j > 0. \end{cases}$$

This is a piecewise linear function with fixed negative slope $2\Psi_{jj}$ and a constant jump of fixed size 2γ at $\beta_j = 0$ (Ψ_{jj} can be proven to always be negative by looking at the update formula for Ψ and using the fact that $\forall i, a_i < 0$). Using the optimality criteria (now for maximization since $-|\beta_j|$ is a concave function) naturally leads to the modified Shooting algorithm, illustrated in Figure 6.

Now to questions regarding the efficient implementation of the Shooting algorithm, used by the online, MP and RMMP algorithms. In the modified Shooting algorithm, after each component update (change in β_j) we need to modify Ω (the update Ω step in the algorithm). This can be implemented efficiently using the following result (similar to the trick detailed in Minka, 2001):

$$\Omega^{new} = \Omega^{old} + 2\Psi'_{(\cdot j)}(\Delta\beta_j)$$

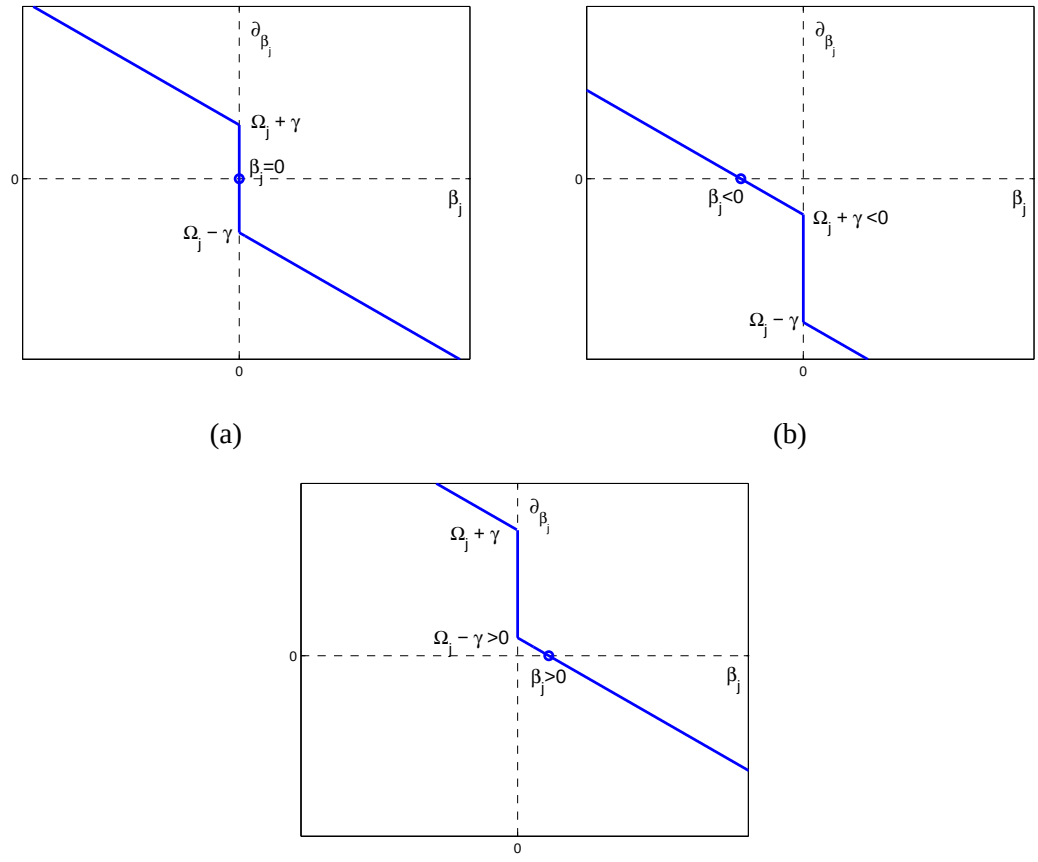


Figure 6: Illustration of cases occurring in the Shooting algorithm (a) If $|\Omega_j| \leq \gamma$ the constant portion of the subdifferential contains zero. In this case, set $\beta_j = 0$ (b) If instead, $\Omega_j < -\gamma$, the optimality conditions will be satisfied by setting $\beta_j = \frac{-\gamma - \Omega_j}{2\Psi_{jj}}$ (c) The case analogous to (b) but when $\Omega_j > \gamma$. Here the subdifferential is set equal to zero when $\beta_j = \frac{\gamma - \Omega_j}{2\Psi_{jj}}$.

where $\Delta\beta_j$ is the change in β_j and $\Psi'_{(.j)}$ is the j'th column of Ψ'. Thus each component update of Shooting can be done in $O(d)$ computational time. Now, if as before the maximum number of non-zero components of β along the solution path to MAP β is m, only m such updates will need to be made, giving a total time requirement per iteration of $O(md)$.

Finally we detail how to carry out the Ω updates efficiently for the RMMP algorithm, Algorithm 3. Recall that since we are discussing a multi-pass algorithm, the location where we take the quadratic approximation, β_{i-1}, is constant throughout the pass through the fixed data set, D_t. We exploit this fact to show that in this case, you don't explicitly need the matrix Ψ (or $\tilde{\Psi}$) to determine Ω. Indeed, after going through all the observations in the data set (pass z, say):

$$\Omega = 2\Psi'\beta_{z-1} + \theta = 2\left(\sum_{i=1}^{t} a_i\left(\mathbf{x}_i\mathbf{x}_i^T - \mathrm{diag}(\mathbf{x}_i^2)\right)\right)\beta_{z-1} + \sum_{i=1}^{t} b_i\mathbf{x}_i,$$

which follows from the definitions of Ω, θ and Ψ'. In the above equation, $\mathrm{diag}(\mathbf{x}_i^2)$ is a $d \times d$ matrix zero everywhere except the diagonal entries, which consists of the elements of the vector $\mathbf{x}_i$ squared component-wise. This leads to the following equation for Ω:

$$\Omega = 2\sum_{i=1}^{t} a_i(\beta_{z-1}^T\mathbf{x}_i)\mathbf{x}_i - 2\sum_{i=1}^{t} a_i(\mathbf{x}_i^2\beta_{z-1}) + \sum_{i=1}^{t} b_i\mathbf{x}_i,$$

where $(\mathbf{x}_i^2\beta_{z-1})$ is a vector whose entries are $\mathbf{x}_i^2$ multiplied by β_{z-1} component-wise. Note the first sum is just a weighted combination of the input data ($\beta_{z-1}^T\mathbf{x}_i$ is a scalar). Thus, our final update formula results:

$$\Omega^{new} = \Omega^{old} + (2a_i\beta_{z-1}^T\mathbf{x}_i + b_i)\mathbf{x}_i - 2a_i(\mathbf{x}_i^2\beta_{z-1}).$$

As can be seen, computing this update per observation takes time and space $O(d)$, and having restricted the number of non-zero components of β to k, a total computational cost per iteration of Shooting to $O(kd)$.

Appendix C.

We present a proof sketch for the convergence behavior of the online algorithm in the infinite data limit. The intuition for is as follows: as $t \to \infty$, the Bayesian central limit theorems dictate that the posterior distribution tends (in distribution) to a multivariate Gaussian with ever shrinking covariance, (Bernardo and Smith, 1994). Thus, less and less information is required to encode the posterior distribution as more and more data is added—to a point. Indeed, in the limit, only the vector of the maximum likelihood value of the parameters, β_{MLE}, is required to completely describe the posterior distribution.

Suppose now that the online algorithm converges to a particular fixed point. In the infinite data limit, an infinite number of term approximations are taken at this fixed point. Now, our Taylor polynomial based approximation preserves both the function value and its gradient, and an infinite number of approximations are jointly maximum at this fixed point. This implies the fixed point is an optima of the posterior distribution.

Thus, if the approximation converges to a fixed point, it is the correct optima location. The above is a modification of the fixed point Lemma in the paper on Laplace Propagation (Eskin et al., 2003). One can also prove unbiasedness which follows from our update rules and a minor modification of

a theorem in Opper (1998). Even though Opper derives his results based on a Gaussian prior on the parameters β (corresponding to L_2 regularization), the general format of Opper's theorem is still applicable in our case because, in the infinite data limit, the prior is inconsequential.

References

J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. John Wiley and Sons, Inc., 1994.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by Basis Pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, January 1999.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

E. Eskin, A. J. Smola, and S.V.N. Vishwanathan. Laplace Propagation. In *Neural Information Processing Systems*, 16. MIT Press, 2003.

M. A. T. Figueiredo and A. K. Jain. Bayesian learning of sparse classifiers. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, volume 1, pages 35–41, 2001.

W. J. Fu. Penalized regressions: The Bridge versus the Lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998.

A. Genkin, D. D. Lewis, and D. Madigan. Large-scale Bayesian logisitic regression for text categorization. *Technometrics*, 49:291–304, 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.

T. Jaakkola and M. Jordan. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10:25–37, 2000.

R. E. Kass and A. E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90: 773–795, 1995.

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.

P. Komarek and A. Moore. Making logistic regression a core data mining tool: A practical investigation of accuracy, speed, and simplicity. Technical Report TR-05-27, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2005.

B. Krishnapuram, L. Carin, M. A. T. Figueiredo, and A. J. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, 2005.

D. D. Lewis. Reuters-21578 text categorization test collection: Distribution 1.0 readme file (v 1.3)., 2004. URL `http://www.daviddlewis.com/resources/testcollections/reuters21578/readme.txt`.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

D. J. C. MacKay. Probable networks and plausible predictions: a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6:469–505, 1995.

T. P. Minka. Expectation Propagation for approximate Bayesian inference. In Jack Breese and Daphne Koller, editors, *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 362–369, San Francisco, CA, August 2–5 2001a. Morgan Kaufmann Publishers.

T. P. Minka. *A Family of Algorithms for Approximate Bayesian Inference.* PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2001b.

M. Opper. A Bayesian approach to on-line learning. In D. Saad, editor, *Online Learning in Neural Networks*, pages 363–378. Cambridge University Press, 1998.

M. R. Osborne, B. Presnell, and B. A. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403, July 2000.

Y. Qi, T. P. Minka, R. W. Picard, and Z. Ghahramani. Predictive automatic relevance determination by Expectation Propagation. In *Proceedings of Twenty-first International Conference on Machine Learning*, Banff, Alberta, Canada, July 4-8 2004.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, N.J, 1970.

S. K. Shevade and S. S. Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics.*, 19(17):2246–2253, 2003.

R. J. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

T. Zhang. On the dual formulation of regularized linear systems. *Machine Learning*, 46:91–129, 2002.

T. Zhang and F. J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31, 2001.

H. Zou. The adaptive Lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429, 2006.

Journal of Machine Learning Research 9 (2008) 339-370 Submitted 11/06; Revised 9/07; Published 3/08

Generalization from Observed to Unobserved Features by Clustering

Eyal Krupka EYAL.KRUPKA@MAIL.HUJI.AC.IL
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: Sanjoy Dasgupta

Abstract

We argue that when objects are characterized by many attributes, clustering them on the basis of a *random* subset of these attributes can capture information on the unobserved attributes as well. Moreover, we show that under mild technical conditions, clustering the objects on the basis of such a random subset performs almost as well as clustering with the full attribute set. We prove finite sample generalization theorems for this novel learning scheme that extends analogous results from the supervised learning setting. We use our framework to analyze generalization to unobserved features of two well-known clustering algorithms: *k*-means and the maximum likelihood multinomial mixture model. The scheme is demonstrated for collaborative filtering of users with movie ratings as attributes and document clustering with words as attributes.

Keywords: clustering, unobserved features, learning theory, generalization in clustering, information bottleneck

1. Introduction

Data clustering can be defined as unsupervised classification of objects into groups based on their similarity (see, for example, Jain et al., 1999). Often, it is desirable to have the clusters match some labels that are unknown to the clustering algorithm. In this context, good data clustering is expected to have homogeneous labels in each cluster, under some constraints on the number or complexity of the clusters. This can be quantified by mutual information (see, for example, Cover and Thomas, 1991) between the objects' cluster identity and their (unknown) labels, for a given complexity of clusters. However, since the clustering algorithm has no access to the labels, it is unclear how it can optimize the quality of the clustering. Even worse, the clustering quality depends on the specific choice of the unobserved labels. For example, good document clustering with respect to topics is very different from clustering with respect to authors.

In our setting, instead of attempting to cluster by some arbitrary labels, we try to predict unobserved features from observed ones. In this sense our target labels are simply other features that happened to be unobserved. For example, when clustering fruits based on their observed features such as shape, color and size, the target of clustering is to match unobserved features such as nutritional value or toxicity. When clustering users based on their movie ratings, the target of clustering is to match ratings of movies that were not rated, or not even created as yet.

In order to theoretically analyze and quantify this new learning scheme, we make the following assumptions. Consider a very large set of features, and assume that we observe only a *random*

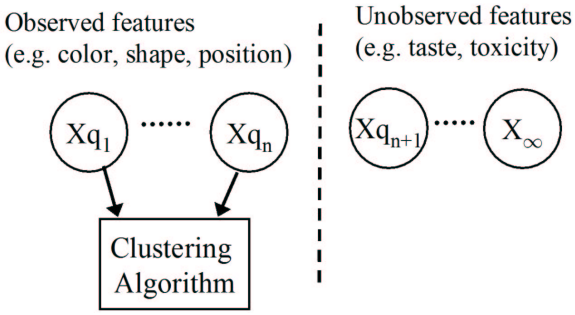


Figure 1: The learning scheme. The clustering algorithm has access to a random subset of features $(X_{q_1},...,X_{q_n})$ of m instances. The goal of the clustering algorithm is to assign a class label t_i to each instance, such that the expected mutual information between the class labels and a randomly selected *unobserved* feature is maximized.

subset of n features, called *observed features*. The other features are called *unobserved features*. We assume that the random selection of observed features is made from some unknown distribution $\mathcal{D}$ and each feature is selected independently.[1]

The clustering algorithm has access only to the observed features of m instances. After clustering, one of the *unobserved* features is randomly selected to be the target label. This selection is done using the same distribution, $\mathcal{D}$, of the observed feature selection. Clustering performance is measured with respect to this feature. Obviously, the clustering algorithm cannot be directly optimized for this specific feature.

The question is whether we can optimize the *expected* performance on the unobserved features, based only on the observed features. The expectation is over the *random* selection of the unobserved target features. In other words, can we find the clustering that is most likely to match a randomly selected unobserved feature? Perhaps surprisingly, for a large enough number of observed features, the answer is yes. We show that for any clustering algorithm, the average performance of the clustering with respect to the observed and unobserved features is similar. Hence we can indirectly optimize clustering performance with respect to unobserved features by analogy with generalization in supervised learning. These results are universal and do not require any additional assumptions such as an underlying model or a distribution that created the instances.

In order to quantify these results, we define two terms: the average observed information and the expected unobserved information. Let T be the variable which represents the cluster for each instance, and $\{X_1,...,X_L\}$ $(L \to \infty)$ the set of discrete random variables which denotes the features. The average observed information, denoted by I_{ob}, is the average mutual information between T and each of the observed features. In other words, if the observed features are $\{X_1,...,X_n\}$ then $I_{ob} = \frac{1}{n}\sum_{j=1}^{n} I(T;X_j)$. The expected unobserved information, denoted by I_{un}, is the *expected* value of the mutual information between T and a new *randomly* selected feature, that is, $E_{q\sim\mathcal{D}}\left\{I(T;X_q)\right\}$. We are interested in cases where this new selected feature is most likely to be one of the unobserved features, and therefore we use the term unobserved information. Note that whereas I_{ob} can be measured directly, this paper deals with the question of how to infer and maximize I_{un}.

1. For simplicity, we also assume that the probability of selecting the same feature more than once is near zero.

Our main results consist of two theorems. The first is a generalization theorem. It gives an upper bound on the probability of a large difference between I_{ob} and I_{un} for all possible partitions. It also states a *uniform convergence in probability* of $|I_{ob} - I_{un}|$ as the number of observed features increases. Conceptually, the average observed information, I_{ob}, is analogous to the training error in standard supervised learning (Vapnik, 1998), whereas the unobserved information, I_{un}, is similar to the generalization error.

The second theorem states that under constraints on the number of clusters, and a large enough number of observed features, one can achieve nearly the best possible performance, in terms of I_{un}. Analogous to the principle of Empirical Risk Minimization (ERM) in statistical learning theory (Vapnik, 1998), this is done by maximizing I_{ob}.

We use our framework to analyze clustering by the maximum likelihood of multinomial mixture model (also called Naive Bayes Mixture Model, see Figure 2 and Section 2.2). This clustering assumes a generative model of the data, where the *instances* are assumed to be sampled independently from a mixture of distributions, and for each such distribution all features are independent. These assumptions are quite different from our assumptions of fixed instances and randomly observed features.[2] Nevertheless, in Section 2.2 we show that this clustering achieves nearly the best possible clustering in terms of information on unobserved features.

In Section 3 we extend our framework to distance-based clustering. In this case the measure of the quality of clustering is based on some distance function instead of mutual information. We show that the *k*-means clustering algorithm (Lloyd, 1957; MacQueen, 1967) not only minimizes the observed intra-cluster variance, but also minimizes the unobserved intra-cluster variance, that is, the variance of unobserved features within each cluster.

Table 1 summarizes the similarities and differences of our setting to that of supervised learning. The key difference is that in supervised learning, the set of features is fixed and the training instances are assumed to be randomly drawn from some distribution. Hence, the generalization is to new instances. In our setting, the set of instances is fixed, but the set of observed features is assumed to be randomly selected. Hence, the generalization is to new features.

Our new theorems are evaluated empirically in Section 4, on two different data sets. The first is a movie ratings data set, where we cluster users based on their movie ratings. The second is a document data set, with words as features. Our main point in this paper, however, is the new conceptual framework and not a specific algorithm or experimental performance.

Section 5 discusses related work and Section 6 presents conclusions and ideas for future research. A notation table is available in Appendix B.

2. Feature Generalization of Information

In this section we analyze feature generalization in terms of mutual information between the clusters and the features. Consider a fixed set of m instances denoted by $\{\mathbf{x}[1],\ldots,\mathbf{x}[m]\}$. Each instance is represented by a vector of L features $\{x_1,\ldots,x_L\}$. The value of the qth feature of the jth instance is denoted by $x_q[j]$. Out of this set of features, n features are randomly and independently selected according to some distribution $\mathcal{D}$. The n randomly selected features are the *observed features* (variables) and their indices are denoted by $\tilde{\mathbf{q}} = (q_1,...,q_n)$, where $q_i \sim \mathcal{D}$. The ith observed feature of the jth instance is denoted by $x_{q_i}[j]$. After selecting the observed features, we also select *unobserved*

2. Note that in our framework, random independence refers to the *selection* of observed features, not to the feature values.

		Prediction of unobserved features	
	Supervised learning	Information	Distance-based
Training set	Randomly selected *instances*	*n* randomly selected features (observed features)	
Test set	Randomly selected *unlabeled instances*	Randomly selected *unobserved features*	
Hypothesis class	Class of functions from instances to labels	All possible partitions of *m* instances into *k* clusters	
Output of learning algorithm	Select hypothesis function	Cluster the *instances* into *k* clusters	
Goal	Minimize *expected* error on test set	Maximize *expected* information on *unobserved* features	Minimize *expected* intra-cluster variance of *unobserved* features
Assumption	Training and test instances are randomly and independently drawn from the same distribution	Observed and unobserved features are randomly and independently selected using the same distribution	
Strategy	Empirical Risk Minimization (ERM)	Observed Information Maximization (OIM)	Minimize observed intra-cluster variance
Related clustering algorithm		Maximum likelihood multinomial mixture model (Figure 2)	*k*-means
Good generalization	The training and test errors are similar	The observed and unobserved information are similar	The observed and unobserved intra-cluster variance are similar

Table 1: Analogy with supervised learning

features according to the same distribution $\mathcal{D}$. For simplicity, we assume that the total number of features, L, is large and the probability of selecting the same feature more than once is near zero (as in the case where $L \gg n^2$, where $\mathcal{D}$ is uniform distribution). This means that we can assume that a randomly selected unobserved feature is not one of the n observed features. It is important to emphasize that we have a fixed and finite set of instances; that is, we do not need to assume that the m instances were drawn from any distribution. Only the features are randomly selected.

We further assume that each of the features is a discrete variable with no more than s different values.[3] The clustering algorithm clusters the instances into k clusters. The clustering is denoted by the function $\mathbf{t} : [m] \to [k]$ that maps each of the m instances to one of the k clusters. The cluster label of the jth instance is denoted by $\mathbf{t}(j)$. Our measures for the quality of clustering are based on Shannon's mutual information. Let random variable Z denote a number chosen uniformly at random from $\{1,\ldots,m\}$. We define the quality of clustering with respect to a single feature, q, as $I(\mathbf{t}(Z); x_q[Z])$, that is, the empirical mutual information between the cluster labels and the feature.

Our measure of performance assumes that the number of clusters is predetermined. There is an obvious tradeoff between the preserved mutual information and the number of clusters. For example, one could put each instance in a different cluster, and thus get the maximum possible mutual information for all features. Obviously all clusters will be homogeneous with respect to all features but this clustering is pointless. Therefore, we need to have some constraints on the number of clusters.

Definition 1 *The **average observed information** of a clustering* $\mathbf{t}$ *and the observed features is denoted by* $I_{ob}(\mathbf{t}, \tilde{\mathbf{q}})$ *and defined by*

$$I_{ob}(\mathbf{t}, \tilde{\mathbf{q}}) = \frac{1}{n} \sum_{i=1}^{n} I(\mathbf{t}(Z); x_{q_i}[Z]).$$

*The **expected unobserved information** of a clustering is denoted by* $I_{un}(t)$ *and defined by*

$$I_{un}(\mathbf{t}) = \mathbf{E}_{q \sim \mathcal{D}} \left\{ I(\mathbf{t}(Z); x_q[Z]) \right\}.$$

In general, I_{ob} is higher when clusters are more coherent; that is, elements within each cluster have many identical observed features. I_{un} is high if there is a high probability that the clusters are informative on a randomly selected feature q (where $q \sim \mathcal{D}$). In the special case where the distribution $\mathcal{D}$ is uniform and $L \gg n^2$, I_{un} can also be written as the average mutual information between the cluster label and the unobserved features set; that is, $I_{un} \approx \frac{1}{L-n} \sum_{q \notin \{q_1,\ldots,q_n\}} I(\mathbf{t}(Z); x_q[Z])$. Recall that L is the total number of features, both observed and unobserved.

The goal of the clustering algorithm is to cluster the instances into k clusters that maximize the *unobserved* information, I_{un}. Before discussing how to maximize I_{un}, we first consider the problem of estimating it. Similar to the generalization error in supervised learning, I_{un} cannot be calculated directly in the learning algorithm, but we may be able to bound the difference between the observed information I_{ob}—our "training error"—and the unobserved information I_{un}—our "generalization error". To obtain generalization, this bound should be *uniform over all possible clusterings* with a high probability over the randomly selected features. The following lemma argues that *uniform convergence in probability* of I_{ob} to I_{un} always occurs.

3. Since we are exploring an empirical distribution of a finite set of instances, dealing with continuous features is not meaningful.

Lemma 2 *With the definitions above,*

$$\Pr_{\tilde{\mathbf{q}}=(q_1,\ldots,q_n)}\left\{\sup_{\mathbf{t}:[m]\to[k]}|I_{ob}(\mathbf{t},\tilde{\mathbf{q}})-I_{un}(\mathbf{t})|>\varepsilon\right\}\leq 2e^{-2n\varepsilon^2/(\log k)^2+m\log k},\quad\forall\varepsilon>0.$$

Proof For any q,

$$0\leq I(\mathbf{t}(Z);x_q[Z])\leq H(\mathbf{t}(Z))\leq\log k.$$

Using Hoeffding's inequality, for any specific (predetermined) clustering

$$\Pr_{\tilde{\mathbf{q}}=(q_1,\ldots,q_n)}\{|I_{ob}(\mathbf{t},\tilde{\mathbf{q}})-I_{un}(\mathbf{t})|>\varepsilon\}\leq 2e^{-2n\varepsilon^2/(\log k)^2}.$$

Since there are at most k^m possible partitions, the union bound is sufficient to prove Lemma 2.■

Note that for any $\varepsilon>0$, the probability that $|I_{ob}-I_{un}|>\varepsilon$ goes to zero, as $n\to\infty$. The convergence rate of I_{ob} to I_{un} is bounded by $O((\log k)/\sqrt{n})$. As expected, this upper bound decreases as the number of clusters, k, decreases.

Unlike the standard bounds in supervised learning, this bound increases with the number of instances (m), and decreases with increasing numbers of observed features (n). This is because in our scheme the training size is not the number of instances, but rather the number of observed features (see Table 1). However, in the next theorem we obtain an upper bound that is independent of m, and hence is tighter for large m.

Consider the case where n is fixed, and m increases infinitely. We can select a random subset of instances of size m'. For large enough m', the empirical distribution of this subset is similar to the distribution over all instances. By fixing m', we can get a bound which is independent of m. Using this observation, the next theorem gives a bound that is independent of m.

Theorem 3 ***(Information Generalization)*** *With the definitions above,*

$$\Pr_{\tilde{\mathbf{q}}=(q_1,\ldots,q_n)}\left\{\sup_{\mathbf{t}:[m]\to[k]}|I_{ob}(\mathbf{t},\tilde{\mathbf{q}})-I_{un}(\mathbf{t})|>\varepsilon\right\}\leq 8(\log k)e^{-n\varepsilon^2/\left(8(\log k)^2\right)+4sk\log k/\varepsilon-\log\varepsilon},\quad\forall\varepsilon>0.$$

The proof of this theorem is given in appendix A.1. In this theorem, the bound does not depend on the number of instances, but rather on s which is the maximum alphabet size of the features. The convergence rate here is bounded by $O((\log k)/\sqrt[3]{n})$. However, for relatively large n one can use the bound in Lemma 2, which converges faster.

As shown in Table 1, Theorem 3 is clearly analogous to the standard uniform convergence results in supervised learning theory (see, for example, Vapnik, 1998), where the random sample is replaced by our randomly selected features, the hypotheses are replaced by the clustering, and I_{ob} and I_{un} replace the empirical and expected risks, respectively. The complexity of the clustering (our hypothesis class) is controlled by the number of clusters, k.

We can now return to the problem of specifying a clustering that maximizes I_{un}, using only the observed features. For reference, we will first define I_{un} of the best possible clustering.

Definition 4 ***Maximally achievable unobserved information:*** *Let $I^*_{un,k}$ be the maximum value of I_{un} that can be achieved by any partition into k clusters,*

$$I^*_{un,k} = \sup_{\mathbf{t}:[m]\to[k]} I_{un}(\mathbf{t}).$$

The clustering that achieves this value is called ***the best clustering****. The average observed information of this clustering is denoted by $I^*_{ob,k}$.*

Definition 5 ***Observed information maximization algorithm:*** *Let* IobMax *be any clustering algorithm that, based on the values of the observed features, selects a clustering $\mathbf{t}^{opt,ob} : [m] \to [k]$ having the maximum possible value of I_{ob}, that is,*

$$\mathbf{t}^{opt,ob} = \arg\max_{\mathbf{t}:[m]\to[k]} I_{ob}(\mathbf{t}, \tilde{\mathbf{q}}).$$

Let $\tilde{I}_{ob,k}$ be the average observed information of this clustering and $\tilde{I}_{un,k}$ be the expected unobserved information of this clustering, that is,

$$\begin{aligned} \tilde{I}_{ob,k}(\tilde{\mathbf{q}}) &= I_{ob}\left(\mathbf{t}^{opt,ob}, \tilde{\mathbf{q}}\right), \\ \tilde{I}_{un,k}(\tilde{\mathbf{q}}) &= I_{un}\left(\mathbf{t}^{opt,ob}\right). \end{aligned}$$

The next theorem states that *IobMax* not only maximizes I_{ob}, but also maximizes I_{un}.

Theorem 6 ***(Achievability)*** *With the definitions above,*

$$\Pr_{\tilde{\mathbf{q}}=(q_1,\ldots,q_n)}\left\{\tilde{I}_{un,k}(\tilde{\mathbf{q}}) \le I^*_{un,k} - \varepsilon\right\} \le 8(\log k) e^{-n\varepsilon^2/\left(32(\log k)^2\right)+8sk\log k/\varepsilon-\log(\varepsilon/2)}, \quad \forall \varepsilon > 0. \qquad (1)$$

Proof We now define a *bad clustering* as a clustering whose expected unobserved information satisfies $I_{un} \le I^*_{un,k} - \varepsilon$. Using Theorem 3, the probability that $|I_{ob} - I_{un}| > \varepsilon/2$ for any of the clusterings is upper bounded by the right term of Equation 1. If for all clusterings $|I_{ob} - I_{un}| \le \varepsilon/2$, then surely $I^*_{ob,k} \ge I^*_{un,k} - \varepsilon/2$ (see Definition 4) and I_{ob} of all bad clusterings satisfies $I_{ob} \le I^*_{un,k} - \varepsilon/2$. Hence the probability that a bad clustering has a higher average observed information than the best clustering is upper bounded as in Theorem 6. ■

For small m, a tighter bound, similar to that of Lemma 2 can easily be formulated.

As a result of this theorem, when n is large enough, even an algorithm that knows the value of *all* features (observed and unobserved) cannot find a clustering which is significantly better than the clustering found by the *IobMax* algorithm. This is demonstrated empirically in Section 4.

Informally, this theorem means that for a large number of features we can find a clustering that is informative on unobserved features. For example, clustering users based on similar ratings of current movies are likely to match future movies as well (see Section 4).

In the generalization and achievability theorems (Theorems 3, 6) we assumed that we are dealing only with hard clustering. In Appendix A.2 we show that the generalization theorem is also applicable to soft clustering; that is, assigning a probability distribution among the clusters to each instance. Moreover, we show that soft clustering is not required to maximize I_{ob}, since its maximum value can be achieved by hard clustering.

2.1 Toy Examples

In the first two examples below, we assume that the instances are drawn from a given distribution (although this assumption is not necessary for the theorems above). We also assume that the number of instances is large, so the empirical and the actual distributions of the instances are about the same.

Example 1 *Let $X_1,...,X_\infty$ be Bernoulli($\frac{1}{2}$) random variables, such that all variables with an even index are equal to each other ($x_2 = x_4 = x_6 = ...$), and all variables with an odd index are independent of each other and of all other variables. If the number of randomly observed features is large enough we can find a clustering rule with two clusters such that $I_{ob} \cong \frac{1}{2}$. This is done by assigning the cluster labels based on the set of features that are correlated, for example, $\mathbf{t}(i) = x_2[i] + 1 \quad \forall i$, assuming that x_2 is one of the observed features. $I(\mathbf{t}(Z); x_i(Z))$ is one for even i, and zero for odd i. For large n, the number of randomly selected features with odd indices and even indices[4] is about the same (with high probability), and hence $I_{ob} \cong \frac{1}{2}$. For this clustering rule $I_{un} \cong \frac{1}{2}$, since half of the unobserved features match this clustering (all features with an even index).*

Example 2 *When $X_1,...,X_\infty$ are i.i.d. (independent and identically distributed) Bernoulli($\frac{1}{2}$) random variables, $I_{un} = 0$ for any clustering rule, regardless of the number of observed features. For a finite number of clusters, I_{ob} will necessarily approach zero as the number of observed features increases. More specifically, if we use two clusters, where the clustering is determined by one of the observed features (i.e., $\mathbf{t}(i) = x_j(i)$, where x_j is an observed feature), then $I_{ob} = \frac{1}{n}$ (because $I(\mathbf{t}(Z); x_j(Z)) = 1$ and $I(\mathbf{t}(Z); x_l(Z)) = 0$ for $l \neq j$).*

Example 3 *Clustering fruits based on the observed features (color, size, shape etc.) also matches many unobserved features. Indeed, people clustered fruits into oranges, bananas and others (by giving names in the language) long before vitamin C was discovered. Nevertheless, this clustering was very informative about the amount of vitamin C in fruits, that is, most oranges have similar amounts of vitamin C, which is different from the amount in bananas.*

Based on the generalization theorem, we now suggest a qualitative explanation of why clustering into bananas and oranges provides relatively high information on unobserved features, while clustering based on position (e.g., right/left in the field of view) does not. Clustering into bananas and oranges contains information on many observed features (size, shape, color, texture), and thus has relatively large I_{ob}. By the generalization theorem, this implies that it also has high I_{un}. By contrast, a clustering rule which puts all items that appeared in our right visual field in one cluster, and the others in a second cluster, has much smaller I_{ob} (since it does not match many observed features), and indeed it is not predictive about unobserved features.

Example 4 *As a negative example, if the type of observed features and the target unobserved features are very different, our assumptions do not hold. For example, when the observations are pixels of an image, and the target variable is the label of the image, we cannot generalize from information about the pixels to information about the label.*

2.2 Feature Generalization of Maximum Likelihood Multinomial Mixture Models

In the framework of Bayesian graphical models, the multinomial mixture model is commonly used. The assumption of this model is that all features are conditionally independent, given the value of

4. Note that the indices are arbitrary. The learning algorithm does not use the indices of the features.

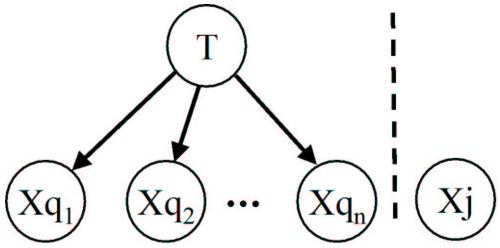


Figure 2: The Bayesian network (Pearl, 1988) of the multinomial mixture model. The observed random variables $\{X_{q_1},\ldots,X_{q_n}\}$ are statistically independent given the parent hidden random variable, T. This parent variable represents the cluster label. Although the basic assumptions of the multinomial mixture model are very different from ours, Theorem 7 tells us that this method of clustering generalizes well to unobserved features.

some hidden variable, that is,

$$\Pr(T=t,x_{q_1},\ldots,x_{q_n}) = \Pr(T=t)\prod_{r=1}^{n}\Pr(x_{q_r}|T=t),$$

where T denotes the hidden variable. The Bayesian network (Pearl, 1988) of this model is given in Figure 2. This standard model does not assume the existence of unobserved features, so we use the notation $x_{q_1},\ldots,x_{q_n}$ to denote the observed features which are used by the model. The set of instances are assumed to be drawn from such a distribution, with unknown parameters. Given the set of instances, the goal is to learn the distributions $\Pr(T=t)$ and $\Pr(x_{q_r}|T=t)$ that maximizes the probability of the observation, that is, values of the instances. This maximum-likelihood problem is typically solved using an EM algorithm (Dempster et al., 1977) with a fixed number of clusters (values of T). The output of this algorithm includes a soft clustering of all instances; that is, $P(T|Y)$, where Y denotes the index of the instance.

In the following theorem we analyze the feature generalization properties of soft clustering by the multinomial mixture model. We show that under some technical conditions, it pursues nearly the same goal as *IobMax* algorithm (Definition 5), that is, maximizing $\sum_j I\left(\mathbf{t}(Z);X_{q_j}(Z)\right)$.

Theorem 7 *Let $I_{ob,ML,k}$ be the observed information of clustering achieved by the maximum likelihood solution of a multinomial mixture model for k clusters. Then*

$$I_{ob,ML,k} \geq \tilde{I}_{ob,k} - \frac{2H(T)}{n},$$

where $\tilde{I}_{ob,k}$ is the observed information achieved by the IobMax clustering algorithm (Definition 5).

Proof

Elidan and Friedman (2003) showed that learning a hidden variable can be formulated as the multivariate information bottleneck (Friedman et al., 2001). Based on their work, in Appendix A.3 we show that maximizing the likelihood of observed variables is equivalent to maximizing $\sum_{j=1}^{n} I(T;X_{q_j}) - I(T;Y)$. Using our notations, this is equivalent to maximizing $I_{ob} - \frac{1}{n}I(T;Y)$. Since $I(T;Y) \leq H(T)$, the difference between maximizing I_{ob} and $I_{ob} - \frac{1}{n}I(T;Y)$ is at most $2H(T)/n$. ■

The meaning of Theorem 7 is that for large n, finding the maximum likelihood of mixture models is similar to finding the maximum unobserved information. Thus the standard EM-algorithm for maximum likelihood of mixture models can be viewed as a form of the *IobMax* algorithm.[5]

The standard mixture model assumes a generative model for generating the instances from some distribution, and finds the maximum likelihood of this model. This model does not assume anything about the selection of features or the existence of unobserved features. Our setup assumes that the instances are fixed and the observed features are randomly selected and we try to maximize information on unobserved features. Interestingly, while the initial assumptions are quite different, the results are nearly equivalent. We show that finding the maximum likelihood of the mixture model indirectly predicts unobserved features as well.

The maximum likelihood mixture model was used by Breese et al. (1998) to cluster users by their voting on movies. This clustering is used to predict the rating of new movies. Our analysis shows that for a large number of rated (observed) movies, it is nearly the best clustering method in terms of information on new movies.

The multinomial mixture model is also used for learning with labeled and unlabeled instances, and is considered a baseline method (see Section 2.3 in Seeger, 2002). The idea is to cluster the instances based on their features. Then, the prediction of a label for an unlabeled instance is estimated from the labels of other instances in the same cluster. From our analysis, this is nearly the best clustering method for preserving information on the label, assuming that the label is yet another feature that happened to be unobserved in some instances. This provides another interpretation regrading the hidden assumption of this clustering scheme for labeled and unlabeled data.

3. Distance-Based Clustering

In this section we extend the framework and include analysis of feature generalization bounds for distance-based clustering. We apply this to analyze feature generalization of the k-means clustering algorithm (See Table 1). The setup in this section is the same as the setup defined in Section 2 except as described below. We assume that we have a distance function, denoted by f, that measures the distance for every two values of any of the features. We assume that f has the following properties:

$$0 \leq f(x_q[j], x_q[l]) \leq c \quad \forall q, j, l, \tag{2}$$

$$f(a,a) = 0 \qquad \forall a, \tag{3}$$

$$f(a,b) = f(b,a) \qquad \forall a, b, \tag{4}$$

where c is some positive constant. An example of such a function is the square error, that is, $f(a,b) = (a-b)^2$, where we assume that the value of all features is bounded as follows $\left|x_q[j]\right| \leq \sqrt{c}/2$ ($\forall q, j$), for some constant c. The features themselves can be discrete or continuous. Although we do not directly use the function f in the definitions of the theorems in the following section, it is required for their proofs (Appendix A.4).

5. Ignoring the fact that achieving a global maximum is not guaranteed.

3.1 Generalization of Distance-Based Clustering

As in Section 2, we have a set of m fixed instances $\{\mathbf{x}[1],...,\mathbf{x}[m]\}$, and the clustering algorithm clusters these instances into k clusters. For better readability, in this section the partition is denoted by $\{C_1,...,C_k\}$. $|C_r|$ denotes the size of the rth cluster.

The standard objective of the k-means algorithm is to achieve minimum intra-cluster variance, that is, minimize the function

$$\sum_{r=1}^{k}\sum_{j\in C_r}|\mathbf{x}[j]-\mu_r|^2,$$

where μ_r is the mean point of all instances in the rth cluster.

In our setup, however, we assume that the clustering algorithm has access only to the *observed* features over the m instances. The goal of clustering is to achieve minimum intra-cluster variance of the *unobserved* features. To do so, we need to generalize from the observed to the unobserved intra-class variance. To formalize this type of generalization, let's first define these variances formally.

Definition 8 *The **observed intra-cluster variance** $D_{ob}\{C_1,...,C_k\}$ of a clustering $\{C_1,...,C_k\}$ is defined by*

$$D_{ob}\{C_1,...,C_k\}=\frac{1}{nm}\sum_{r=1}^{k}\sum_{j\in C_r}\sum_{i=1}^{n}\left(x_{q_i}[j]-\mu_{q_i}[r]\right)^2,$$

where $\mu_q[r]$ is the mean of feature q over all instances in cluster r, that is,

$$\mu_q[r]=\frac{1}{|C_r|}\sum_{l\in C_r}x_q[l].$$

In other words, D_{ob} is the average square distance of each observed feature from the mean of the value of the feature in its cluster. The average is over all observed features and instances. The k-means algorithm minimizes the observed intra-cluster variance.

Definition 9 *The **expected unobserved intra-cluster variance** $D_{un}\{C_1,...,C_k\}$ is defined by*

$$D_{un}\{C_1,...,C_k\}=\frac{1}{m}\sum_{r=1}^{k}\sum_{j\in C_r}\mathbf{E}_{q\sim\mathcal{D}}\left(x_q[j]-\mu_q[r]\right)^2.$$

D_{ob} and D_{un} are the distance-based variables analogous to I_{ob} and I_{un} defined in Section 2. In our setup, the goal of the clustering algorithm is to create clusters with minimal unobserved intra-class variance (D_{un}). As in the case of information-based clustering, we first consider the problem of estimating D_{un}. Before presenting the generalization theorem for distance-based clustering, we need the following definition.

Definition 10 *Let α be the ratio between the size of smallest cluster and the average cluster size, that is,*

$$\alpha(\{C_1,...,C_k\})=\frac{\min_r|C_r|}{m/k}.$$

Now we are ready for the generalization theorem for distance-based clustering.

Theorem 11 *With the above definitions, if* $\left|x_q[j]\right| \leq R$ *for every* q, j *then for every* $\alpha_c > 0$, $\varepsilon > 0$,

$$\Pr_{\{q_1,\ldots,q_n\}} \left\{ \sup_{\alpha(\{C_1,\ldots,C_k\})\geq\alpha_c} \left| D_{ob}\{C_1,\ldots,C_k\} - D_{un}\{C_1,\ldots,C_k\} \right| \leq \varepsilon \right\} \geq 1-\delta,$$

where

$$\delta = \frac{8k}{\alpha_c} e^{-n\varepsilon^2/8R^4 + \log\left(R^2/\varepsilon\right)}.$$

The proof of this theorem is given in Appendix A.4. Theorem 11 is a special case of a more general theorem (Theorem 14) that we present in the appendix. Theorem 14 can be applied to other distance-based metrics, beyond the intra-cluster variance defined in Definition 9.

Note that for any $\varepsilon > 0$, the probability that $|D_{ob} - D_{un}| \leq \varepsilon$ goes to one, as $n \to \infty$. The convergence rate of D_{ob} to D_{un} is bounded by $O(1/\sqrt{n})$. As expected, for a fixed value of δ the upper bound on $|D_{ob} - D_{un}|$ decreases as the number of clusters, k, decreases.

Theorem 11 bounds the difference between observed and unobserved variances. We now use it to find a clustering that minimizes the expected unobserved intra-cluster variance, using only the observed features.

Theorem 12 *Let* $\left\{C_1^{opt},\ldots,C_k^{opt}\right\}$ *be the clustering that achieves the minimum unobserved intra-cluster variance under the constraint* $\alpha(\{C_1,\ldots,C_k\}) \geq \alpha_c$ *for some constant* $0 < \alpha_c \leq 1$, *that is,*

$$\left\{C_1^{opt},\ldots,C_k^{opt}\right\} = \arg \min_{\{C_1,\ldots,C_k\}:\alpha\geq\alpha_c} D_{un}\{C_1,\ldots,C_k\},$$

and let D_{un}^{opt} *the best unobserved intra-cluster variance, be defined by* $D_{un}^{opt} = D_{un}\left\{C_1^{opt},\ldots,C_k^{opt}\right\}$.

Let $\left\{\hat{C}_1^{opt},\ldots,\hat{C}_k^{opt}\right\}$ *be the clustering with the minimum observed intra-cluster variance, under the same constraint on* $\alpha(\{C_1,\ldots,C_k\})$, *that is,*

$$\left\{\hat{C}_1^{opt},\ldots,\hat{C}_k^{opt}\right\} = \arg \min_{\alpha(\{C_1,\ldots,C_k\})\geq\alpha_c} D_{ob}\{C_1,\ldots,C_k\},$$

and let $\hat{D}_{un}^{opt}$ *be the unobserved intra-cluster variance of this clustering, that is,* $\hat{D}_{un}^{opt} = D_{un}\left\{\hat{C}_1^{opt},\ldots,\hat{C}_k^{opt}\right\}$.

For any $\varepsilon > 0$,

$$\Pr_{\{q_1,\ldots,q_n\}} \left\{\hat{D}_{un}^{opt} \leq D_{un}^{opt} + \varepsilon\right\} \geq 1-\delta,$$

where

$$\delta = \frac{16k}{\alpha_c} e^{-n\varepsilon^2/32R^4 + \log\left(R^2/\varepsilon\right)}. \tag{5}$$

Proof We now define a *bad clustering* as a clustering whose expected unobserved intra-cluster variance satisfies $D_{un} > D_{un}^{opt} + \varepsilon$. Using Theorem 11, the probability that $|D_{ob} - D_{un}| \leq \varepsilon/2$ for all possible clusterings (under the constraint on α) is at least $1-\delta$, where δ defined in Equation 5. If for all clusterings $|D_{ob} - D_{un}| \leq \varepsilon/2$, then surely $D_{ob}\left\{C_1^{opt},\ldots,C_k^{opt}\right\} \leq D_{un}^{opt} + \varepsilon/2$ and D_{ob} of all bad clusterings satisfies $D_{ob} > D_{un}^{opt} + \varepsilon/2$. Hence the probability that any of the bad clusterings has a lower observed intra-cluster variance than the best clustering is upper bounded by δ. Therefore, with a probability of at least $1-\delta$ none of the bad clusterings is selected by an algorithm that selects

the clustering with the minimum D_{ob}. ■

We cannot directly calculate the unobserved intra-cluster variance. However, Theorem 12 means that an algorithm that selects the clustering with the minimum observed intra-cluster variance indirectly finds the clustering with nearly minimum unobserved intra-cluster variance.

In general, minimizing observed intra-cluster variance is the optimization objective of *k*-means. Hence, *k*-means indirectly minimizes the unobserved intra-cluster variance. This means that in our context, *k*-means can be viewed as an analog to the empirical risk minimization (ERM) in the standard supervised learning context. We minimize the observed variance (training error) in order to indirectly minimize the expected unobserved variance (test error).

k-means is used in collaborative filtering such as movie rating predictions for grouping users based on similar ratings (see, for example, Marlin, 2004). After clustering, we can predict ratings of a new movie based on the ratings of a few users for this movie. If the intra-cluster variance of a new, previously unobserved movie is small, then we can estimate the rating of one user from the average ratings of other users in the same cluster.

An experimental illustration of the behavior of the observed and unobserved intra-cluster variances for *k*-means is available in Section 4.1.

4. Empirical Evaluation

In this section we test experimentally the generalization properties of *IobMax* and the *k*-means clustering algorithm for a finite number of features. For *IobMax* we examine the difference between I_{ob} and I_{un} as a function of the number of observed features, and number of clusters used. We also compare the value of I_{un} achieved by the *IobMax* algorithm to I^*_{un}, which is the maximum achievable I_{un} (see Definition 4). Similarly, for distance-based clustering we use *k*-means to examine the behavior of the observed and unobserved intra-cluster variances (see Definitions 8, 9).

The purpose of this section is *not* to suggest new algorithms for collaborative filtering or compare it to other methods, but simply to illustrate our new theorems on empirical data.

4.1 Collaborative Filtering

In this section, our evaluation uses a data set typically employed for collaborative filtering. Collaborative filtering refers to methods of making predictions about a user's preferences, by collecting the preferences of many users. For example, collaborative filtering for movie ratings can make predictions about the rating of movies by a user given a partial list of ratings from this user and many other users. Clustering methods are used for collaborative filtering by clustering users based on the similarity of their ratings (see, for example, Marlin, 2004; Ungar and Foster, 1998).

In our setting, each user is described as a vector of movie ratings. The rating of each movie is regarded as a feature. We cluster users based on the set of observed features, that is, rated movies. In our context, the goal of the clustering is to maximize the information between the clusters and unobserved features, that is, movies that have not yet been rated by any of the users. These can be movies that have not yet been made. By Theorem 6, given a large enough number of rated movies, we can achieve the best possible clustering of users with respect to unseen movies. In this region, no additional information (such as user age, taste, rating of more movies) beyond the observed features can improve the unobserved information, I_{un}, by more than some small ε.

For distance-based clustering, we cluster the users by the *k*-means algorithm based on a subset of features (movies). As we show in Section 3.1 the goal of *k*-means is to minimize the observed intra-cluster variance. From Theorem 12, this indirectly minimizes the unobserved intra-cluster variance as well. Here we empirically evaluate this type of generalization.

Data set. We use MovieLens (www.movielens.umn.edu), which is a movie rating data set. It was collected and distributed by GroupLens Research at the University of Minnesota. It contains approximately 1 million ratings of 3900 movies by 6040 users. Ratings are on a scale of 1 to 5. We use only a subset consisting of 2400 movies rated by 4000 users (or 2000 by 2000 for distance-based clustering). In our setting, each instance is a vector of ratings $(x_1,\ldots,x_{2400})$ by a specific user. Each movie is viewed as a feature, where the rating is the value of the feature.

Experimental Setup. We randomly split the 2400 movies into two groups, denoted by "A" and "B", of 1200 movies (features) each. We use a subset of the movies from group "A" as observed features and all movies from group "B" as the unobserved features. The experiment was repeated with 20 random splits and the results averaged. We estimate I_{un} by the mean information between the clusters and ratings of movies from group "B". We use a uniform distribution of feature selection ($\mathcal{D}$), and hence I_{un} can be estimated as the average information on the unobserved features, that is, $I_{un} = \frac{1}{1200}\sum_{j\in B} I(T;X_j)$. A similar setup is used for the distance-based clustering (with two groups of 1000 movies).

Handling Missing Values. In this data set, most of the values are missing (not rated). For information based-clustering, we handle this by defining the feature variable as 1,2,...,5 for the ratings and 0 for a missing value. We maximize the mutual information based on the empirical distribution of values that are present, and weight it by the probability of presence for this feature. Hence, $I_{ob} = \sum_{j=1}^{n} P(X_j \neq 0) I(T;X_j|X_j \neq 0)$ and $I_{un} = E_j\left\{P(X_j \neq 0) I(T;X_j|X_j \neq 0)\right\}$. The weighting prevents overfitting to movies with few ratings. Since the observed features are selected at random, the statistics of missing values of the observed and unobserved features are the same. Hence, all our theorems are applicable to these definitions of I_{ob} and I_{un} as well.

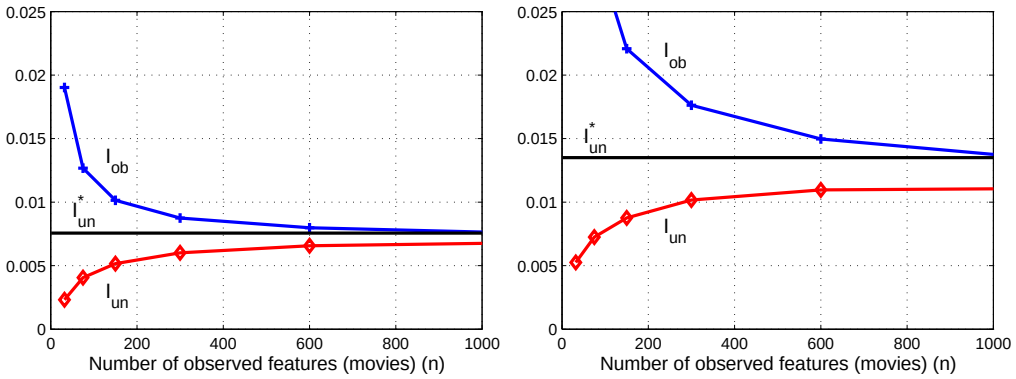
In order to verify that the estimated mutual information is not just an artifact of the finite sample size, we tested the mutual information after random permutation of ratings of each movie among users. Indeed, the resulting mutual information was significantly lower in the case of random permutation.

For the distance based clustering, we handle missing data by defining a default square distance between a feature and the cluster center where one (or two) of the values is missing. We select this default square distance to be the average variance of movie ratings (which is about 0.9).

4.2 Greedy *IobMax* Algorithm

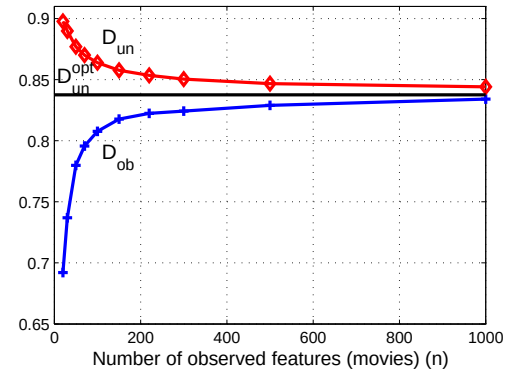
For information-based clustering, we cluster the users using a simple greedy clustering algorithm (see Algorithm 1). The input to the algorithm is all users, represented solely by the observed features. Since this algorithm can only find a local maximum of I_{ob}, we ran the algorithm 10 times (each used a different random initialization) and selected the results that had a maximum value of I_{ob}.

In our experiment, the number of observed features is large. Therefore, based on Theorem 7, the greedy *IobMax* can be replaced by the standard EM-algorithm which finds the maximum likelihood for multinomial mixture models. Although, in general, this algorithm finds soft clustering, in our

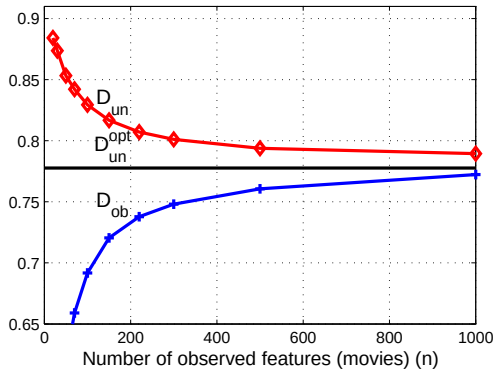


(a) 2 Clusters

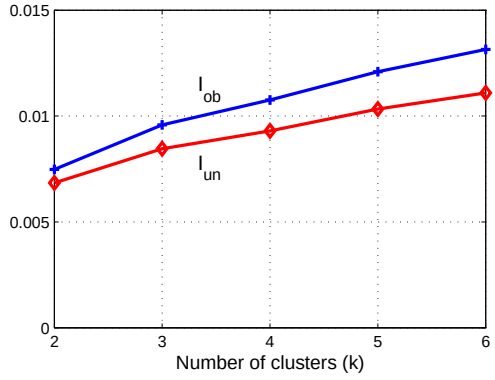
(b) 6 Clusters



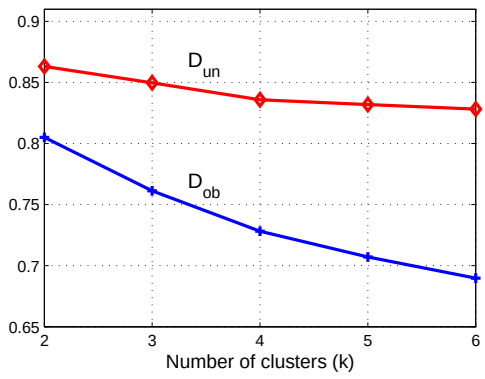
(c) 2 Clusters



(d) 6 Clusters



(e) Fixed number of features (n=1200)



(f) Fixed number of features (n=100)

Figure 3: Feature generalization as a function of the number of training features (movies) and the number of clusters. (a) (b) and (e) show the observed and unobserved information for various numbers of features and clusters (high is good). The overall mean information is low, since the rating matrix is sparse. (c) (d) and (f) shows the observed and unobserved intra-cluster variance (low is good). In these figures, the variance is only calculated on values which are not missing. Figures (e) and (f) show the effect of the number of clusters when the number of features is fixed.

case the empirical result clusterings are not soft, that is, one cluster is assigned to each instance (see Appendix A.2). As expected, the results of both algorithms are nearly the same.

Algorithm 1 A simple greedy *IobMax* algorithm

1. Assign a random cluster to each of the instances.
2. For $r = 1$ to R (where R is the upper limit on the number of iterations)
   (a) For each instance,
      i. Calculate I_{ob} for all possible clustering assignments of the current instance.
      ii. Choose the clustering that maximizes I_{ob}.
   (b) Exit if the clusters of all documents do not change.

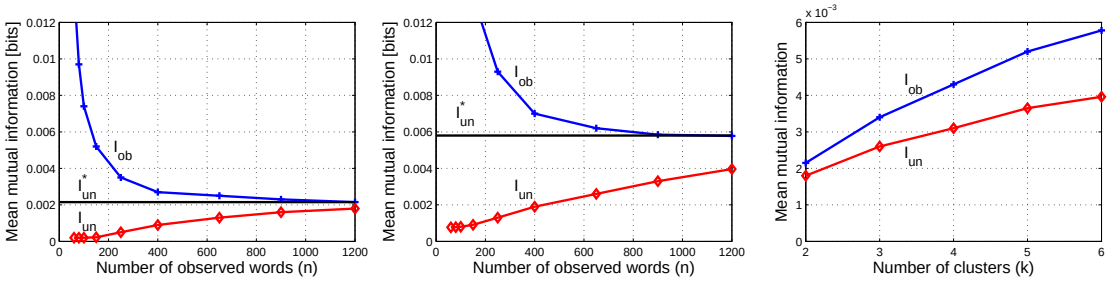
In order to estimate $I^*_{un,D}$ (see Definition 4), we also ran the same algorithm when all the features were available to the algorithm (i.e., also features from group "B"). In this case the algorithm tries directly to find the clustering that maximizes the mean mutual information on features from group "B".

4.3 Results

The results are shown in Figure 3. It is clear that as the number of observed features increases, I_{ob} decreases while I_{un} increases (see Figure 3(a,b)). When there is only one feature, two clusters can contain all the available information on this feature (e.g., by assigning $t(j) = x_{q_1}[j]$), so I_{ob} reaches its maximum value (which is $H(X_{q_1}[Z])$). As the number of observed features increases, we cannot preserve all the information on all the features in a few clusters, so the observed mutual information (I_{ob}) decreases. On the other hand, as the number of observed features increases, the cluster variable, $T = t(Z)$, captures the structure of the distribution (users' tastes), and hence contains more information on unobserved features. The generalization theorem (Theorem 3) tells us that the difference between I_{un} and I_{ob} will approach zero as the number of observed features increases. This is similar to the behavior of training and test errors in supervised learning. Informally, the achievability theorem (Theorem 6) tells as that for a large enough number of observed features, even though our clustering algorithm is based only on observed features, it can achieve nearly the best possible clustering, in terms of I_{un}. This can be seen in Figures 3 (a,b), where I_{un} approaches I^*_{un}, which is the unobserved information of the best clustering (Definition 4). As the number of clusters increases, both I_{ob}, I_{un} increase (Figure 3e), but the difference between them also increases.

Similar results were obtained for distance based clustering. The goal here is to minimize the unobserved intra-cluster variance (D_{un}), and this is done by minimizing the observed intra-cluster variance (D_{ob}). As discussed in Section 3.1, this can be achieved by k-means.[6] Again, for a small numbers of features (n) the clustering overfits the observed features, that is, the D_{ob} is relatively low but D_{un} is large. However, for large n, D_{un} and D_{ob} approach each other and both of them approach the unobserved intra-cluster variance of the best possible clustering (D^{opt}_{un}) as expected

6. Since k-means does not necessarily find the global optimum, we ran it 20 times with different initialization points, and chose the results with minimal observed intra-cluster variance. This does not guarantee a global optimum, but no other tractable algorithm is available today to achieve global optima.



(a) 2 Clusters (b) 6 Clusters (c) Fixed 1200 observed words

Figure 4: I_{ob}, I_{un} and I^*_{un} per number of training words and clusters. In (a) and (b) the number of words is variable, and the number of clusters is fixed. In (c) the number of observed words is fixed (1200), and the number of clusters is variable. The overall mean information is low, since a relatively small number of words contributes to the information (see Table 2)

from Theorem 12. When the number of clusters increases, both D_{ob} and D_{un} decrease, but the difference between them increases.

4.4 Words and Documents

In this section we repeat the information-based clustering experiment, but this time for document clustering with words as features. We show how clustering which is based on a subset of words (observed words) is also informative about the unobserved words. The obtained curves of information vs. number of features are similar to those in the previous section. However, in this section we also examine the resulting clustering (Table 2) to get a better intuition as to how this generalization occurs.

Data set. We use the 20-newsgroups (20NG) corpus, collected by Lang (1995). This collection contains about 20,000 messages from 20 Usenet discussion groups, some of which have similar topics.

Preprocessing. In order to prevent effects caused by different document lengths, we truncate each document to 100 words (by randomly selecting 100 words), and ignore documents which consist of fewer than 100 words. We use the "bag of words" representation: namely we convert each document into a binary vector $(x_1, x_2, \ldots)$, where each element in the vector represents a word, and equals one if the word appears in the document and zero otherwise. We select the 2400 words whose corresponding X_i has maximum entropy,[7] and remove all other words. After this preprocessing each document is represented by a vector $(x_1, \ldots, x_{2400})$.

Experimental setup. We randomly split the 2400 words into two groups of 1200 words (features) each. The groups were called "A" and "B". We use a variable number of words (1 to 1200) from group "A" as observed features. All the features from group "B" are used as the unobserved features. We repeat the test with 10 random splits and present the mean results.

7. In other words, the probability of the word appearing in a document is not near zero or near one.

	Word	t=1	t=2	t=3
Observed words	he	0.47	0.04	0.25
	game	0.23	0.01	0
	team	0.20	0	0
	x	0.01	0.15	0.01
	hockey	0.11	0	0
	jesus	0.01	0	0.09
	christian	0	0	0.08
	use	0.04	0.21	0.09
	file	0	0.08	0.01
Unobserved words	god	0.02	0.01	0.15
	players	0.13	0	0
	baseball	0.10	0	0
	window	0	0.10	0
	server	0	0.06	0

Table 2: Probability of a word appearing in a document from each cluster. Each column in the table represents a cluster (total of three clusters), and the numbers are the probabilities that a document from a cluster will contain the word (e.g., The word "he" appears in 47% of the documents from cluster 1). The results presented here are for learning from 1200 observed words, but only a few of the most informative words appear in the table.

4.5 Results

The results are shown in Figure 4 and Table 2. The qualitative explanation of the figure is the same as for collaborative filtering (see Section 4.1 and Figure 3). Table 2 presents a list of the most informative words, and their appearance in each cluster. This helps understand the way clustering learned from observed words matches unobserved words. We can see, for example, that although the word "player" is not part of the inputs to the clustering algorithm, it appears much more in the first cluster than in other clusters. Intuitively this can be explained as follows. The algorithm finds clusters that are informative on many observed words together, and thus matches the co-occurrence of words. This clustering reveals the hidden topics of the documents (sports, computers and religious), and these topics contain information on the unobserved words. We see that generalization to unobserved features can be explained from a standpoint of a generative model (a hidden variable which represents the topics of the documents) or from a statistical point of view (relationship between observed and unobserved information). In Section 6 we further discuss this dual view.

5. Related Work

In the framework of learning with labeled and unlabeled data (see, for example, Seeger, 2002), a fundamental issue is the link between the marginal distribution $P(\mathbf{x})$ over instances $\mathbf{x}$ and the conditional $P(y|\mathbf{x})$ for the label y (Szummer and Jaakkola, 2003). From this point of view our approach assumes that the label, y, is a feature in itself.

In the context of supervised learning, Ando and Zhang (2005) proposed an approach that regards features in the input data as labels in order to improve prediction in the target supervised problem. Their idea is to create many auxiliary problems that are related to the target supervised problem. They do this by masking some features in the input data, that is, making them unobserved features, and training classifiers to predict these unobserved features from the observed features. Then they transfer the knowledge acquired from these related classification tasks to the target supervised problem. A similar idea was used by Caruana and de Sa (1997) in supervised training of a neural net. The authors used some features as extra outputs of the neural net, rather than inputs, and show empirically that this can improve the classifier performance. In our framework, this could be interpreted as follows. We regard the label as a feature, and hence we can learn from prediction of these features to the prediction of the label. Loosely speaking, if we successfully predict many such features by the classifier, we expect to generalize better to the target feature (label).

The idea of an information tradeoff between complexity and information on target variables is similar to the idea of the information bottleneck (Tishby et al., 1999). But unlike the bottleneck method, here we are trying to maximize information on *unobserved* variables, using a finite sample.

In a recent paper, von Luxburg and Ben-David (2005) discuss the goal of clustering in two very different cases. The first is when we have complete knowledge about our data generating process, and the second is how to approximate an optimal clustering when we have incomplete knowledge about our data. In most current analyses of clustering methods, incomplete knowledge refers to getting a finite sample of instances rather than the distribution itself. Then, we can define the desired properties of a good clustering. An example of such a property is the stability of the clustering with respect to the sampling process, for example, the clusters do not change significantly if we add some data points to our sample. In our framework, even if the distribution of the instances is completely known, we assume that there are other features that we might not be aware of at the time of clustering. Another way to view this is that in our framework, incomplete knowledge refers to the existence of unobserved features rather than to an unknown distribution of the observed features. From this point of view, further research could concentrate on analyzing the feature stability of a clustering algorithm, for example, stability with respect to the adding of new features.

Another interesting work which addresses the difficulty of defining good clustering was presented by Kleinberg (2002). In this work the author states the desired properties a clustering algorithm should satisfy, such as scale invariances and richness of possible clusterings. Then he proves that it is impossible to construct a clustering that satisfies all the required properties. In his work the clustering depends on pairwise distances between data points. In our work, however, the analysis is feature oriented. We are interested in the information (or distance) per feature. Hence, our basic assumptions and analysis are very different.

The idea of generalization to unobserved features by clustering was first presented in a short version of this paper (Krupka and Tishby, 2005).

6. Discussion

We introduce a new learning paradigm: clustering based on observed features that generalizes to unobserved features. Our main results include two theorems that tell us how, without knowing the value of the unobserved features, one can estimate and maximize information between the clusters and the unobserved features. Using this framework we analyze feature generalization of the Maximum Likelihood Multinomial Mixture Model (Figure 2). The multinomial mixture model is a

generative probabilistic model which approximates the probability distribution of the observed data ($\mathbf{x}$), from a finite sample of instances. Our model does not assume any distribution that generated the instances, but instead assumes that the set of observed features is simply a random subset of features. Then, using statistical arguments we show that we can cluster by the unobserved features. Despite the very different assumptions of these models, we show that clustering by multinomial mixture models is nearly optimal in terms of maximizing information on unobserved features. However, to analyze and quantify this generalization our framework is required.

This dual view on the multinomial mixture model can also be applied to two different approaches that may explain our "natural clustering" of objects in the world (e.g., assigning object names in language). Let's return to our clustering of bananas and oranges (Example 3). From the generative point of view, we find a model with the cluster labels bananas and oranges as values of a hidden variable that created the distribution. This means that we have a mixture of two distributions, each related to one object type that is assigned to a different cluster. Since we have two types of objects (distributions), we expect that their unobserved features will correspond to these two types as well. However, the generative model does not quantify this expectation. In our framework, we view fruits in the world, and cluster them based on some kind of *IobMax* algorithm; that is, we find a representation (clustering) that contains significant information on as many observed features as possible, while still remaining simple. From our generalization theorem (Theorem 3), such a representation is expected to contain information on other rarely viewed salient features as well. Moreover, we expect this unobserved information to be similar to the information we have on the clustering on the observed features.

In addition to information-based clustering, we present similar generalization theorems for distance-based clustering, and use these to analyze generalization properties of *k*-means. Under some assumptions, *k*-means is also known as a solution for the maximum likelihood Gaussian mixture model. Analogous to what we show for information based clustering and multinomial mixture models, we show that this optimization goal of *k*-means is also optimal in terms of generalization to unobserved features.

The key assumption that enables us to prove these theorems is the *random independent selection* of the observed features. Note that a contrary assumption to random selection would be that given two instances $\{\mathbf{x}[1],\mathbf{x}[2]\}$, there is a correlation between the distance of a feature $\left|x_q[1]-x_q[2]\right|$ and the probability of observing this feature; for example, the probability of *observing* features that are similar is higher. If no such correlation exists, then the selection can be considered random in our context. Hence, we believe that in practice the random selection assumption is reasonable. However, in many cases, the assumption of complete independence in the selection of features is less natural. Therefore, we believe that further research on the effects of dependence in selection is required.

Another interpretation of the generalization theorem, without using the *random independence* assumption, might be combinatorial. The difference between the observed and unobserved information is large only for a small portion of all possible partitions into observed and unobserved features. This means that almost any arbitrary partition generalizes well.

The value of clustering which preserves information on unobserved features is that it enables us to learn new—previously unobserved—attributes from a small number of examples. Suppose that after clustering fruits based on their observed features (Example 3), we eat a chinaberry[8] and thus,

8. Chinaberries are the fruits of the Melia azedarach tree, and are poisonous.

we "observe" (by getting sick), the previously unobserved attribute of toxicity. Assuming that in each cluster, all fruits have similar unobserved attributes, we can conclude that all the fruits in the same cluster, that is, all chinaberries are likely to be poisonous.

Clustering is often used in scientific research, when collecting measurements on objects such as stars or neurons. In general, the quality of a theory in science is measured by its predictive power. Therefore, a reasonable measure of the quality of clustering, as used in scientific research, is its ability to predict unobserved features, or measurements. This is different from clustering that merely describes the observed measurements, and supports the rationale for defining the quality of clustering by its predictivity on unobserved features.

6.1 Further Research

Our clustering maximizes expected information on randomly selected features. Although on average this information may be high, there might be features the clustering has no information about. To address this problem, we could create more than one clustering, in such a way that each clustering contains information on other features. To achieve this, we want each new clustering to discover new information, that is, not to be redundant with previously created clusterings. This can be done based on the works of Gondek and Hofmann (2004) and Chechik and Tishby (2002) in the context of the Information Bottleneck. Another alternative is to represent each instance by a low dimensional vector, and then use this vector to predict unobserved features. Blitzer et al. (2005) represented words in a model called Distributed Binary Latent Variables, and used this representation to predict another word. Adopting this idea in our context, we can replace cluster labels by a vector of binary variables assigned to each instance, where each such variable encodes an independent aspect of the instance. Generalization in this case refers to the ability to predict unobserved features from these latent variables.

Our framework can also be extended beyond clustering by formulating a general question. Given the (empirical) marginal distribution of a random subset of features $P(X_{q_1},\ldots,X_{q_n})$, what can we say about the distribution of the full set $P(X_1,\ldots,X_L)$? In this paper we proposed a clustering based on a subset of features, and analyzed the information that the clustering yielded on features outside this subset. It would be useful to find more sophisticated representations than clustering, and analyze other theoretical aspects of the relationship between the distribution of the subset to that of the full set. This type of theoretical analysis can help in deriving prediction algorithms, where there are many instances for some of the variables (features), but other variables are rarely viewed, as in collaborative filtering. By relating the distribution of some variables to the distribution of others, we can also analyze and improve the estimation of $p(\mathbf{x})$ from a finite sample, even without assuming the existence of unobserved features. In a different context, Han (1978) analyzed the relationship between the average (per variable) entropy of random subsets of variables. He showed that the average entropy of a random subset of variables monotonically decreases with the size of the subset (see also Cover and Thomas, 1991). These results were developed in the context of information theory and compression, but may be applicable to learning theory as well.

In this paper, we assumed that we do not have additional prior information on the features. In practice, we often do have such information. For instance, in the movie ratings data set, we have some knowledge about each of the movies (genre, actors, year, etc.). This knowledge about the features can be regarded as meta-features. A possible extension of our framework is to use this knowledge to improve and analyze generalization as a function of the meta-features of the

unobserved features. The idea of learning along the features axis by using meta-features was implemented by Krupka et al. (submitted) for feature selection. They propose a method for learning to select features based on the meta-features. Using the meta-features we can learn what *types* of features are good, and predict the quality of unobserved features. They show that this is useful for feature selection out of a huge set of potentially extracted features; that is, features that are functions of the input variables. In this case all features can be observed, but in order to measure their quality we must calculate them for all instances, which might be computationally intractable. By predicting feature quality without calculating it, we can focus the search for good features on a small subset of the features. In a recent paper (Krupka and Tishby, 2007), we propose a method for learning the weights of a linear classifier based on meta-features. The idea is to learn weights as a function of the meta-features just as we learn labels as a function of features. Then, we can learn from feature to feature and not only from instance to instance. As shown empirically, this can significantly improve classification performance in the standard supervised learning setting.

In this work we focused on a new feature generalization analysis. Another research direction is to combine standard instance generalization with feature generalization. In problems like collaborative filtering or gene expression, there is an inherent symmetry between features and instances that have been used before in various ways (see, for example, Ungar and Foster, 1998). In the context of supervised learning, a recent work by Globerson and Roweis (2006) addresses the issue of handling differences in the set of observed features between training and test time. However, a general framework for generalization to both unobserved features and unobserved instances is still lacking.

Acknowledgments

We thank Aharon Bar-Hillel, Amir Globerson, Ran Bachrach, Amir Navot and Ohad Shamir for helpful discussions. We also thank the GroupLens Research Group at the University of Minnesota for use of the MovieLens data set. Our work is partly supported by the Center of Excellence grant from the Israeli Academy of Science and by the NATO SfP 982480 project.

Appendix A. Proofs

This appendix contains the proofs of Theorem 3 and Theorem 11. It also contains additional technical details that were used in the proof of Theorem 7.

A.1 Proof of Theorem 3

We start by introducing the following lemma, which is required for the proof of Theorem 3.

Lemma 13 *Consider a function g of two independent discrete random variables (U,V). We assume that $g(u,v) \leq c$, $\forall u,v$, where c is some constant. If $\Pr\{g(U,V) > \tilde{\varepsilon}\} \leq \delta$, then*

$$\Pr_V\{\mathbf{E}_u(g(u,V)) \geq \varepsilon\} \leq \frac{c-\tilde{\varepsilon}}{\varepsilon-\tilde{\varepsilon}}\delta, \quad \forall \varepsilon > \tilde{\varepsilon}.$$

Proof of lemma 13: Let $\mathcal{V}_L'$ be the set of values of V, such that for every $v' \in \mathcal{V}_L'$, $E_u(g(y,v')) \geq \varepsilon$. For every such v' we get,

$$\varepsilon \leq \mathbf{E}_u\left(g(u,v')\right) \leq c\Pr\left\{g(U,V) > \tilde{\varepsilon}|V=v'\right\} + \tilde{\varepsilon}\Pr\left\{g(U,V) \leq \tilde{\varepsilon}|V=v'\right\}.$$

Hence, $\Pr\{g(U,V) > \tilde{\varepsilon} | V = v'\} \geq \frac{\varepsilon - \tilde{\varepsilon}}{c - \tilde{\varepsilon}}$. From the complete probability formula,

$$\begin{aligned} \delta \geq \Pr\{g(U,V) > \tilde{\varepsilon}\} &= \sum_z \Pr\{g(U,V) > \tilde{\varepsilon} | V = v\} P(v) \\ &\geq \frac{\varepsilon - \tilde{\varepsilon}}{c - \tilde{\varepsilon}} \sum_{V : V \in V_L} P(v) \\ &= \frac{\varepsilon - \tilde{\varepsilon}}{c - \tilde{\varepsilon}} \Pr_V \{\mathbf{E}_u (g(u,V)) \geq \varepsilon\}. \end{aligned}$$

Lemma 13 follows directly from the last inequality. □

We first provide an outline of the proof of Theorem **3** and then provide a detailed proof.

Theorem 3—Proof outline: For the given m instances and any clustering $\mathbf{t}$, draw uniformly and independently m' instances (repeats allowed). For any feature index q, we can estimate $I(\mathbf{t}(Z); x_q[Z])$ from the empirical distribution of $(\mathbf{t}, x_q)$ over the m' instances. This empirical distribution is $p(\mathbf{t}(Z'), x_q[Z'])$ where Z' is a random variable denoting the index of instance chosen uniformly from the m' instances (defined formally below). The proof is built up from the following upper bounds, which are independent of m, but depend on the choice of m'. The first bound is on $\mathbf{E}\left\{\left|I(\mathbf{t}(Z); x_q[Z]) - I(\mathbf{t}(Z'); x_q[Z'])\right|\right\}$, where q is fixed and the expectation is over random selection of the m' *instances*. From this bound we derive an upper bound on $|I_{ob} - \mathbf{E}(\hat{I}_{ob})|$ and $|I_{un} - \mathbf{E}(\hat{I}_{un})|$, where $\hat{I}_{ob}$, $\hat{I}_{un}$ are the estimated values of I_{ob}, I_{un} based on the subset of m' instances, that is, the empirical distribution. The last required bound is on the probability that $\sup_{\mathbf{t}:[m]\to[k]} \left|\mathbf{E}\left(\hat{I}_{ob}\right) - \mathbf{E}\left(\hat{I}_{un}\right)\right| > \varepsilon_1$, for any $\varepsilon_1 > 0$. This bound is obtained from Lemmas 2 and 13. The choice of m' is independent of m. Its value should be large enough for the estimations $\hat{I}_{ob}$, $\hat{I}_{un}$ to be accurate, but not too large, so as to limit the number of possible clusterings over the m' instances.

Note that we do not assume the m instances are drawn from a distribution. The m' instances are drawn from the empirical distribution over the m instances. □

Theorem 3—Detailed proof: Let $\tilde{\mathbf{l}} = (l_1, \ldots, l_{m'})$ be indices of m' instances, where each index is selected randomly, uniformly and independently from $\{1, \ldots, m\}$. Let random variable Z' denote a number chosen uniformly at random from $\{1, \ldots, m'\}$. For any feature index q, we can estimate $I(\mathbf{t}(Z); x_q[Z])$ from $I(\mathbf{t}(l_{Z'}); x_q[l_{Z'}])$ as follows. The maximum likelihood estimation of entropy given a discrete empirical distribution $(\hat{p}_1, ..., \hat{p}_N)$, is defined as $\hat{H}_{MLE} = -\sum_{i=1}^{N} \hat{p}_i \log \hat{p}_i$. Note that N is the alphabet size of our discrete distribution. From Paninski (2003) (Proposition 1) the bias between the empirical and actual entropy $H(p)$ is bounded as follows:

$$-\log\left(1 + \frac{N-1}{m'}\right) \leq \mathbf{E}\left(\hat{H}_{MLE}(\hat{p})\right) - H(p) \leq 0.$$

where the empirical estimation $\hat{H}_{MLE}$ is based on m' instances drawn from the distribution p. The expectation is over random sampling of these m' instances. Since $I(\mathbf{t}(Z); x_q[Z]) = -H(\mathbf{t}(Z), x_q[Z]) + H(\mathbf{t}(Z)) + H(x_q(Z))$, we can upper bound the bias between the actual and the empirical estimation of the mutual information as follows:

$$\mathbf{E}_{\tilde{\mathbf{l}}=(l_1,\ldots,l_{m'})}\left\{\left|I(\mathbf{t}(Z); x_q[Z]) - I(\mathbf{t}(l_{Z'}); x_q[l_{Z'}])\right|\right\} \leq \log\left(1 + \frac{ks-1}{m'}\right) \leq \frac{ks}{m'}. \tag{6}$$

Recall that s is the upper bound on the alphabet size of x_q.

Let $\hat{I}_{ob}(\mathbf{t}, \tilde{\mathbf{q}}, \tilde{\mathbf{l}})$ and $\hat{I}_{un}\left(\mathbf{t}, \tilde{\mathbf{l}}\right)$ be the estimated values of $I_{ob}(\mathbf{t}, \tilde{\mathbf{q}})$, $I_{un}(\mathbf{t})$ based on $(l_1, \ldots, l_{m'})$, that is,

$$\hat{I}_{ob}\left(\mathbf{t}, \tilde{\mathbf{q}}, \tilde{\mathbf{l}}\right) = \frac{1}{n} \sum_{i=1}^{n} I\left(t(l_{z'}); x_{q_i}[l_{z'}]\right),$$

$$\hat{I}_{un}(\mathbf{t},\tilde{\mathbf{l}}) = \mathbf{E}_{q\sim\mathcal{D}}\left\{I\left(\mathbf{t}(l_{z'});x_q[l_{z'}]\right)\right\}.$$

From Equation 6 we obtain,

$$|I_{ob}(\mathbf{t},\tilde{\mathbf{q}}) - \mathbf{E}_{\tilde{\mathbf{l}}}\left(\hat{I}_{ob}\left(\mathbf{t},\tilde{\mathbf{q}},\tilde{\mathbf{l}}\right)\right)|,\ |I_{un}(\mathbf{t}) - \mathbf{E}_{\tilde{\mathbf{l}}}\left(\hat{I}_{un}\left(\mathbf{t},\tilde{\mathbf{l}}\right)\right)| \le ks/m',$$

and hence,

$$\begin{aligned} |I_{ob}(\mathbf{t},\tilde{\mathbf{q}}) - I_{un}(\mathbf{t})| &\le \left|\mathbf{E}_{\tilde{\mathbf{l}}}\left(\hat{I}_{ob}\left(\mathbf{t},\tilde{\mathbf{q}},\tilde{\mathbf{l}}\right)\right) - \mathbf{E}_{\tilde{\mathbf{l}}}\left(\hat{I}_{un}\left(\mathbf{t},\tilde{\mathbf{l}}\right)\right)\right| + 2ks/m' \\ &\le \mathbf{E}_{\tilde{\mathbf{l}}}\left(\left|\hat{I}_{ob}\left(\mathbf{t},\tilde{\mathbf{q}},\tilde{\mathbf{l}}\right) - \hat{I}_{un}\left(\mathbf{t},\tilde{\mathbf{l}}\right)\right|\right) + 2ks/m'. \end{aligned} \tag{7}$$

Using Lemma 2 we have an upper bound on the probability that

$$\sup_{\mathbf{t}:[m]\to[k]} \left|\hat{I}_{ob}\left(\mathbf{t},\tilde{\mathbf{q}},\tilde{\mathbf{l}}\right) - \hat{I}_{un}\left(\mathbf{t},\tilde{\mathbf{l}}\right)\right| > \varepsilon$$

over the random selection of *features*, as a function of m'. However, the upper bound we need is on the probability that

$$\sup_{\mathbf{t}:[m]\to[l]} \left\{\mathbf{E}_{\tilde{\mathbf{l}}}\left(\hat{I}_{ob}\left(\mathbf{t},\tilde{\mathbf{q}},\tilde{\mathbf{l}}\right)\right) - \mathbf{E}_{\tilde{\mathbf{l}}}\left(\hat{I}_{un}\left(\mathbf{t},\tilde{\mathbf{l}}\right)\right)\right\} > \varepsilon_1.$$

Note that the expectations $\mathbf{E}_l(\hat{I}_{ob})$, $\mathbf{E}_l(\hat{I}_{un})$ are done over a random selection of the subset of m' *instances*, for a set of features that is randomly selected *once*. In order to link these two probabilities, we use Lemma 13.

From Lemmas 2 and 13 it is easy to show that

$$\Pr_{\tilde{\mathbf{q}}}\left\{\mathbf{E}_{\tilde{\mathbf{l}}}\left(\sup_{\mathbf{t}:[m]\to[k]}\left|\hat{I}_{ob}\left(\mathbf{t},\tilde{\mathbf{q}},\tilde{\mathbf{l}}\right) - \hat{I}_{un}\left(\mathbf{t},\tilde{\mathbf{l}}\right)\right|\right) > \varepsilon_1\right\} \le \frac{4\log k}{\varepsilon_1} e^{-n\varepsilon_1^2/\left(2(\log k)^2\right)+m'\log k}. \tag{8}$$

Lemma 13 is used, where V represents the random selection of features, U represents the random selection of m' instances, $g(u,v) = \sup_{\mathbf{t}:[m]\to[k]}|\hat{I}_{ob} - \hat{I}_{un}|$, $c = \log k$, and $\tilde{\varepsilon} = \varepsilon_1/2$. Since

$$\mathbf{E}_{\tilde{\mathbf{l}}}\left(\sup_{\mathbf{t}:[m]\to[k]}\left|\hat{I}_{ob}\left(\mathbf{t},\tilde{\mathbf{q}},\tilde{\mathbf{l}}\right) - \hat{I}_{un}\left(\mathbf{t},\tilde{\mathbf{l}}\right)\right|\right) \ge \sup_{\mathbf{t}:[m]\to[k]} \mathbf{E}_{\tilde{\mathbf{l}}}\left(\left|\hat{I}_{ob}\left(\mathbf{t},\tilde{\mathbf{q}},\tilde{\mathbf{l}}\right) - \hat{I}_{un}\left(\mathbf{t},\tilde{\mathbf{l}}\right)\right|\right),$$

and from Equations 7 and 8 we obtain

$$\Pr_{\tilde{\mathbf{q}}}\left\{\sup_{\mathbf{t}:[m]\to[k]}|I_{ob}(\mathbf{t},\tilde{\mathbf{q}}) - I_{un}(\mathbf{t})| > \varepsilon_1 + \frac{2ks}{m'}\right\} \le \frac{4\log k}{\varepsilon_1} e^{-n\varepsilon_1^2/\left(2(\log k)^2\right)+m'\log k}.$$

By selecting $\varepsilon_1 = \varepsilon/2$, $m' = 4ks/\varepsilon$, we obtain Theorem 3. □

Note that the selection of m' depends on s (maximum alphabet size of the features). This reflects the fact that in order to accurately estimate $I(\mathbf{t}(Z);x_q[Z])$, we need a number of instances, m', which is much larger than the product of k and the alphabet size of x_q.

A.2 Information Generalization for Soft Clustering

In Section 2 we assumed that we are dealing with hard clustering. Here we show that the generalization theorem (Theorem 3) is also applicable to soft clustering. Nevertheless, we also show that soft clustering is not required, since the maximum value of I_{ob} can be achieved by hard clustering. Hence, although *IobMax*, as appears in Definition 5, is a hard clustering algorithm, it also achieves maximum I_{ob} (and nearly maximum I_{un}) of all possible soft clusterings.

Theorem 3 is applicable to soft clustering from the following arguments. In terms of the distributions $P(\mathbf{t}(Z), x_q(Z))$, assigning a soft clustering to an instance can be approximated by a second empirical distribution, $\hat{P}$, achieved by duplicating each of the instances, and then using hard clustering. Consider, for example, a case where we create a new set of instances by duplicating each of the original instances by 100 identical instances. Using hard clustering on the $\times 100$ larger set of instances, can approximate any soft clustering of the original set with quantization of $P(T|\mathbf{X})$ in steps of $1/100$. Obviously, for any $\varepsilon > 0$ we can create $\hat{P}$ that satisfies $max\left|P - \hat{P}\right| < \varepsilon$.

Now we show that for any soft clustering of an instance, we can find a hard clustering of the same instance that has the same or a higher value of I_{ob} (without changing the cluster identity of other instances). This is enough to show that soft clustering is not required to achieve the maximum value of I_{ob}, since any soft clustering can be replaced by hard clustering instance by instance. Let $P_\lambda(T|X_{q_1},\ldots,X_{q_n})$ define the distribution of any soft clustering. It can be written as the weighted sum of k distributions as follows

$$P_\lambda(T|X_{q_1},\ldots,X_{q_n}) = \sum_{i=1}^{k} \lambda_i \tilde{P}_i^j(T|X_{q_1},\ldots,X_{q_n}), \quad 0 \le \lambda_i \le 1, \sum_{i=1}^{k} \lambda_i = 1.$$

where $\tilde{P}_i^j$ is created by keeping the same soft clustering of instances $\{1,\ldots,j-1,j+1,...,m\}$, and replacing the soft clustering of the jth instance by a hard clustering $\mathbf{t}(j) = i$. Since $I(T;X_q)$ is a convex function of $P(T|X_q)$ for a fixed $P(X_q)$ for any q (Cover and Thomas, 1991), we get

$$I_{P_\lambda}(T;X_q) \le \sum_{i=1}^{k} \lambda_i I_{\tilde{P}_i^j}(T;X_q).$$

Taking the sum over all observed features $(q_1,\ldots q_n)$, we get

$$\sum_q I_{P_\lambda}(T;X_q) \le \sum_{i=1}^{k} \lambda_i \sum_q I_{\tilde{P}_i^j}(T;X_q),$$

and hence at least one of the distributions $\tilde{P}_1^j,\ldots,\tilde{P}_k^j$ has the same or higher I_{ob} then P_λ. In other words, we can replace the soft clustering of any instance j by a hard clustering without decreasing I_{ob}.

A.3 Maximum Likelihood Mixture Model and IobMax

In the proof of Theorem 7 we claimed that maximizing the likelihood of observed variables is equivalent to maximizing $\sum_{j=1}^{n} I(T;X_j) - I(T;Y)$. In this section we show this based on the work of Elidan and Friedman (2003). For the purpose of better readability in the context of their paper, we use the same notations as in their paper, and review them briefly here. Let Y be a variable that denotes the instance identity, that is, $Y[i] = i$ where $i \in \{1,\ldots,m\}$. Let $Q(Y,\mathbf{X})$ be the empirical distribution

of the features $\mathbf{X}$ in the instances, augmented by the distribution of Y. Let $P(\mathbf{X},T)$ be the maximum likelihood mixture model of the joint distribution $Q(\mathbf{X})$, that is, $P(\mathbf{X},T) = P(T)\prod_j \Pr(x_j|T)$.

From Propositions 4.1, 4.3 in Elidan and Friedman (2003), finding local maxima of the likelihood function is equivalent to minimizing the following Lagrangian

$$\mathcal{L}_{EM} = I_Q(T;Y) - (\mathbf{E}_Q[\log P(\mathbf{X},T)] - \mathbf{E}_Q[\log Q(T)]),$$

as a function of $Q(T|Y)$ and $P(\mathbf{X},T)$. In the stationary point of the EM-algorithm (see Propositions 4.4 and 4.5 in Elidan and Friedman, 2003), $Q(x_j,T) = P(x_j,T)$. Minimizing $\mathcal{L}_{EM}$ is equivalent to minimizing $I(T;Y) - \sum I(T;X_j)$ as shown below:

$$\begin{aligned}
\mathcal{L}_{EM} &= I_Q(T;Y) - (\mathbf{E}_Q[\log P(\mathbf{X},T)] - \mathbf{E}_Q[\log Q(T)]) \\
&= I_Q(T;Y) - \sum_{X,T} Q(\mathbf{X},T)\log\left[P(T)\prod_j P(x_j|T)\right] - H(T) \\
&= I_Q(T;Y) + H(T) - \sum_j \sum_{T,x_j} Q(x_j,T)\log\frac{P(x_j,T)}{P(T)} - H(T) \\
&= I_Q(T;Y) - \sum_j \sum_{T,x_j} Q(x_j,T)\log\frac{P(x_j,T)}{P(x_j)P(T)} + \sum_j \sum_{T,x_j} Q(x_i,T)\log P(x_j) \\
&= I_Q(T;Y) - \sum_j I(T;X_j) + \sum_j H(X_j).
\end{aligned}$$

Since $\sum_j H(X_j)$ is independent of $Q(T|Y)$, and $P(T,Y) = Q(T,Y)$ minimizing $\mathcal{L}_{EM}$ is equivalent to maximizing

$$\sum_j I(T;X_j) - I(T;Y).$$

A.4 Proof of Theorem 11

Before proving Theorem 11, we write generalized definitions of D_{ob}, D_{un} and prove a generalization bound for these generalized definitions (Theorem 14). Then we show that Theorem 11 is a special case of Theorem 14.

The quality of the clustering with respect to a single variable, X_q, is defined by a (weighted) average distance of all pairs of instances within the same cluster (large distance means lower quality). This measure is denoted by D_q which is defined by

$$D_q\{C_1,...,C_k\} = \frac{1}{m}\sum_{r=1}^{k}\frac{1}{|C_r|}\sum_{j,l\in C_r} f(x_q[j],x_q[l]).$$

Using these definitions, we define a generalized observed intra-cluster variable, denoted by $\tilde{D}_{ob}$, as the average of D_q over the observed features within the cluster, that is,

$$\tilde{D}_{ob}\{C_1,...,C_k\} = \frac{1}{n}\sum_{i=1}^{n} D_{q_i}\{C_1,...,C_k\}.$$

When $f(a,b) = \frac{1}{2}(a-b)^2$, we get

$$\begin{aligned} D_{ob}\{C_1,...,C_k\} &= \frac{1}{nm}\sum_{r=1}^{k}\sum_{j\in C_r}\sum_{i=1}^{n}\left(x_{q_i}[j] - \frac{1}{|C_r|}\sum_{l\in C_r} x_{q_i}[l]\right)^2 \\ &= \frac{1}{nm}\sum_{i=1}^{n}\sum_{r=1}^{k}\frac{1}{2|C_r|}\sum_{j\in C_r}\sum_{l\in C_r}\left(x_{q_i}[j] - x_{q_i}[l]\right)^2 \\ &= \frac{1}{nm}\sum_{i=1}^{n}\sum_{r=1}^{k}\frac{1}{|C_r|}\sum_{j\in C_r}\sum_{l\in C_r} f\left(x_{q_i}[j], x_{q_i}[l]\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} D_{q_i}\{C_1,...,C_k\} \\ &= \tilde{D}_{ob}\{C_1,...,C_k\}, \end{aligned} \tag{9}$$

which means that D_{ob} is a special case of $\tilde{D}_{ob}$.

Similarly we define the generalized unobserved intra-cluster variance, denoted by $\tilde{D}_{un}$, as follows:

$$\tilde{D}_{un}\{C_1,...,C_k\} = \mathbf{E}_{q\sim\mathcal{D}}\left\{D_q\{C_1,...,C_k\}\right\}.$$

Again, when $f(a,b) = \frac{1}{2}(a-b)^2$, we get

$$D_{un}\{C_1,...,C_k\} = \tilde{D}_{un}\{C_1,...,C_k\}. \tag{10}$$

Theorem 14 *With the above definitions, for every function, f satisfies Equations 2, 3 and 4 (see Section 3) and for every $\varepsilon > 0$,*

$$\Pr_{\{q_1,...,q_n\}}\left\{\sup_{\alpha(\{C_1,...,C_k\})\geq\alpha_c}\left|\tilde{D}_{ob}\{C_1,...,C_k\} - \tilde{D}_{un}\{C_1,...,C_k\}\right| > \varepsilon\right\} \leq \frac{2k}{\alpha_c}e^{-n\varepsilon^2/2c^2+\log\frac{c}{\varepsilon}},$$

where α is defined in Definition 10.

Before proving Theorem 14, we introduce the following lemma that is required for the proof.

Lemma 15 *Let $\{Z_1,...,Z_S\}$ be a set of jointly S distributed random binary variables, where $z_i \in \{0,1\}$. If $\Pr(z_i = 1) \leq \delta$ for every i then for any $N_b \geq 1$*

$$\Pr\left\{\sum_{i=1}^{S} z_i \geq N_b\right\} \leq \frac{S}{N_b}\delta.$$

Lemma 15 follows directly from Markov's inequality.

The proof of Theorem 14 (given below) is based on the observation that $\tilde{D}_{ob}$ and $\tilde{D}_{un}$ are the weighted average of distance functions over a subset of the pairs of instances. This subset includes only pairs that are within the same cluster. In other words, the calculated inter-cluster variances of a clustering is based on the weighted average of $\frac{1}{2}\sum_{r=1}^{k}|C_r|(|C_r|-1)$ pairs out of the $\frac{1}{2}m(m-1)$ pairs of instances. We define "bad pairs" as pairs of instances with a large difference between the observed and unobserved distances. We use Hoeffding's inequality and Lemma 15 to bound the

probability that we have a large number of "bad pairs". Then we show that if the number of "bad pairs" is small, all clusterings are "good", that is, $\left|\tilde{D}_{ob} - \tilde{D}_{un}\right| \leq \varepsilon$.

Proof of Theorem 14

For each pair of instances j,l $(l > j)$ we define a random variable d_{jl} as the difference between the average observed distance and the expected distance,

$$d_{jl} = \left| \frac{1}{n} \sum_{i=1}^{n} f\left(x_{q_i}[j], x_{q_i}[l]\right) - \mathbf{E}_q \left\{ f\left(x_q[j], x_q[l]\right) \right\} \right|.$$

We also define a binary random variable Z_{jl} $(l > j)$ by

$$z_{jl} = \begin{cases} 1 & \text{if } d_{jl} > \tilde{\varepsilon} \\ 0 & \text{otherwise} \end{cases}$$

where $\tilde{\varepsilon}$ is a positive constant. In other words, z_{jl} is one for "bad" pairs, that is, pairs that have a large difference between the average observed distance and the expected distance. From Hoeffding's inequality,

$$\Pr_{\{q_1,\ldots,q_n\}} \left(z_{jl} = 1\right) \leq \tilde{\delta}, \tag{11}$$

where

$$\tilde{\delta} = 2e^{-2n\tilde{\varepsilon}^2/c^2}. \tag{12}$$

We have $\frac{1}{2}m(m-1)$ of these random binary variables. Let N_{bad} be the number of "bad" pairs, that is, $N_{bad} = \sum_{j,l:l>j} z_{jl}$. First we calculate an upper bound on $\left|\tilde{D}_{ob} - \tilde{D}_{un}\right|$ as a function of N_{bad}, and later we prove an upper bound on the probability of large N_{bad}.

By definition of $\tilde{D}_{ob}, \tilde{D}_{un}, d_{jl}$ and the properties of f (Equations 3 and 4) we can bound the difference between $\tilde{D}_{ob}$ and $\tilde{D}_{un}$ (for any clustering) as follows

$$\begin{aligned} &\left|\tilde{D}_{ob}\left\{C_1,\ldots,C_k\right\} - \tilde{D}_{un}\left\{C_1,\ldots,C_k\right\}\right| \\ &= \left| \frac{1}{mn} \sum_{i=1}^{n} \sum_{r=1}^{k} \frac{1}{|C_r|} \sum_{j,l \in r} f\left(x_{q_i}[j], x_{q_i}[l]\right) - \frac{1}{m} \sum_{i=1}^{k} \frac{1}{|C_r|} \sum_{j,l \in C_r} \mathbf{E}_q \left\{ f\left(x_q[j], x_q[l]\right) \right\} \right| \\ &\leq \frac{2}{m} \sum_{r=1}^{k} \frac{1}{|C_r|} \sum_{j,l \in C_k : l > j} d_{jl}. \end{aligned}$$

By defining ε_d as the following function of the clustering

$$\varepsilon_d\left(\left\{C_1,\ldots,C_k\right\}\right) = \frac{2}{m} \sum_{r=1}^{k} \frac{1}{|C_r|} \sum_{j,l \in C_k : l > j} d_{jl},$$

we have

$$\left|\tilde{D}_{ob}\left\{C_1,\ldots,C_k\right\} - \tilde{D}_{un}\left\{C_1,\ldots,C_k\right\}\right| \leq \varepsilon_d. \tag{13}$$

Recall that r_t is the number of instances in cluster t. Now we calculate an upper bound on ε_d as a function of $\tilde{\varepsilon}$ and N_{bad}. The total number of pairs in the rth cluster is $\frac{1}{2}|C_r|\left(|C_r| - 1\right)$. We have N_{bad}

pairs with a difference above $\tilde{\varepsilon}$ (but not more than c, since f is bounded). The error of each of the other pairs is upper bounded by $\tilde{\varepsilon}$. Hence we get

$$\begin{aligned}\varepsilon_d &\leq \frac{2}{m}\sum_{r=1}^{k}\frac{1}{|C_r|}\sum_{j,l\in C_r:l>j}\left(\tilde{\varepsilon}+z_{jl}\left(c-\tilde{\varepsilon}\right)\right)\\ &\leq \frac{2}{m}\left(\frac{1}{2}\sum_{r=1}^{k}\frac{1}{|C_r|}|C_r|\left(|C_r|-1\right)\tilde{\varepsilon}+\frac{N_{bad}\left(c-\tilde{\varepsilon}\right)}{\min_r|C_r|}\right)\\ &\leq \frac{2}{m}\left(\frac{1}{2}\sum_{r=1}^{k}|C_r|\tilde{\varepsilon}+\frac{N_{bad}c}{\min_r|C_r|}\right).\end{aligned}$$

Note that $\sum_r |C_r| = m$ (the sum of the size of all clusters is m). Hence,

$$\varepsilon_d \leq \tilde{\varepsilon}+\frac{2N_{bad}}{m\min_r|C_r|}c. \tag{14}$$

Let N_b be defined as follows

$$N_b = \frac{m\min_r|C_r|\tilde{\varepsilon}}{2c}. \tag{15}$$

If $N_{bad} \leq N_b$ then $\varepsilon_d \leq 2\tilde{\varepsilon}$ (from Equations 14 and 15). Hence,

$$\Pr_{\{q_1,\ldots,q_n\}}\left\{\varepsilon_d > 2\tilde{\varepsilon}\right\} = \Pr_{\{q_1,\ldots,q_n\}}\left\{N_{bad} > N_b\right\}. \tag{16}$$

From Lemma 15 and Equation 11 for any $N_b \geq 1$

$$\Pr_{\{q_1,\ldots,q_n\}}\left\{N_{bad} \geq N_b\right\} \leq \frac{m(m-1)}{2N_b}\tilde{\delta} \leq \frac{m^2}{2N_b}\tilde{\delta}. \tag{17}$$

Combining Equations 15, 16, 17 and the definition of $\tilde{\delta}$ (Equation 12) we get

$$\Pr_{\{q_1,\ldots,q_n\}}\left\{\varepsilon_d > 2\tilde{\varepsilon}\right\} \leq \frac{mc}{\min_r|C_r|\tilde{\varepsilon}}\tilde{\delta} = \frac{2mc}{\min_r|C_r|\tilde{\varepsilon}}e^{-2n\tilde{\varepsilon}^2/c^2}. \tag{18}$$

By selecting $\varepsilon = \tilde{\varepsilon}/2$ and using Equation 13 and 18 we get

$$\Pr_{\{q_1,\ldots,q_n\}}\left\{\sup_{\alpha(\{C_1,\ldots,C_k\})\geq\alpha_c}\left|\tilde{D}_{ob}\left\{C_1,\ldots,C_k\right\}-\tilde{D}_{un}\left\{C_1,\ldots,C_k\right\}\right| > \varepsilon\right\} \leq 4\frac{m}{\min_r|C_r|}e^{-n\varepsilon^2/2c^2+\log\frac{c}{\varepsilon}}. \tag{19}$$

Using the definition of α we have $m/\min_r|C_r| \leq k/\alpha_c$. Together with Equation 19 we get Theorem 14.

■

Now we are ready to prove Theorem 11 by showing it is a special case of Theorem 14.

Proof of Theorem 11

Let $f(a,b) = \frac{1}{2}(a-b)^2$. In this case f satisfies Equation 2, 3 and 4 where $c = 2R^2$ (since $0 \leq f(x_q[j],x_q[l]) \leq 2R^2$ for all q, j, l). In addition, $\tilde{D}_{ob} = D_{ob}$ and $\tilde{D}_{un} = D_{un}$ (Equations 9 and 10). Therefore, Theorem 11 is a special case of Theorem 14.

■

Appendix B. Notation Table

The following table summaries the notation and definitions used in the paper for quick reference.

Notation	Short Description
m	Number of instances
L	Number of features (both observed and unobserved)
$\{X_1,\ldots,X_L\}$	Random variables (features)
n	Number of observed features
$\tilde{\mathbf{q}}=(q_1,\ldots,q_n)$	Indices of observed features
$\mathcal{D}$	Probability distribution of selecting features
$\{\mathbf{x}[1],\ldots,\mathbf{x}[m]\}$	Instances (each instance is a vector of L elements, where n are observed)
$x_q[j]$	The qth feature of the jth instance
$x_{q_i}[j]$	The ith observed feature of the jth instance
k	Number of clusters
$\mathbf{t}:[m]\rightarrow[k]$	Function that maps instances to clusters
$\{C_1,\ldots,C_k\}$	Clusters (C_r is the set of instances in the rth cluster)
$\lvert C_r\rvert$	Size of rth cluster
T	A variable that represents the cluster label
Z	A random variable taking values uniformly from $\{1,2,\ldots,m\}$ (Random selection of instance index)
s	Upper bound on the number of values a discrete feature can have
I_{ob}	Average observed information (Definition 1)
I_{un}	Expected unobserved information (Definition 1)
$I^*_{un,k}$	Maximum possible value of I_{un} for k clusters (Definition 4)
$I^*_{ob,k}$	Value of I_{ob} for clustering with maximum I_{un} (Definition 4)
$\tilde{I}_{un,k}$	Value of I_{un} for clustering that achieves $\tilde{I}_{ob,k}$ (Definition 5)
$\tilde{I}_{ob,k}$	Maximum possible value of I_{ob} for k clusters (Definition 5)
$f(\cdot,\cdot)$	Distance function between two feature values
c	Constant - upper bound on the distance function
$D_{ob}\{C_1,\ldots,C_k\}$	Observed intra-cluster variance (Definition 8)
$D_{un}\{C_1,\ldots,C_k\}$	Expected unobserved intra-cluster variance (Definition 9)
D^{opt}_{un}	Minimum possible intra-cluster variance (Theorem 12)
$\alpha(\{C_1,\ldots,C_k\})$	Ratio between smallest to average cluster size (Definition 10)

References

R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6, 2005.

J. Blitzer, A. Globerson, and F. Pereira. Distributed latent variable models of lexical co-occurrences. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2005.

J.S. Breese, D. Heckerman, and K. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 1998.

R. Caruana and V. R. de Sa. Promoting poor features to supervisors: Some inputs work better as outputs. In *Advances in Neural Information Processing Systems (NIPS)*, 1997.

G. Chechik and N. Tishby. Extracting relevant structures with side information. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

T. M. Cover and J. A. Thomas. *Elements Of Information Theory*. Wiley Interscience, 1991.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 1977.

G. Elidan and N. Friedman. The information bottleneck EM algorithm. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

N. Friedman, O. Mosenzon, N. Slonim, and N. Tishby. Multivariate information bottleneck. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2001.

A. Globerson and S. Roweis. Nightmare at test time: robust learning by feature deletion. In *International Conference on Machine Learning (ICML)*, 2006.

D. Gondek and T. Hofmann. Non-redundant data clustering. In *Fourth IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, 2004.

T. S. Han. Nonnegative entropy measures of multivariate symmetric correlations. *Information and Control*, 36:133–156, 1978.

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31 (3):264–323, September 1999.

J. Kleinberg. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

E. Krupka and N. Tishby. Generalization in clustering with unobserved features. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

E. Krupka and N. Tishby. Incorporating prior knowledge on features into learnning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

E. Krupka, A. Navot, and N. Tishby. Learning to select features using their properties. *Journal of Machine Learning Research*, submitted.

K. Lang. Learning to filter netnews. *Proc. 12th International Conf. on Machine Learning*, pages 331–339, 1995.

S. P. Lloyd. Least squares quantization in pcm. Technical report, Bell Laboratories, 1957. Published in 1982 in IEEE Transactions on Information Theory 28, 128-137.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

B. Marlin. Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, 2004.

L. Paninski. Estimation of entropy and mutual information. *Neural Computation*, 15:1101–1253, 2003.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

M. Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2002.

M. Szummer and T. Jaakkola. Information regularization with partially labeled data. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. *Proc. 37th Allerton Conf. on Communication and Computation*, 1999.

L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence.*, 1998.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

U. von Luxburg and S. Ben-David. Towards a statistical theory of clustering. In *PASCAL Workshop on Statistics and Optimization of Clustering*, 2005.

Journal of Machine Learning Research 9 (2008) 371-421 Submitted 8/07; Published 3/08

A Tutorial on Conformal Prediction

Glenn Shafer[*] GSHAFER@RUTGERS.EDU
Department of Accounting and Information Systems
Rutgers Business School
180 University Avenue
Newark, NJ 07102, USA

Vladimir Vovk VOVK@CS.RHUL.AC.UK
Computer Learning Research Centre
Department of Computer Science
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK

Editor: Sanjoy Dasgupta

Abstract

Conformal prediction uses past experience to determine precise levels of confidence in new predictions. Given an error probability ε, together with a method that makes a prediction $\hat{y}$ of a label y, it produces a set of labels, typically containing $\hat{y}$, that also contains y with probability $1-\varepsilon$. Conformal prediction can be applied to any method for producing $\hat{y}$: a nearest-neighbor method, a support-vector machine, ridge regression, etc.

Conformal prediction is designed for an on-line setting in which labels are predicted successively, each one being revealed before the next is predicted. The most novel and valuable feature of conformal prediction is that if the successive examples are sampled independently from the same distribution, then the successive predictions will be right $1-\varepsilon$ of the time, even though they are based on an accumulating data set rather than on independent data sets.

In addition to the model under which successive examples are sampled independently, other on-line compression models can also use conformal prediction. The widely used Gaussian linear model is one of these.

This tutorial presents a self-contained account of the theory of conformal prediction and works through several numerical examples. A more comprehensive treatment of the topic is provided in *Algorithmic Learning in a Random World*, by Vladimir Vovk, Alex Gammerman, and Glenn Shafer (Springer, 2005).

Keywords: confidence, on-line compression modeling, on-line learning, prediction regions

1. Introduction

How good is your prediction $\hat{y}$? If you are predicting the label y of a new object, how confident are you that $y = \hat{y}$? If the label y is a number, how close do you think it is to $\hat{y}$? In machine learning, these questions are usually answered in a fairly rough way from past experience. We expect new predictions to fare about as well as past predictions.

Conformal prediction uses past experience to determine precise levels of confidence in predictions. Given a method for making a prediction $\hat{y}$, conformal prediction produces a 95% *prediction*

∗. Also at Computer Learning Research Centre, Royal Holloway.

region—a set $\Gamma^{0.05}$ that contains y with probability at least 95%. Typically $\Gamma^{0.05}$ also contains the prediction $\hat{y}$. We call $\hat{y}$ the *point prediction*, and we call $\Gamma^{0.05}$ the *region prediction*. In the case of regression, where y is a number, $\Gamma^{0.05}$ is typically an interval around $\hat{y}$. In the case of classification, where y has a limited number of possible values, $\Gamma^{0.05}$ may consist of a few of these values or, in the ideal case, just one.

Conformal prediction can be used with any method of point prediction for classification or regression, including support-vector machines, decision trees, boosting, neural networks, and Bayesian prediction. Starting from the method for point prediction, we construct a *nonconformity measure*, which measures how unusual an example looks relative to previous examples, and the *conformal algorithm* turns this nonconformity measure into prediction regions.

Given a nonconformity measure, the conformal algorithm produces a prediction region Γ^ε for every probability of error ε. The region Γ^ε is a $(1-\varepsilon)$-*prediction region*; it contains y with probability at least $1-\varepsilon$. The regions for different ε are nested: when $\varepsilon_1 \geq \varepsilon_2$, so that $1-\varepsilon_1$ is a lower level of confidence than $1-\varepsilon_2$, we have $\Gamma^{\varepsilon_1} \subseteq \Gamma^{\varepsilon_2}$. If Γ^ε contains only a single label (the ideal outcome in the case of classification), we may ask how small ε can be made before we must enlarge Γ^ε by adding a second label; the corresponding value of $1-\varepsilon$ is the confidence we assert in the predicted label.

As we explain in §4, the conformal algorithm is designed for an on-line setting, in which we predict the labels of objects successively, seeing each label after we have predicted it and before we predict the next one. Our prediction $\hat{y}_n$ of the nth label y_n may use observed features x_n of the nth object and the preceding examples $(x_1,y_1),\ldots,(x_{n-1},y_{n-1})$. The size of the prediction region Γ^ε may also depend on these details. Readers most interested in implementing the conformal algorithm may wish to look first at the elementary examples in §4.2.1 and §4.3.1 and then turn back to the earlier more general material as needed.

As we explain in §2, the on-line picture leads to a new concept of validity for prediction with confidence. Classically, a method for finding 95% prediction regions was considered valid if it had a 95% probability of containing the label predicted, because by the law of large numbers it would then be correct 95% of the time when repeatedly applied to independent data sets. But in the on-line picture, we repeatedly apply a method not to independent data sets but to an accumulating data set. After using $(x_1,y_1),\ldots,(x_{n-1},y_{n-1})$ and x_n to predict y_n, we use $(x_1,y_1),\ldots,(x_{n-1},y_{n-1}),(x_n,y_n)$ and x_{n+1} to predict y_{n+1}, and so on. For a 95% on-line method to be valid, 95% of these predictions must be correct. Under minimal assumptions, conformal prediction is valid in this new and powerful sense.

One setting where conformal prediction is valid in the new on-line sense is the one in which the examples (x_i,y_i) are sampled independently from a constant population—that is, from a fixed but unknown probability distribution Q. It is also valid under the slightly weaker assumption that the examples are probabilistically *exchangeable* (see §3) and under other on-line compression models, including the widely used Gaussian linear model (see §5). The validity of conformal prediction under these models is demonstrated in Appendix A.

In addition to the validity of a method for producing 95% prediction regions, we are also interested in its efficiency. It is efficient if the prediction region is usually relatively small and therefore informative. In classification, we would like to see a 95% prediction region so small that it contains only the single predicted label $\hat{y}_n$. In regression, we would like to see a very narrow interval around the predicted number $\hat{y}_n$.

The claim of 95% confidence for a 95% conformal prediction region is valid under exchangeability, no matter what the probability distribution the examples follow and no matter what nonconformity measure is used to construct the conformal prediction region. But the efficiency of conformal prediction will depend on the probability distribution and the nonconformity measure. If we think we know the probability distribution, we may choose a nonconformity measure that will be efficient if we are right. If we have prior probabilities for Q, we may use these prior probabilities to construct a point predictor $\hat{y}_n$ and a nonconformity measure. In the regression case, we might use as $\hat{y}_n$ the mean of the posterior distribution for y_n given the first $n-1$ examples and x_n; in the classification case, we might use the label with the greatest posterior probability. This strategy of first guaranteeing validity under a relatively weak assumption and then seeking efficiency under stronger assumptions conforms to advice long given by John Tukey and others (Tukey, 1986; Small et al., 2006).

Conformal prediction is studied in detail in *Algorithmic Learning in a Random World*, by Vovk, Gammerman, and Shafer (2005). A recent exposition by Gammerman and Vovk (2007) emphasizes connections with the theory of randomness, Bayesian methods, and induction. In this article we emphasize the on-line concept of validity, the meaning of exchangeability, and the generalization to other on-line compression models. We leave aside many important topics that are treated in *Algorithmic Learning in a Random World*, including extensions beyond the on-line picture.

2. Valid Prediction Regions

Our concept of validity is consistent with a tradition that can be traced back to Jerzy Neyman's introduction of confidence intervals for parameters (Neyman, 1937) and even to work by Laplace and others in the late 18th century. But the shift of emphasis to prediction (from estimation of parameters) and to the on-line setting (where our prediction rule is repeatedly updated) involves some rearrangement of the furniture.

The most important novelty in conformal prediction is that its successive errors are probabilistically independent. This allows us to interpret "being right 95% of the time" in an unusually direct way. In §2.1, we illustrate this point with a well-worn example, normally distributed random variables.

In §2.2, we contrast confidence with full-fledged conditional probability. This contrast has been the topic of endless debate between those who find confidence methods informative (classical statisticians) and those who insist that full-fledged probabilities based on all one's information are always preferable, even if the only available probabilities are very subjective (Bayesians). Because the debate usually focuses on estimating parameters rather than predicting future observations, and because some readers may be unaware of the debate, we take the time to explain that we find the concept of confidence useful for prediction in spite of its limitations.

2.1 An Example of Valid On-Line Prediction

A 95% prediction region is *valid* if it contains the truth 95% of the time. To make this more precise, we must specify the set of repetitions envisioned. In the on-line picture, these are successive predictions based on accumulating information. We make one prediction after another, always knowing the outcome of the preceding predictions.

To make clear what validity means and how it can be obtained in this on-line picture, we consider prediction under an assumption often made in a first course in statistics:

> Random variables $z_1, z_2, \ldots$ are independently drawn from a normal distribution with unknown mean and variance.

Prediction under this assumption was discussed in 1935 by R. A. Fisher, who explained how to give a 95% prediction interval for z_n based on $z_1, \ldots, z_{n-1}$ that is valid in our sense. We will state Fisher's prediction rule, illustrate its application to data, and explain why it is valid in the on-line setting.

As we will see, the predictions given by Fisher's rule are too weak to be interesting from a modern machine-learning perspective. This is not surprising, because we are predicting z_n based on old examples $z_1, \ldots, z_{n-1}$ alone. In general, more precise prediction is possible only in the more favorable but more complicated set-up where we know some features x_n of the new example and can use both x_n and the old examples to predict some other feature y_n. But the simplicity of the set-up where we predict z_n from $z_1, \ldots, z_{n-1}$ alone will help us make the logic of valid prediction clear.

2.1.1 FISHER'S PREDICTION INTERVAL

Suppose we observe the z_i in sequence. After observing z_1 and z_2, we start predicting; for $n = 3, 4, \ldots$, we predict z_n after having seen $z_1, \ldots, z_{n-1}$. The natural point predictor for z_n is the average so far:

$$\bar{z}_{n-1} := \frac{1}{n-1} \sum_{i=1}^{n-1} z_i,$$

but we want to give an interval that will contain z_n 95% of the time. How can we do this? Here is Fisher's answer (1935):

1. In addition to calculating the average $\bar{z}_{n-1}$, calculate

$$s_{n-1}^2 := \frac{1}{n-2} \sum_{i=1}^{n-1} (z_i - \bar{z}_{n-1})^2,$$

   which is sometimes called the sample variance. We can usually assume that it is non-zero.

2. In a table of percentiles for t-distributions, find $t_{n-2}^{0.025}$, the point that the t-distribution with $n-2$ degrees of freedom exceeds exactly 2.5% of the time.

3. Predict that z_n will be in the interval

$$\bar{z}_{n-1} \pm t_{n-2}^{0.025}\, s_{n-1} \sqrt{\frac{n}{n-1}}. \tag{1}$$

Fisher based this procedure on the fact that

$$\frac{z_n - \bar{z}_{n-1}}{s_{n-1}} \sqrt{\frac{n-1}{n}}$$

has the t-distribution with $n-2$ degrees of freedom, which is symmetric about 0. This implies that (1) will contain z_n with probability 95% regardless of the values of the mean and variance.

2.1.2 A NUMERICAL EXAMPLE

We can illustrate (1) using some numbers generated in 1900 by the students of Emanuel Czuber (1851–1925). These numbers are integers, but they theoretically have a binomial distribution and are therefore approximately normally distributed.[1]

Here are Czuber's first 19 numbers, $z_1, \ldots, z_{19}$:

$$17, 20, 10, 17, 12, 15, 19, 22, 17, 19, 14, 22, 18, 17, 13, 12, 18, 15, 17.$$

From them, we calculate

$$\bar{z}_{19} = 16.53, \qquad s_{19} = 3.31.$$

The upper 2.5% point for the t-distribution with 18 degrees of freedom, $t_{18}^{0.025}$, is 2.101. So the prediction interval (1) for z_{20} comes out to $[9.40, 24.13]$.

Taking into account our knowledge that z_{20} will be an integer, we can say that the 95% prediction is that z_{20} will be an integer between 10 and 24, inclusive. This prediction is correct; z_{20} is 16.

2.1.3 ON-LINE VALIDITY

Fisher did not have the on-line picture in mind. He probably had in mind a picture where the formula (1) is used repeatedly but in entirely separate problems. For example, we might conduct many separate experiments that each consists of drawing 100 random numbers from a normal distribution and then predicting a 101st draw using (1). Each experiment might involve a different normal distribution (a different mean and variance), but provided the experiments are independent from each other, the law of large numbers will apply. Each time the probability is 95% that z_{101} will be in the interval, and so this event will happen approximately 95% of the time.

The on-line story may seem more complicated, because the experiment involved in predicting z_{101} from $z_1, \ldots, z_{100}$ is not entirely independent of the experiment involved in predicting, say, z_{105} from $z_1, \ldots, z_{104}$. The 101 random numbers involved in the first experiment are all also involved in the second. But as a master of the analytical geometry of the normal distribution (Fisher, 1925; Efron, 1969), Fisher would have noticed, had he thought about it, that this overlap does not actually matter. As we show in Appendix A.3, the events

$$\bar{z}_{n-1} - t_{n-2}^{0.025}\, s_{n-1} \sqrt{\frac{n}{n-1}} \le z_n \le \bar{z}_{n-1} + t_{n-2}^{0.025}\, s_{n-1} \sqrt{\frac{n}{n-1}} \tag{2}$$

for successive n are probabilistically independent in spite of the overlap. Because of this independence, the law of large numbers again applies. Knowing each event has probability 95%, we can conclude that approximately 95% of them will happen. We call the events (2) *hits*.

The prediction interval (1) generalizes to linear regression with normally distributed errors, and on-line hits remain independent in this general setting. Even though formulas for these linear-regression prediction intervals appear in textbooks, the independence of their on-line hits was not noted prior to our work on conformal prediction. Like Fisher, the textbook authors did not have the

1. Czuber's students randomly drew balls from an urn containing six balls, numbered 1 to 6. Each time they drew a ball, they noted its label and put it back in the urn. After each 100 draws, they recorded the number of times that the ball labeled with a 1 was drawn (Czuber, 1914, pp. 329–335). This should have a binomial distribution with parameters 100 and $1/6$, and it is therefore approximately normal with mean $100/6 = 16.67$ and standard deviation $\sqrt{500/36} = 3.73$.

on-line setting in mind. They imagined just one prediction being made in each case where data is accumulated.

We will return to the generalization to linear regression in §5.3.2. There we will derive the textbook intervals as conformal prediction regions within the *on-line Gaussian linear model*, an on-line compression model that uses slightly weaker assumptions than the classical assumption of independent and normally distributed errors.

2.2 Confidence Says Less than Probability.

Neyman's notion of confidence looks at a procedure before observations are made. Before any of the z_i are observed, the event (2) involves multiple uncertainties: $\bar{z}_{n-1}$, s_{n-1}, and z_n are all uncertain. The probability that these three quantities will turn out so that (2) holds is 95%.

We might ask for more than this. It is after we observe the first $n-1$ examples that we calculate $\bar{z}_{n-1}$ and s_{n-1} and then calculate the interval (1), and we would like to be able to say at this point that there is still a 95% probability that z_n will be in (1). But this, it seems, is asking for too much. The assumptions we have made are insufficient to enable us to find a numerical probability for (2) that will be valid at this late date. In theory there is a conditional probability for (2) given $z_1,\ldots,z_{n-1}$, but it involves the unknown mean and variance of the normal distribution.

Perhaps the matter is best understood from the game-theoretic point of view. A probability can be thought of as an offer to bet. A 95% probability, for example, is an offer to take either side of a bet at 19 to 1 odds. The probability is valid if the offer does not put the person making it at a disadvantage, inasmuch as a long sequence of equally reasonable offers will not allow an opponent to multiply the capital he or she risks by a large factor (Shafer and Vovk, 2001). When we assume a probability model (such as the normal model we just used or the on-line compression models we will study later), we are assuming that the model's probabilities are valid in this sense before any examples are observed. Matters may be different afterwards.

In general, a 95% conformal predictor is a rule for using the preceding examples $(x_1,y_1),\ldots,(x_{n-1},y_{n-1})$ and a new object x_n to give a set, say

$$\Gamma^{0.05}((x_1,y_1),\ldots,(x_{n-1},y_{n-1}),x_n), \tag{3}$$

that we predict will contain y_n. If the predictor is valid, the prediction

$$y_n \in \Gamma^{0.05}((x_1,y_1),\ldots,(x_{n-1},y_{n-1}),x_n)$$

will have a 95% probability before any of the examples are observed, and it will be safe, at that point, to offer 19 to 1 odds on it. But after we observe $(x_1,y_1),\ldots,(x_{n-1},y_{n-1})$ and x_n and calculate the set (3), we may want to withdraw the offer.

Particularly striking instances of this phenomenon can arise in the case of classification, where there are only finitely many possible labels. We will see one such instance in §4.3.1, where we consider a classification problem in which there are only two possible labels, s and v. In this case, there are only four possibilities for the prediction region:

1. $\Gamma^{0.05}((x_1,y_1),\ldots,(x_{n-1},y_{n-1}),x_n)$ contains only s.
2. $\Gamma^{0.05}((x_1,y_1),\ldots,(x_{n-1},y_{n-1}),x_n)$ contains only v.
3. $\Gamma^{0.05}((x_1,y_1),\ldots,(x_{n-1},y_{n-1}),x_n)$ contains both s and v.

William S. Gossett
1876–1937

Ronald A. Fisher
1890–1962

Jerzy Neyman
1894–1981

Figure 1: **Three influential statisticians.** Gossett, who worked as a statistician for the Guinness brewery in Dublin, introduced the t-distribution to English-speaking statisticians (Student, 1908). Fisher, whose applied and theoretical work invigorated mathematical statistics in the 1920s and 1930s, refined, promoted, and extended Gossett's work. Neyman was one of the most influential leaders in the subsequent movement to use advanced probability theory to give statistics a firmer foundation and further extend its applications.

4. $\Gamma^{0.05}((x_1,y_1),\ldots,(x_{n-1},y_{n-1}),x_n)$ is empty.

The third and fourth cases can occur even though $\Gamma^{0.05}$ is valid. When the third case happens, the prediction, though uninformative, is certain to be correct. When the fourth case happens, the prediction is clearly wrong. These cases are consistent with the prediction being right 95% of the time. But when we see them arise, we know whether the particular value of n is one of the 95% where we are right or the one of the 5% where we are wrong, and so the 95% will not remain valid as a probability defining betting odds.

In the case of normally distributed examples, Fisher called the 95% probability for z_n being in the interval (1) a "fiducial probability," and he seems to have believed that it would not be susceptible to a gambling opponent who knows the first $n-1$ examples (see Fisher, 1973, pp. 119–125). But this turned out not to be the case (Robinson, 1975). For this and related reasons, most scientists who use Fisher's methods have adopted the interpretation offered by Neyman, who wrote about "confidence" rather than fiducial probability and emphasized that a confidence level is a full-fledged probability only before we acquire data. It is the procedure or method, not the interval or region it produces when applied to particular data, that has a 95% probability of being correct.

Neyman's concept of confidence has endured in spite of its shortcomings. It is widely taught and used in almost every branch of science. Perhaps it is especially useful in the on-line setting. It is useful to know that 95% of our predictions are correct even if we cannot assert a full-fledged 95% probability for each prediction when we make it.

3. Exchangeability

Consider variables $z_1,\ldots,z_N$. Suppose that for any collection of N values, the $N!$ different orderings are equally likely. Then we say that $z_1,\ldots,z_N$ are *exchangeable*. The assumption that $z_1,\ldots,z_N$ are exchangeable is slightly weaker than the more familiar assumption that they are drawn independently from a probability distribution.

In our book (Vovk et al., 2005), conformal prediction is first explained under the assumption that $z_1, \ldots, z_N$ are independently drawn from a probability distribution (or that they are "random," as we say there), and then it is pointed out that this assumption can be relaxed to the assumption that $z_1, \ldots, z_N$ are exchangeable. When we introduce conformal prediction in this article, in §4, we assume only exchangeability from the outset, hoping that this will make the logic of the method as clear as possible. Once this logic is clear, it is easy to see that it works not only for the exchangeability model but also for other on-line compression models (§5).

In this section we look at the relationship between exchangeability and independence and then give a backward-looking definition of exchangeability that can be understood game-theoretically. We conclude with a law of large numbers for exchangeable sequences, which will provide the basis for our confidence that our 95% prediction regions are right 95% of the time.

3.1 Exchangeability and Independence

Although the definition of exchangeability we just gave may be clear enough at an intuitive level, it has two technical problems that make it inadequate as a formal mathematical definition: (1) in the case of continuous distributions, any specific values for $z_1, \ldots, z_N$ will have probability zero, and (2) in the case of discrete distributions, two or more of the z_i might take the same value, and so a list of possible values $a_1, \ldots, a_N$ might contain fewer than n distinct values.

One way of avoiding these technicalities is to use the concept of a permutation, as follows:

> *Definition of exchangeability using permutations.* The variables $z_1, \ldots, z_N$ are exchangeable if for every permutation τ of the integers $1, \ldots, N$, the variables $w_1, \ldots, w_N$, where $w_i = z_{\tau(i)}$, have the same joint probability distribution as $z_1, \ldots, z_N$.

We can extend this to a definition of exchangeability for an infinite sequence of variables: $z_1, z_2, \ldots$ are exchangeable if $z_1, \ldots, z_N$ are exchangeable for every N.

This definition makes it easy to see that independent and identically distributed random variables are exchangeable. Suppose $z_1, \ldots, z_N$ all take values from the same example space $\mathbf{Z}$, all have the same probability distribution Q, and are independent. Then their joint distribution satisfies

$$\Pr(z_1 \in A_1 \ \& \ \ldots \ \& \ z_N \in A_N) = Q(A_1) \cdots Q(A_N)$$

for any[2] subsets $A_1, \ldots, A_N$ of $\mathbf{Z}$, where $Q(A)$ is the probability Q assigns to an example being in A. Because permuting the factors $Q(A_n)$ does not change their product, and because a joint probability distribution for $z_1, \ldots, z_N$ is determined by the probabilities it assigns to events of the form $\{z_1 \in A_1 \ \& \ \ldots \ \& \ z_N \in A_N\}$, this makes it clear that $z_1, \ldots, z_N$ are exchangeable.

Exchangeability implies that variables have the same distribution. On the other hand, exchangeable variables need not be independent. Indeed, when we average two or more distinct joint probability distributions under which variables are independent, we usually get a joint probability distribution under which they are exchangeable (averaging preserves exchangeability) but not independent (averaging usually does not preserve independence). According to a famous theorem by de Finetti, an exchangeable joint distribution for an infinite sequence of distinct variables is exchangeable only if it is a mixture of joint distributions under which the variables are independent (Hewitt and Savage, 1955). As Table 1 shows, the picture is more complicated in the finite case.

2. We leave aside technicalities involving measurability.

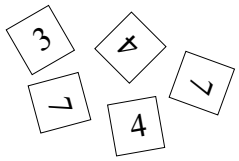
$\Pr(z_1 = \text{H} \,\&\, z_2 = \text{H})$	$\Pr(z_1 = \text{H} \,\&\, z_2 = \text{T})$
$\Pr(z_1 = \text{T} \,\&\, z_2 = \text{H})$	$\Pr(z_1 = \text{T} \,\&\, z_2 = \text{T})$

0.81	0.09
0.09	0.01

0.41	0.09
0.09	0.41

0.10	0.40
0.40	0.10

Table 1: **Examples of exchangeability.** We consider variables z_1 and z_2, each of which comes out H or T. Exchangeability requires only that $\Pr(z_1 = \text{H} \,\&\, z_2 = \text{T}) = \Pr(z_1 = \text{T} \,\&\, z_2 = \text{H})$. Three examples of distributions for z_1 and z_2 with this property are shown. On the left, z_1 and z_2 are independent and identically distributed; both come out H with probability 0.9. The middle example is obtained by averaging this distribution with the distribution in which the two variables are again independent and identically distributed but T's probability is 0.9. The distribution on the right, in contrast, cannot be obtained by averaging distributions under which the variables are independent and identically distributed. Examples of this last type disappear as we ask for a larger and larger number of variables to be exchangeable.



4	3	4	7	7

Figure 2: **Ordering the tiles.** Joe gives Bill a bag containing five tiles, and Bill arranges them to form the list 43477. Bill can calculate conditional probabilities for which z_i had which of the five values. His conditional probability for $z_5 = 4$, for example, is $2/5$. There are $(5!)/(2!)(2!) = 30$ ways of assigning the five values to the five variables; $(z_1, z_2, z_3, z_4, z_5) = (4, 3, 4, 7, 7)$ is one of these, and they all have the same probability, $1/30$.

3.2 Backward-Looking Definitions of Exchangeability

Another way of defining exchangeability looks backwards from a situation where we know the unordered values of $z_1, \ldots, z_N$.

Suppose Joe has observed $z_1, \ldots, z_N$. He writes each value on a tile resembling those used in Scrabble©, puts the N tiles in a bag, shakes the bag, and gives it to Bill to inspect. Bill sees the N values (some possibly equal to each other) without knowing their original order. Bill also knows the joint probability distribution for $z_1, \ldots, z_N$. So he obtains probabilities for the ordering of the tiles by conditioning this joint distribution on his knowledge of the bag. The joint distribution is exchangeable if and only if these conditional probabilities are the same as the probabilities for the result of ordering the tiles by successively drawing them at random from the bag without replacement.

To make this into a definition of exchangeability, we formalize the notion of a bag. A *bag* (or *multiset*, as it is sometimes called) is a collection of elements in which repetition is allowed. It is like a set inasmuch as its elements are unordered but like a list inasmuch as an element can occur more than once. We write $\wr a_1,\dots,a_N\int$ for the bag obtained from the list $a_1,\dots,a_N$ by removing information about the ordering.

Here are three equivalent conditions on the joint distribution of a sequence of random variables $z_1,\dots,z_N$, any of which can be taken as the definition of exchangeability.

1. For any bag B of size N, and for any examples $a_1,\dots,a_N$,

$$\Pr(z_1 = a_1 \,\&\, \dots \,\&\, z_N = a_N \mid \wr z_1,\dots,z_N\int = B)$$

   is equal to the probability that successive random drawings from the bag B without replacement produces first a_N, then a_{N-1}, and so on, until the last element remaining in the bag is a_1.

2. For any n, $1 \le n \le N$, z_n is independent of $z_{n+1},\dots,z_N$ given the bag $\wr z_1,\dots,z_n\int$ and for any bag B of size n,

$$\Pr(z_n = a \mid \wr z_1,\dots,z_n\int = B) = \frac{k}{n}, \tag{4}$$

   where k is the number of times a occurs in B.

3. For any bag B of size N, and for any examples $a_1,\dots,a_N$,

$$\Pr(z_1 = a_1 \,\&\, \dots \,\&\, z_N = a_N \mid \wr z_1,\dots,z_N\int = B) = \begin{cases} \frac{n_1!\cdots n_k!}{N!} & \text{if } B = \wr a_1,\dots,a_N\int \\ 0 & \text{if } B \neq \wr a_1,\dots,a_N\int, \end{cases} \tag{5}$$

   where k is the number of distinct values among the a_i, and $n_1,\dots,n_k$ are the respective numbers of times they occur. (If the a_i are all distinct, the expression $n_1!\cdots n_k!/(N!)$ reduces to $1/(N!)$.)

We leave it to the reader to verify that these three conditions are equivalent to each other. The second condition, which we will emphasize, is represented pictorially in Figure 3.

The backward-looking conditions are also equivalent to the definition of exchangeability using permutations given on p. 378. This equivalence is elementary in the case where every possible sequence of values $a_1,\dots,a_n$ has positive probability. But complications arise when this probability is zero, because the conditional probability on the left-hand side of (5) is then defined only with probability one by the joint distribution. We do not explore these complications here.

3.3 The Betting Interpretation of Exchangeability

The framework for probability developed in Shafer and Vovk (2001) formalizes classical results of probability theory, such as the law of large numbers, as theorems of game theory: a bettor can multiply the capital he risks by a large factor if these results do not hold. This allows us to express the empirical interpretation of given probabilities in terms of betting, using what we call *Cournot's principle*: the odds determined by the probabilities will not allow a bettor to multiply the capital he or she risks by a large factor (Shafer, 2007).

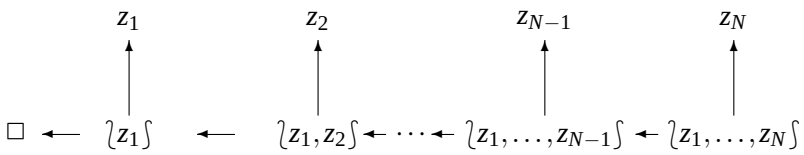


Figure 3: **Backward probabilities, step by step.** The two arrows backwards from each bag $\{z_1, \ldots, z_n\}$ symbolize drawing an example z_n out at random, leaving the smaller bag $\{z_1, \ldots, z_{n-1}\}$. The probabilities for the result of the drawing are given by (4). Readers familiar with Bayes nets (Cowell et al., 1999) will recognize this diagram as an example; conditional on each variable, a joint probability distribution is given for its children (the variables to which arrows from it point), and given the variable, its descendants are independent of its ancestors.

By applying this idea to the sequence of probabilities (4), we obtain a betting interpretation of exchangeability. Think of Joe and Bill as two players in a game that moves backwards from point N in Figure 3. At each step, Joe provides new information and Bill bets. Designate by $\mathcal{K}_N$ the total capital Bill risks. He begins with this capital at N, and at each step n he bets on what z_n will turn out to be. When he bets at step n, he cannot risk losing more than he has at that point (because he is not risking more than $\mathcal{K}_N$ in the whole game), but otherwise he can bet as much as he wants for or against each possible value a for z_n at the odds $(k/n) : (1-k/n)$, where k is the number of elements in the current bag equal to a.

For brevity, we write B_n for the bag $\{z_1, \ldots, z_n\}$, and for simplicity, we set the initial capital K_N equal to \$1. This gives the following protocol:

THE BACKWARD-LOOKING BETTING PROTOCOL

Players: Joe, Bill

$\mathcal{K}_N := 1$.
Joe announces a bag B_N of size N.
FOR $n = N, N-1, \ldots, 2, 1$
    Bill bets on z_n at odds set by (4).
    Joe announces $z_n \in B_n$.
    $\mathcal{K}_{n-1} := \mathcal{K}_n +$ Bill's net gain.
    $B_{n-1} := B_n \setminus \{z_n\}$.

Constraint: Bill must move so that his capital $\mathcal{K}_n$ will be nonnegative for all n no matter how Joe moves.

Our betting interpretation of exchangeability is that Bill will not multiply his initial capital $\mathcal{K}_N$ by a large factor in this game.

The permutation definition of exchangeability does not lead to an equally simple betting interpretation, because the probabilities for $z_1, \ldots, z_N$ to which the permutation definition refers are not determined by the mere assumption of exchangeability.

3.4 A Law of Large Numbers for Exchangeable Sequences

As we noted when we studied Fisher's prediction interval in §2.1.3, the validity of on-line prediction requires more than having a high probability of a hit for each individual prediction. We also need a law of large numbers, so that we can conclude that a high proportion of the high-probability predictions will be correct. As we show in §A.3, the successive hits in the case of Fisher's region predictor are independent, so that the usual law of large numbers applies. What can we say in the case of conformal prediction under exchangeability?

Suppose $z_1,\ldots,z_N$ are exchangeable, drawn from an example space $\mathbf{Z}$. In this context, we adopt the following definitions.

- An event E is an *n-event*, where $1 \le n \le N$, if its happening or failing is determined by the value of z_n and the value of the bag $\{\!|z_1,\ldots,z_{n-1}|\!\}$.
- An n-event E is ε-*rare* if
$$\Pr(E \mid \{\!|z_1,\ldots,z_n|\!\}) \le \varepsilon. \tag{6}$$

The left-hand side of the inequality (6) is a random variable, because the bag $\{\!|z_1,\ldots,z_n|\!\}$ is random. The inequality says that this random variable never exceeds ε.

As we will see in the next section, the successive errors for a conformal predictor are ε-rare n-events. So the validity of conformal prediction follows from the following informal proposition.

Informal Proposition 1 *Suppose N is large, and the variables $z_1,\ldots,z_N$ are exchangeable. Suppose E_n is an ε-rare n-event for $n = 1,\ldots,N$. Then the law of large numbers applies; with very high probability, no more than approximately the fraction ε of the events $E_1,\ldots,E_N$ will happen.*

In Appendix A, we formalize this proposition in two ways: classically and game-theoretically.

The classical approach appeals to the classical weak law of large numbers, which tells us that if $E_1,\ldots,E_N$ are mutually independent and each have probability exactly ε, and N is sufficiently large, then there is a very high probability that the fraction of the events that happen will be close to ε. We show in §A.1 that if (6) holds with equality, then E_n are mutually independent and each of them has unconditional probability ε. Having the inequality instead of equality means that the E_n are even less likely to happen, and this will not reverse the conclusion that few of them will happen.

The game-theoretic approach is more straightforward, because the game-theoretic version law of large numbers does not require independence or exact levels of probability. In the game-theoretic framework, the only question is whether the probabilities specified for successive events are rates at which a bettor can place successive bets. The Backward-Looking Betting Protocol says that this is the case for ε-rare n-events. As Bill moves through the protocol from N to 1, he is allowed to bet against each error E_n at a rate corresponding to its having probability ε or less. So the game-theoretic weak law of large numbers (Shafer and Vovk, 2001, pp. 124–126) applies directly. Because the game-theoretic framework is not well known, we state and prove this law of large numbers, specialized to the Backward-Looking Betting Protocol, in §A.2.

4. Conformal Prediction under Exchangeability

We are now in a position to state the conformal algorithm under exchangeability and explain why it produces valid nested prediction regions.

We distinguish two cases of on-line prediction. In both cases, we observe examples $z_1,\ldots,z_N$ one after the other and repeatedly predict what we will observe next. But in the second case we have more to go on when we make each prediction.

1. *Prediction from old examples alone.* Just before observing z_n, we predict it based on the previous examples, $z_1,\ldots,z_{n-1}$.

2. *Prediction using features of the new object.* Each example z_i consists of an object x_i and a label y_i. In symbols: $z_i = (x_i,y_i)$. We observe in sequence $x_1,y_1,\ldots,x_N,y_N$. Just before observing y_n, we predict it based on what we have observed so far, x_n and the previous examples $z_1,\ldots,z_{n-1}$.

Prediction from old examples may seem relatively uninteresting. It can be considered a special case of prediction using features x_n of new examples—the case in which the x_n provide no information, and this special case we may have too little information to make useful predictions. But its simplicity makes prediction with old examples alone advantageous as a setting for explaining the conformal algorithm, and as we will see, it is then straightforward to take account of the new information x_n.

Conformal prediction requires that we first choose a nonconformity measure, which measures how different a new example is from old examples. In §4.1, we explain how nonconformity measures can be obtained from methods of point prediction. In §4.2, we state and illustrate the conformal algorithm for predicting new examples from old examples alone. In §4.3, we generalize to prediction with the help of features of a new example. In §4.4, we explain why conformal prediction produces the best possible valid nested prediction regions under exchangeability. Finally, in §4.5 we discuss the implications of the failure of the assumption of exchangeability.

For some readers, the simplicity of the conformal algorithm may be obscured by its generality and the scope of our preliminary discussion of nonconformity measures. We encourage such readers to look first at §4.2.1, §4.3.1, and §4.3.2, which provide largely self-contained accounts of the algorithm as it applies to some small data sets.

4.1 Nonconformity Measures

The starting point for conformal prediction is what we call a *nonconformity measure*, a real-valued function $A(B,z)$ that measures how different an example z is from the examples in a bag B. The conformal algorithm assumes that a nonconformity measure has been chosen. The algorithm will produce valid nested prediction regions using any real-valued function $A(B,z)$ as the nonconformity measure. But the prediction regions will be efficient (small) only if $A(B,z)$ measures well how different z is from the examples in B.

A method $\hat{z}(B)$ for obtaining a point prediction $\hat{z}$ for a new example from a bag B of old examples usually leads naturally to a nonconformity measure A. In many cases, we only need to add a way of measuring the distance $d(z,z')$ between two examples. Then we define A by

$$A(B,z) := d(\hat{z}(B),z). \tag{7}$$

The prediction regions produced by the conformal algorithm do not change when the nonconformity measure A is transformed monotonically. If A is nonnegative, for example, replacing A with A^2 will make no difference. Consequently, the choice of the distance measure $d(z,z')$ is relatively unimportant. The important step in determining the nonconformity measure A is choosing the point predictor $\hat{z}(B)$.

To be more concrete, suppose the examples are real numbers, and write $\bar{z}_B$ for the average of the numbers in B. If we take this average as our point predictor $\hat{z}(B)$, and we measure the distance between two real numbers by the absolute value of their difference, then (7) becomes

$$A(B,z) := |\bar{z}_B - z|. \tag{8}$$

If we use the median of the numbers in B instead of their average as $\hat{z}(B)$, we get a different nonconformity measure, which will produce different prediction regions when we use the conformal algorithm. On the other hand, as we have already said, it will make no difference if we replace the absolute difference $d(z,z') = |z-z'|$ with the squared difference $d(z,z') = (z-z')^2$, thus squaring A.

We can also vary (8) by including the new example in the average:

$$A(B,z) := |(\text{average of } z \text{ and all the examples in } B) - z|. \tag{9}$$

This results in the same prediction regions as (8), because if B has n elements, then

$$|(\text{average of } z \text{ and all the examples in } B) - z| = \left| \frac{n\bar{z}_B + z}{n+1} - z \right| = \frac{n}{n+1}|\bar{z}_B - z|,$$

and as we have said, conformal prediction regions are not changed by a monotonic transformation of the nonconformity measure. In the numerical example that we give in §4.2.1 below, we use (9) as our nonconformity measure.

When we turn to the case where features of a new object help us predict a new label, we will consider, among others, the following two nonconformity measures:

Distance to the nearest neighbors for classification. Suppose $B = \lbag z_1, \ldots, z_{n-1} \rbag$, where each z_i consists of a number x_i and a nonnumerical label y_i. Again we observe x but not y for a new example $z = (x,y)$. The nearest-neighbor method finds the x_i closest to x and uses its label y_i as our prediction of y. If there are only two labels, or if there is no natural way to measure the distance between labels, we cannot measure how wrong the prediction is; it is simply right or wrong. But it is natural to measure the nonconformity of the new example (x,y) to the old examples (x_i,y_i) by comparing x's distance to old objects with the same label to its distance to old objects with a different label. For example, we can set

$$\begin{aligned} A(B,z) &:= \frac{\min\{|x_i - x| : 1 \le i \le n-1, y_i = y\}}{\min\{|x_i - x| : 1 \le i \le n-1, y_i \ne y\}} \\ &= \frac{\text{distance to } z\text{'s nearest neighbor in } B \text{ with the same label}}{\text{distance to } z\text{'s nearest neighbor in } B \text{ with a different label}}. \end{aligned} \tag{10}$$

Distance to a regression line. Suppose $B = \lbag (x_1,y_1), \ldots, (x_l,y_l) \rbag$, where the x_i and y_i are numbers. The most common way of fitting a line to such pairs of numbers is to calculate the averages

$$\bar{x}_l := \sum_{j=1}^{l} x_j \quad \text{and} \quad \bar{y}_l := \sum_{j=1}^{l} y_j,$$

and then the coefficients

$$b_l = \frac{\sum_{j=1}^{l}(x_j - \bar{x}_l)y_j}{\sum_{j=1}^{l}(x_j - \bar{x}_l)^2} \quad \text{and} \quad a_l = \bar{y}_l - b_l\bar{x}_l.$$

This gives the *least-squares line* $y = a_l + b_l x$. The coefficients a_l and b_l are not affected if we change the order of the z_i; they depend only on the bag B.

If we observe a bag $B = \{z_1, \ldots, z_{n-1}\}$ of examples of the form $z_i = (x_i, y_i)$ and also x but not y for a new example $z = (x, y)$, then the least-squares prediction of y is

$$\hat{y} = a_{n-1} + b_{n-1} x. \tag{11}$$

We can use the error in this prediction as a nonconformity measure:

$$A(B, z) := |y - \hat{y}| = |y - (a_{n-1} + b_{n-1} x)|.$$

We can obtain other nonconformity measures by using other methods to estimate a line.

Alternatively, we can include the new example as one of the examples used to estimate the least squares line or some other regression line. In this case, it is natural to write (x_n, y_n) for the new example. Then a_n and b_n designate the coefficients calculated from all n examples, and we can use

$$|y_i - (a_n + b_n x_i)| \tag{12}$$

to measure the nonconformity of each of the (x_i, y_i) with the others. In general, the inclusion of the new example simplifies the implementation or at least the explanation of the conformal algorithm. In the case of least squares, it does not change the prediction regions.

4.2 Conformal Prediction from Old Examples Alone

Suppose we have chosen a nonconformity measure A for our problem. Given A, and given the assumption that the z_i are exchangeable, we now define a valid prediction region

$$\gamma^{\varepsilon}(z_1, \ldots, z_{n-1}) \subseteq \mathbf{Z},$$

where $\mathbf{Z}$ is the example space. We do this by giving an algorithm for deciding, for each $z \in \mathbf{Z}$, whether z should be included in the region. For simplicity in stating this algorithm, we provisionally use the symbol z_n for z, as if we were assuming that z_n is in fact equal to z.

The Conformal Algorithm Using Old Examples Alone

Input: Nonconformity measure A, significance level ε, examples $z_1, \ldots, z_{n-1}$, example z,

Task: Decide whether to include z in $\gamma^{\varepsilon}(z_1, \ldots, z_{n-1})$.

Algorithm:

1. Provisionally set $z_n := z$.
2. For $i = 1, \ldots, n$, set $\alpha_i := A(\{z_1, \ldots, z_n\} \setminus \{z_i\}, z_i)$.
3. Set $p_z := \dfrac{\text{number of } i \text{ such that } 1 \le i \le n \text{ and } \alpha_i \ge \alpha_n}{n}$.
4. Include z in $\gamma^{\varepsilon}(z_1, \ldots, z_{n-1})$ if and only if $p_z > \varepsilon$.

If $\mathbf{Z}$ has only a few elements, this algorithm can be implemented in a brute-force way: calculate p_z for every $z \in \mathbf{Z}$. If $\mathbf{Z}$ has many elements, we will need some other way of identifying the z satisfying $p_z > \varepsilon$.

The number p_z is the fraction of the examples in $\wr z_1,\dots,z_{n-1},z\int$ that are at least as different from the others as z is, in the sense measured by A. So the algorithm tells us to form a prediction region consisting of the z that are not among the fraction ε most out of place when they are added to the bag of old examples.

The definition of $\gamma^\varepsilon(z_1,\dots,z_{n-1})$ can be framed as an application of the widely accepted Neyman-Pearson theory for hypothesis testing and confidence intervals (Lehmann, 1986). In the Neyman-Pearson theory, we test a hypothesis H using a random variable T that is likely to be large only if H is false. Once we observe $T=t$, we calculate $p_H := \Pr(T \geq t \,|\, H)$. We reject H at level ε if $p_H \leq \varepsilon$. Because this happens under H with probability no more than ε, we can declare $1-\varepsilon$ confidence that the true hypothesis H is among those not rejected. Our procedure makes these choices of H and T:

- The hypothesis H says the bag of the first n examples is $\wr z_1,\dots,z_{n-1},z\int$.
- The test statistic T is the random value of α_n.

Under H—that is, conditional on the bag $\wr z_1,\dots,z_{n-1},z\int$, T is equally likely to come out equal to any of the α_i. Its observed value is α_n. So

$$p_H = \Pr(T \geq \alpha_n \,|\, \wr z_1,\dots,z_{n-1},z\int) = p_z.$$

Since $z_1,\dots,z_{n-1}$ are known, rejecting the bag $\wr z_1,\dots,z_{n-1},z\int$ means rejecting $z_n = z$. So our $1-\varepsilon$ confidence is in the set of z for which $p_z > \varepsilon$.

The regions $\gamma^\varepsilon(z_1,\dots,z_{n-1})$ for successive n are based on overlapping sequences of examples rather than independent samples. But the successive errors are ε-rare n-events. The event that our nth prediction is an error, $z_n \notin \gamma^\varepsilon(z_1,\dots,z_{n-1})$, is the event $p_{z_n} \leq \varepsilon$. This is an n-event, because the value of p_{z_n} is determined by z_n and the bag $\wr z_1,\dots,z_{n-1}\int$. It is ε-rare because it is the event that α_n is among a fraction ε or fewer of the α_i that are strictly larger than all the other α_i, and this can have probability at most ε when the α_i are exchangeable. So it follows from Informal Proposition 1 (§3.4) that we can expect at least $1-\varepsilon$ of the $\gamma^\varepsilon(z_1,\dots,z_{n-1})$, $n=1,\dots,N$, to be correct.

4.2.1 EXAMPLE: PREDICTING A NUMBER WITH AN AVERAGE

In §2.1, we discussed Fisher's 95% prediction interval for z_n based on $z_1,\dots,z_{n-1}$, which is valid under the assumption that the z_i are independent and normally distributed. We used it to predict z_{20} when the first 19 z_i are

$$17,20,10,17,12,15,19,22,17,19,14,22,18,17,13,12,18,15,17.$$

Taking into account our knowledge that the z_i are all integers, we arrived at the 95% prediction that z_{20} is an integer between 10 to 24, inclusive.

What can we predict about z_{20} at the 95% level if we drop the assumption of normality and assume only exchangeability? To produce a 95% prediction interval valid under the exchangeability assumption alone, we reason as follows. To decide whether to include a particular value z in the interval, we consider twenty numbers that depend on z:

- First, the deviation of z from the average of it and the other 19 numbers. Because the sum of the 19 is 314, this is
$$\left|\frac{314+z}{20} - z\right| = \frac{1}{20}\,|314-19z|\,. \tag{13}$$

- Then, for $i = 1, \ldots, 19$, the deviation of z_i from this same average. This is

$$\left| \frac{314 + z}{20} - z_i \right| = \frac{1}{20} \left| 314 + z - 20 z_i \right|. \tag{14}$$

Under the hypothesis that z is the actual value of z_n, these 20 numbers are exchangeable. Each of them is as likely as the other to be the largest. So there is at least a 95% (19 in 20) chance that (13) will not exceed the largest of the 19 numbers in (14). The largest of the 19 z_is being 22 and the smallest 10, we can write this condition as

$$|314 - 19z| \leq \max \left\{ \left| 314 + z - (20 \times 22) \right|, \left| 314 + z - (20 \times 10) \right| \right\},$$

which reduces to

$$10 \leq z \leq \frac{214}{9} \approx 23.8.$$

Taking into account that z_{20} is an integer, our 95% prediction is that it will be an integer between 10 and 23, inclusive. This is nearly the same prediction we obtained by Fisher's method. We have lost nothing by weakening the assumption that the z_i are independent and normally distributed to the assumption that they are exchangeable. But we are still basing our prediction region on the average of old examples, which is an optimal estimator in various respects under the assumption of normality.

4.2.2 ARE WE COMPLICATING THE STORY UNNECESSARILY?

The reader may feel that we are vacillating about whether to include the new example in the bag with which we are comparing it. In our statement of the conformal algorithm, we define the nonconformity scores by

$$\alpha_i := A(\wr z_1, \ldots, z_n \int \setminus \wr z_i \int, z_i), \tag{15}$$

apparently signaling that we do not want to include z_i in the bag to which it is compared. But then we use the nonconformity measure

$$A(B, z) := \left| (\text{average of } z \text{ and all the examples in } B) - z \right|,$$

which seems to put z back in the bag, reducing (15) to

$$\alpha_i = \left| \frac{\sum_{j=1}^{n} z_j}{n} - z_i \right|.$$

We could have reached this point more easily by writing

$$\alpha_i := A(\wr z_1, \ldots, z_n \int, z_i) \tag{16}$$

in the conformal algorithm and using $A(B, z) := \left| \bar{z}_B - z \right|$.

The two ways of defining nonconformity scores, (15) and (16), are equivalent, inasmuch as whatever we can get with one of them we can get from the other by changing the nonconformity measure. In this case, (16) might be more convenient. But we will see other cases where (15) is more convenient. We also have another reason for using (15). It is the form that generalizes, as we will see in §5, to on-line compression models.

4.3 Conformal Prediction Using a New Object

Now we turn to the case where our example space $\mathbf{Z}$ is of the form $\mathbf{Z} = \mathbf{X} \times \mathbf{Y}$. We call $\mathbf{X}$ the *object space*, $\mathbf{Y}$ the *label space*. We observe in sequence examples $z_1, \ldots, z_N$, where $z_i = (x_i, y_i)$. At the point where we have observed

$$(z_1, \ldots, z_{n-1}, x_n) = ((x_1, y_1), \ldots, (x_{n-1}, y_{n-1}), x_n),$$

we want to predict y_n by giving a prediction region

$$\Gamma^\varepsilon(z_1, \ldots, z_{n-1}, x_n) \subseteq \mathbf{Y}$$

that is valid at the $(1-\varepsilon)$ level. As in the special case where the x_i are absent, we start with a nonconformity measure $A(B, z)$.

We define the prediction region by giving an algorithm for deciding, for each $y \in \mathbf{Y}$, whether y should be included in the region. For simplicity in stating this algorithm, we provisionally use the symbol z_n for (x_n, y), as if we were assuming that y_n is in fact equal to y.

The Conformal Algorithm

Input: Nonconformity measure A, significance level ε, examples $z_1, \ldots, z_{n-1}$, object x_n, label y

Task: Decide whether to include y in $\Gamma^\varepsilon(z_1, \ldots, z_{n-1}, x_n)$.

Algorithm:

1. Provisionally set $z_n := (x_n, y)$.
2. For $i = 1, \ldots, n$, set $\alpha_i := A(\{z_1, \ldots, z_n\} \setminus \{z_i\}, z_i)$.
3. Set $p_y := \dfrac{\#\{i = 1, \ldots, n \mid \alpha_i \geq \alpha_n\}}{n}$.
4. Include y in $\Gamma^\varepsilon(z_1, \ldots, z_{n-1}, x_n)$ if and only if $p_y > \varepsilon$.

This differs only slightly from the conformal algorithm using old examples alone (p. 385). Now we write p_y instead of p_z, and we say that we are including y in $\Gamma^\varepsilon(z_1, \ldots, z_{n-1}, x_n)$ instead of saying that we are including z in $\gamma^\varepsilon(z_1, \ldots, z_{n-1})$.

To see that this algorithm produces valid prediction regions, it suffices to see that it consists of the algorithm for old examples alone together with a further step that does not change the frequency of hits. We know that the region the old algorithm produces,

$$\gamma^\varepsilon(z_1, \ldots, z_{n-1}) \subseteq \mathbf{Z}, \tag{17}$$

contains the new example $z_n = (x_n, y_n)$ at least 95% of the time. Once we know x_n, we can rule out all $z = (x, y)$ in (17) with $x \neq x_n$. The y not ruled out, those such that (x_n, y) is in (17), are precisely those in the set

$$\Gamma^\varepsilon(z_1, \ldots, z_{n-1}, x_n) \subseteq \mathbf{Y} \tag{18}$$

produced by our new algorithm. Having (x_n, y_n) in (17) $1-\varepsilon$ of the time is equivalent to having y_n in (18) $1-\varepsilon$ of the time.

4.3.1 EXAMPLE: CLASSIFYING IRIS FLOWERS

In 1936, R. A. Fisher used discriminant analysis to distinguish different species of iris on the basis of measurements of their flowers. The data he used included measurements by Edgar Anderson of flowers from 50 plants each of two species, *iris setosa* and *iris versicolor*. Two of the measurements, sepal length and petal width, are plotted in Figure 4.

To illustrate how the conformal algorithm can be used for classification, we have randomly chosen 25 of the 100 plants. The sepal lengths and species for the first 24 of them are listed in Table 2 and plotted in Figure 5. The 25th plant in the sample has sepal length 6.8. On the basis of this information, would you classify it as *setosa* or *versicolor*, and how confident would you be in the classification? Because 6.8 is the longest sepal in the sample, nearly any reasonable method will classify the plant as *versicolor*, and this is in fact the correct answer. But the appropriate level of confidence is not so obvious.

We calculate conformal prediction regions using three different nonconformity measures: one based on distance to the nearest neighbors, one based on distance to the species average, and one based on a support-vector machine. Because our evidence is relatively weak, we do not achieve the high precision with high confidence that can be achieved in many applications of machine learning (see, for example, §4.5). But we get a clear view of the details of the calculations and the interpretation of the results.

Distance to the nearest neighbor belonging to each species. Here we use the nonconformity measure (10). The fourth and fifth columns of Table 2 (labeled NN for nearest neighbor) give nonconformity scores α_i obtained with $y_{25} = \mathrm{s}$ and $y_{25} = \mathrm{v}$, respectively. In both cases, these scores are given by

$$\begin{aligned} \alpha_i &= A(\{z_1, \ldots, z_{25}\} \setminus \{z_i\}, z_i) \\ &= \frac{\min\{|x_j - x_i| : 1 \le j \le 25 \;\&\; j \ne i \;\&\; y_j = y_i\}}{\min\{|x_j - x_i| : 1 \le j \le 25 \;\&\; j \ne i \;\&\; y_j \ne y_i\}}, \end{aligned} \tag{19}$$

but for the fourth column $z_{25} = (6.8, \mathrm{s})$, while for the fifth column $z_{25} = (6.8, \mathrm{v})$.

If both the numerator and the denominator in (19) are equal to zero, we take the ratio also to be zero. This happens in the case of the first plant, for example. It has the same sepal length, 5.0, as the 7th and 13th plants, which are *setosa*, and the 15th plant, which is *versicolor*.

Step 3 of the conformal algorithm yields $p_{\mathrm{s}} = 0.08$ and $p_{\mathrm{v}} = 0.32$. Step 4 tells us that

- s is in the $1 - \varepsilon$ prediction region when $1 - \varepsilon > 0.92$, and
- v is in the $1 - \varepsilon$ prediction region when $1 - \varepsilon > 0.68$.

Here are prediction regions for a few levels of ε.

- $\Gamma^{0.08} = \{\mathrm{v}\}$. With 92% confidence, we predict that $y_{25} = \mathrm{v}$.
- $\Gamma^{0.05} = \{\mathrm{s}, \mathrm{v}\}$. If we raise the confidence with which we want to predict y_{25} to 95%, the prediction is completely uninformative.
- $\Gamma^{1/3} = \emptyset$. If we lower the confidence to $2/3$, we get a prediction we know is false: y_{25} will be in the empty set.

	Data		Nonconformity scores					
			NN		Species Average		SVM	
	sepal length	species	α_i for $y_{25} = \text{s}$	α_i for $y_{25} = \text{v}$	α_i for $y_{25} = \text{s}$	α_i for $y_{25} = \text{v}$	α_i for $y_{25} = \text{s}$	α_i for $y_{25} = \text{v}$
z_1	5.0	s	0	0	0.06	0.06	0	0
z_2	4.4	s	0	0	0.66	0.54	0	0
z_3	4.9	s	1	1	0.16	0.04	0	0
z_4	4.4	s	0	0	0.66	0.54	0	0
z_5	5.1	s	0	0	0.04	0.16	0	0
z_6	5.9	v	0.25	0.25	0.12	0.20	0	0
z_7	5.0	s	0	0	0.06	0.06	0	0
z_8	6.4	v	0.50	0.22	0.38	0.30	0	0
z_9	6.7	v	0	0	0.68	0.60	0	0
z_{10}	6.2	v	0.33	0.29	0.18	0.10	0	0
z_{11}	5.1	s	0	0	0.04	0.16	0	0
z_{12}	4.6	s	0	0	0.46	0.34	0	0
z_{13}	5.0	s	0	0	0.06	0.06	0	0
z_{14}	5.4	s	0	0	0.34	0.46	0	0
z_{15}	5.0	v	∞	∞	1.02	1.10	∞	∞
z_{16}	6.7	v	0	0	0.68	0.60	0	0
z_{17}	5.8	v	0	0	0.22	0.30	0	0
z_{18}	5.5	s	0.50	0.50	0.44	0.56	0	0
z_{19}	5.8	v	0	0	0.22	0.30	0	0
z_{20}	5.4	s	0	0	0.34	0.46	0	0
z_{21}	5.1	s	0	0	0.04	0.16	0	0
z_{22}	5.7	v	0.50	0.50	0.32	0.40	0	0
z_{23}	4.6	s	0	0	0.46	0.34	0	0
z_{24}	4.6	s	0	0	0.46	0.34	0	0
z_{25}	6.8	s	13		1.74		∞	
z_{25}	6.8	v		0.077		0.7		0
p_s			0.08		0.04		0.08	
p_v				0.32		0.08		1

Table 2: **Conformal prediction of iris species from sepal length, using three different nonconformity measures.** The data used are sepal length and species for a random sample of 25 of the 100 plants measured by Edgar Anderson. The second column gives x_i, the sepal length. The third column gives y_i, the species. The 25th plant has sepal length $x_{25} = 6.8$, and our task is to predict its species y_{25}. For each nonconformity measure, we calculate nonconformity scores under each hypothesis, $y_{25} = \text{s}$ and $y_{25} = \text{v}$. The p-value in each column is computed from the 25 nonconformity scores in that column; it is the fraction of them equal to or larger than the 25th. The results from the three nonconformity measures are consistent, inasmuch as the p-value for v is always larger than the p-value for s.

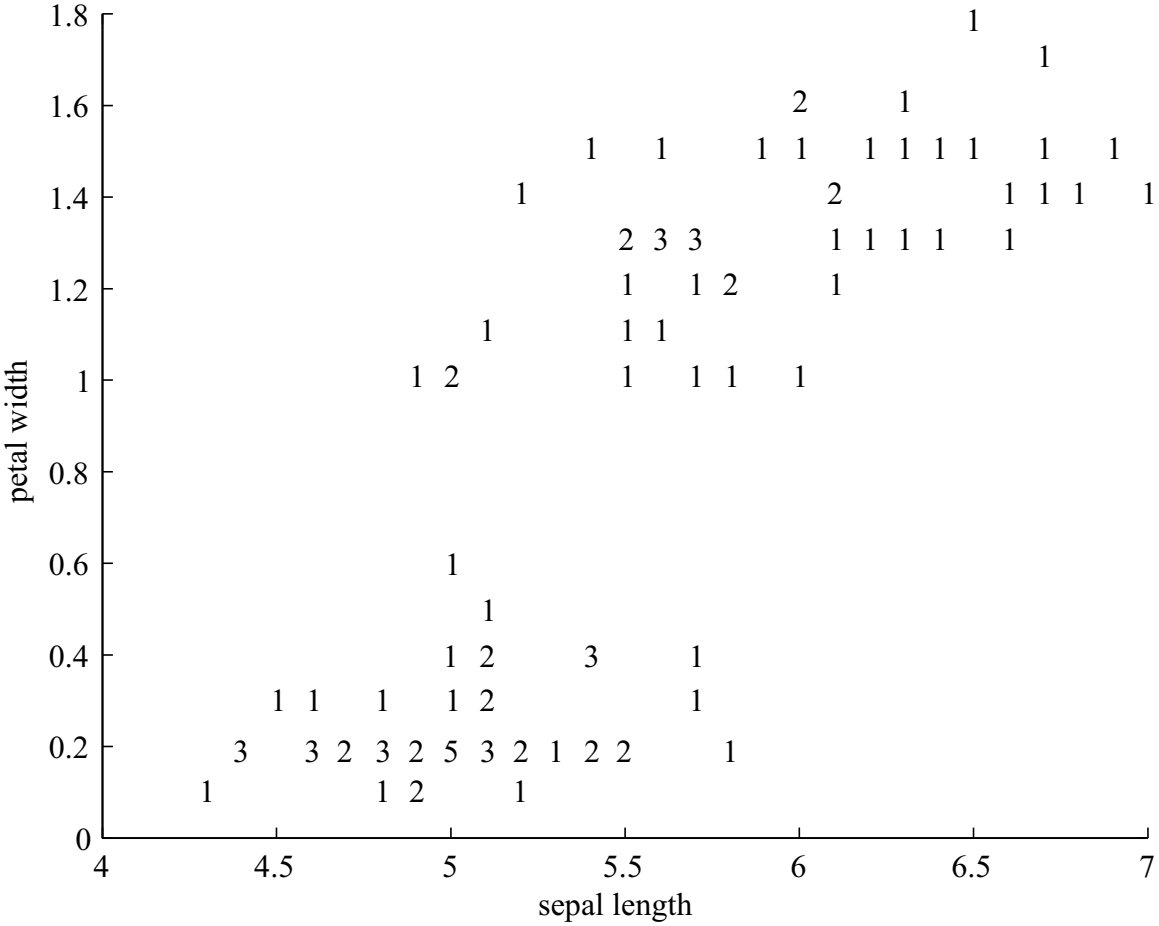


Figure 4: **Sepal length, petal width, and species for Edgar Anderson's** 100 **flowers.** The 50 *iris setosa* are clustered at the lower left, while the 50 *iris versicolor* are clustered at the upper right. The numbers indicate how many plants have exactly the same measurement; for example, there are 5 plants that have sepals 5 inches long and petals 0.2 inches wide. Petal width separates the two species perfectly; all 50 *versicolor* petals are 1 inch wide or wider, while all *setosa* petals are narrower than 1 inch. But there is substantial overlap in sepal length.

In fact, $y_{25} = \text{v}$. Our 92% prediction is correct.

The fact that we are making a known-to-be-false prediction with $2/3$ confidence is a signal that the 25th sepal length, 6.8, is unusual for either species. A close look at the nonconformity scores reveals that it is being perceived as unusual simply because $2/3$ of the plants have other plants in the sample with exactly the same sepal length, whereas there is no other plant with the sepal length 6.8.

In classification problems, it is natural to report the greatest $1-\varepsilon$ for which Γ^ε is a single label. In our example, this produces the statement that we are 92% confident that y_{25} is v. But in order

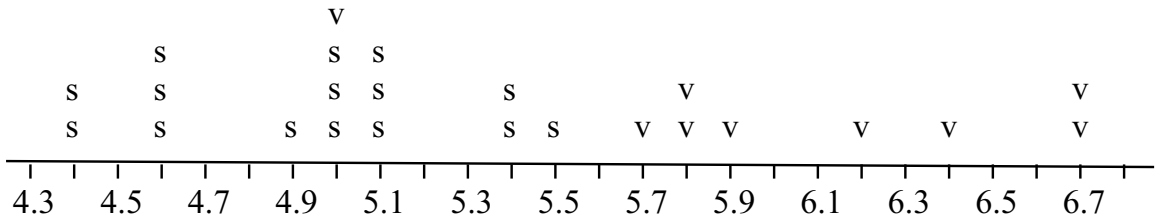


Figure 5: **Sepal length and species for the first** 24 **plants in our random sample of size** 25**.** Except for one *versicolor* with sepal length 5.0, the *versicolor* in this sample all have longer sepals than the *setosa*. This high degree of separation is an accident of the sampling.

to avoid overconfidence when the object x_n is unusual, it is wise to report also the largest ε for which Γ^ε is empty. We call this the *credibility* of the prediction (Vovk et al., 2005, p. 96).[3] In our example, the prediction that y_{25} will be v has credibility of only 32%, indicating that the example is somewhat unusual for the method that produces the prediction—so unusual that the method has 68% confidence in a prediction of y_{25} that we know is false before we observe y_{25} ($\Gamma^{0.68} = \emptyset$).

Distance to the average of each species. The nearest-neighbor nonconformity measure, because it considers only nearby sepal lengths, does not take full advantage of the fact that a *versicolor* flower typically has longer sepals than a *setosa* flower. We can expect to obtain a more efficient conformal predictor (one that produces smaller regions for a given level of confidence) if we use a nonconformity measure that takes account of average sepal length for the two species.

We use the nonconformity measure A defined by

$$A(B,(x,y)) = |\bar{x}_{B\cup\{(x,y)\},y} - x|, \tag{20}$$

where $\bar{x}_{B,y}$ denotes the average sepal length of all plants of species y in the bag B, and $B\cup\{z\}$ denotes the bag obtained by adding z to B. To test $y_{25} = \mathrm{s}$, we consider the bag consisting of the 24 old examples together with $(6.8,\mathrm{s})$, and we calculate the average sepal lengths for the two species in this bag: 5.06 for *setosa* and 6.02 for *versicolor*. Then we use (20) to calculate the nonconformity scores shown in the sixth column of Table 2:

$$\alpha_i = \begin{cases} |5.06 - x_i| & \text{if } y_i = \mathrm{s} \\ |6.02 - x_i| & \text{if } y_i = \mathrm{v} \end{cases}$$

for $i = 1,\ldots,25$, where we take y_{25} to be s. To test $y_{25} = \mathrm{v}$, we consider the bag consisting of the 24 old examples together with $(6.8,\mathrm{v})$, and we calculate the average sepal lengths for the two species in this bag: 4.94 for *setosa* and 6.1 for *versicolor*. Then we use (20) to calculate the nonconformity scores shown in the seventh column of Table 2:

$$\alpha_i = \begin{cases} |4.94 - x_i| & \text{if } y_i = \mathrm{s} \\ |6.1 - x_i| & \text{if } y_i = \mathrm{v} \end{cases}$$

for $i = 1,\ldots,25$, where we take y_{25} to be v.

We obtain $p_\mathrm{s} = 0.04$ and $p_\mathrm{v} = 0.08$, so that

3. This notion of credibility is one of the novelties of the theory of conformal prediction. It is not found in the prior literature on confidence and prediction regions.

- s is in the $1-\varepsilon$ prediction region when $1-\varepsilon > 0.96$, and
- v is in the $1-\varepsilon$ prediction region when $1-\varepsilon > 0.92$.

Here are the prediction regions for some different levels of ε.

- $\Gamma^{0.04} = \{\text{v}\}$. With 96% confidence, we predict that $y_{25} = \text{v}$.
- $\Gamma^{0.03} = \{\text{s}, \text{v}\}$. If we raise the confidence with which we want to predict y_{25} to 97%, the prediction is completely uninformative.
- $\Gamma^{0.08} = \emptyset$. If we lower the confidence to 92%, we get a prediction we know is false: y_{25} will be in the empty set.

In this case, we predict $y_{25} = \text{v}$ with confidence 96% but credibility only 8%. The credibility is lower with this nonconformity measure because it perceives 6.8 as being even more unusual than the nearest-neighbor measure did. It is unusually far from the average sepal lengths for both species.

A support-vector machine. As Vladimir Vapnik explains on pp. 408–410 of his *Statistical Learning Theory* (1998), support-vector machines grew out of the idea of separating two groups of examples with a hyperplane in a way that makes as few mistakes as possible—that is, puts as few examples as possible on the wrong side. This idea springs to mind when we look at Figure 5. In this one-dimensional picture, a hyperplane is a point. We are tempted to separate the *setosa* from the *versicolor* with a point between 5.5 and 5.7.

Vapnik proposed to separate two groups not with a single hyperplane but with a band: two hyperplanes with few or no examples between them that separate the two groups as well as possible. Examples on the wrong side of both hyperplanes would be considered very strange; those between the hyperplanes would also be considered strange but less so. In our one-dimensional example, the obvious separating band is the interval from 5.5 to 5.7. The only strange example is the *versicolor* with sepal length 5.0.

Here is one way of making Vapnik's idea into an algorithm for calculating nonconformity scores for all the examples in a bag $⦇(x_1,y_1),\ldots(x_n,y_n)⦈$. First plot all the examples as in Figure 5. Then find numbers a and b such that $a \le b$ and the interval $[a,b]$ separates the two groups with the fewest mistakes—that is, minimizes[4]

$$\#\{i \mid 1 \le i \le n, x_i < b, \text{ and } y_i = \text{v}\} + \#\{i \mid 1 \le i \le n, x_i > a, \text{ and } y_i = \text{s}\}.$$

There may be many intervals that minimize this count; choose one that is widest. Then give the ith example the score

$$\alpha_i = \begin{cases} \infty & \text{if } y_i = \text{v and } x_i < a \quad \text{or } y_i = \text{s and } b < x_i \\ 1 & \text{if } y_i = \text{v and } a \le x_i < b \quad \text{or } y_i = \text{s and } a < x_i \le b \\ 0 & \text{if } y_i = \text{v and } b \le x_i \quad \text{or } y_i = \text{s and } x_i \le a. \end{cases}$$

When applied to the bags in Figure 6, this algorithm gives the circled examples the score ∞ and all the others the score 0. These scores are listed in the last two columns of Table 2.

4. Here we are implicitly assuming that the *setosa* flowers will be on the left, with shorter sepal lengths. A general algorithm should also check the possibility of a separation with the *versicolor* flowers on the left.

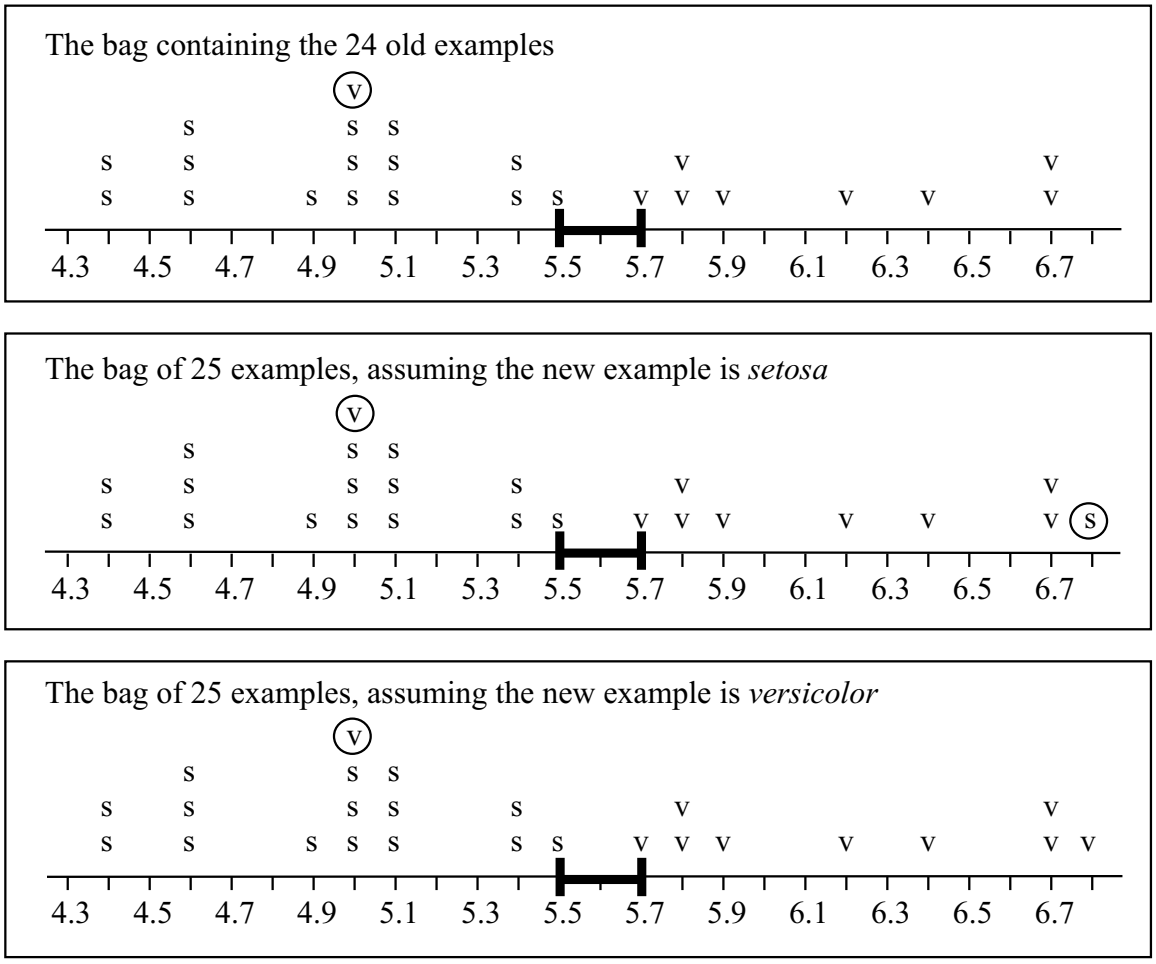


Figure 6: **Separation for three bags.** In each case, the separating band is the interval $[5.5, 5.7]$. Examples on the wrong side of the interval are considered strange and are circled.

As we see from the table, the resulting p-values are $p_{\text{s}} = 0.08$ and $p_{\text{v}} = 1$. So this time we obtain 92% confidence in $y_{25} = \text{v}$, with 100% credibility.

The algorithm just described is too complex to implement when there are thousands of examples. For this reason, Vapnik and his collaborators proposed instead a quadratic minimization that balances the width of the separating band against the number and size of the mistakes it makes. Support-vector machines of this type have been widely used. They usually solve the dual optimization problem, and the Lagrange multipliers they calculate can serve as nonconformity scores. Implementations sometimes fail to treat the old examples symmetrically because they make various uses of the order in which examples are presented, but this difficulty can be overcome by a preliminary randomization (Vovk et al., 2005, p. 58).

A systematic comparison. The random sample of 25 plants we have considered is odd in two ways: (1) except for the one *versicolor* with sepal length of only 5.0, the two species do not overlap

	NN	Species average	SVM
singleton hits	164	441	195
uncertain	795	477	762
total hits	959	918	957
empty	9	49	1
singleton errors	32	33	42
total errors	41	82	43
total examples	1000	1000	1000
% hits	96%	92%	96%
total singletons	196	474	237
% hits	84%	93%	82%
total errors	41	82	43
% empty	22%	60%	2%

Table 3: **Performance of** 92% **prediction regions based on three nonconformity measures.** For each nonconformity measure, we have found $1,000$ prediction regions at the 92% level, using each time a different random sample of 25 from Anderson's 100 flowers. The "uncertain" regions are those equal to the whole label space, $\mathbf{Y} = \{\mathrm{s}, \mathrm{v}\}$.

in sepal length, and (2) the flower whose species we are trying to predict has a sepal that is unusually long for either species.

In order to get a fuller picture of how the three nonconformity measures perform in general on the iris data, we have applied each of them to $1,000$ different samples of size 25 selected from the population of Anderson's 100 plants. The results are shown in Table 3.

The 92% regions based on the species average were correct about 92% of the time (918 times out of 1000), as advertised. The regions based on the other two measures were correct more often, about 96% of the time. The reason for this difference is visible in Table 2; the nonconformity scores based on the species average take a greater variety of values and therefore produce ties less often. The regions based on the species averages are also more efficient (smaller); 441 of its hits were informative, as opposed to fewer than 200 for each of the other two nonconformity measures. This efficiency also shows up in more empty regions among the errors. The species average produced an empty 92% prediction region for the random sample used in Table 2, and Table 3 shows that this happens 5% of the time.

As a practical matter, the uncertain prediction regions ($\Gamma^{0.08} = \{\mathrm{s}, \mathrm{v}\}$) and the empty ones ($\Gamma^{0.08} = \emptyset$) are equally uninformative. The only errors that mislead are the singletons that are wrong, and the three methods all produce these at about the same rate—3 or 4%.

4.3.2 EXAMPLE: PREDICTING PETAL WIDTH FROM SEPAL LENGTH

We now turn to the use of the conformal algorithm to predict a number. We use the same 25 plants, but now we use the data in the second and third columns of Table 4: the sepal length and petal width for the first 24 plants, and the sepal length for the 25th. Our task is to predict the petal width for the 25th.

	sepal length	petal width	Nearest neighbor	Linear regression
z_1	5.0	0.3	0.3	$\lvert 0.003y_{25} - 0.149\rvert$
z_2	4.4	0.2	0	$\lvert 0.069y_{25} + 0.050\rvert$
z_3	4.9	0.2	0.25	$\lvert 0.014y_{25} - 0.199\rvert$
z_4	4.4	0.2	0	$\lvert 0.069y_{25} + 0.050\rvert$
z_5	5.1	0.4	0.15	$\lvert 0.008y_{25} + 0.099\rvert$
z_6	5.9	1.5	0.3	$\lvert 0.096y_{25} - 0.603\rvert$
z_7	5.0	0.2	0.4	$\lvert 0.003y_{25} - 0.249\rvert$
z_8	6.4	1.3	0.2	$\lvert 0.151y_{25} - 0.154\rvert$
z_9	6.7	1.4	0.3	$\lvert 0.184y_{25} - 0.104\rvert$
z_{10}	6.2	1.5	0.2	$\lvert 0.129y_{25} - 0.453\rvert$
z_{11}	5.1	0.2	0.15	$\lvert 0.008y_{25} + 0.299\rvert$
z_{12}	4.6	0.2	0.05	$\lvert 0.047y_{25} - 0.050\rvert$
z_{13}	5.0	0.6	0.3	$\lvert 0.003y_{25} + 0.151\rvert$
z_{14}	5.4	0.4	0	$\lvert 0.041y_{25} + 0.248\rvert$
z_{15}	5.0	1.0	0.75	$\lvert 0.003y_{25} + 0.551\rvert$
z_{16}	6.7	1.7	0.3	$\lvert 0.184y_{25} - 0.404\rvert$
z_{17}	5.8	1.2	0.2	$\lvert 0.085y_{25} - 0.353\rvert$
z_{18}	5.5	0.2	0.2	$\lvert 0.052y_{25} + 0.498\rvert$
z_{19}	5.8	1.0	0.2	$\lvert 0.085y_{25} - 0.153\rvert$
z_{20}	5.4	0.4	0	$\lvert 0.041y_{25} + 0.248\rvert$
z_{21}	5.1	0.3	0	$\lvert 0.008y_{25} + 0.199\rvert$
z_{22}	5.7	1.3	0.2	$\lvert 0.074y_{25} - 0.502\rvert$
z_{23}	4.6	0.3	0.1	$\lvert 0.047y_{25} + 0.050\rvert$
z_{24}	4.6	0.2	0.05	$\lvert 0.047y_{25} - 0.050\rvert$
z_{25}	6.8	y_{25}	$\lvert y_{25} - 1.55\rvert$	$\lvert 0.805y_{25} - 1.345\rvert$

Table 4: **Conformal prediction of petal width from sepal length.** We use the same random 25 plants that we used for predicting the species. The actual value of y_{25} is 1.4.

The most conventional way of analyzing this data is to calculate the least-squares line (11):

$$\hat{y} = a_{24} + b_{24}x = -2.96 + 0.68x.$$

The sepal length for the 25th plant being $x_{25} = 6.8$, the line predicts that y_{25} should be near $-2.96 + 0.68 \times 6.8 = 1.66$. Under the textbook assumption that the y_i are all independent and normally distributed with means on the line and a common variance, we estimate the common variance by

$$s_{24}^2 = \frac{\sum_{i=1}^{24}(y_i - (a_{24} + b_{24}x_i))^2}{22} = 0.0780.$$

The textbook $1-\varepsilon$ interval for y_{25} based on $\wr(x_1,y_1),\ldots,(x_{24},y_{24})\wr$ and x_{25} is

$$1.66 \pm t_{22}^{\varepsilon/2} s_{24} \sqrt{1 + \frac{1}{24} + \frac{(x_{25} - \bar{x}_{24})^2}{\sum_{j=1}^{24}(x_j - \bar{x}_{24})^2}} = 1.66 \pm 0.311 t_{22}^{\varepsilon/2} \tag{21}$$

(Draper and Smith 1998, p. 82; Ryan 1997, pp. 21–22; Seber and Lee 2003, p. 145). Taking into account the fact y_{25} is measured to only one decimal place, we obtain $[1.0, 2.3]$ for the 96% interval and $[1.1, 2.2]$ for the 92% interval.

The prediction interval (21) is analogous to Fisher's interval for a new example from the same normally distributed population as a bag of old examples (§2.1.1). In §5.3.2 we will review the general model of which both are special cases.

As we will now see, the conformal algorithm under exchangeability gives confidence intervals comparable to (21), without the assumption that the errors are normal. We use two different nonconformity measures: one based on the nearest neighbor, and one based on the least-squares line.

Conformal prediction using the nearest neighbor. Suppose B is a bag of old examples and (x,y) is a new example, for which we know the sepal length x but not the petal width y. We can predict y using the nearest neighbor in an obvious way: We find the $z' \in B$ for which the sepal length x' is closest to x, and we predict that y will be the same as the petal width y'. If there are several examples in the bag with sepal length equally close to x, then we take the median of their petal widths as our predictor $\hat{y}$. The associated nonconformity measure is $|y - \hat{y}|$.

The fourth column of Table 4 gives the nonconformity scores for our sample using this nonconformity measure. We see that $\alpha_{25} = |y_{25} - 1.55|$. The other nonconformity scores do not involve y_{25}; the largest is 0.75, and the second largest is 0.40. So we obtain these prediction regions y_{25}:

- The 96% prediction region consists of all the y for which $p_y > 0.04$, which requires that at least one of the other α_i be as large as α_{25}, or that $0.75 \geq |y - 1.55|$. This is the interval $[0.8, 2.3]$.

- The 92% prediction region consists of all the y for which $p_y > 0.08$, which requires that at least two of the other α_i be as large as α_{25}, or that $0.40 \geq |y - 1.55|$. This is the interval $[1.2, 1.9]$.

Conformal prediction using least-squares. Now we use the least-squares nonconformity measure with inclusion, given by (12). In our case, $n = 25$, so our nonconformity scores are

$$\begin{aligned}\alpha_i &= |y_i - (a_{25} + b_{25} x_i)| \\ &= \left| y_i - \frac{\sum_{j=1}^{25} y_j}{25} - \frac{\sum_{j=1}^{25} (x_j - \bar{x}_{25}) y_j}{\sum_{j=1}^{25} (x_j - \bar{x}_{25})^2} \left(x_i - \frac{\sum_{j=1}^{25} x_j}{25} \right) \right| .\end{aligned}$$

When we substitute values of $\sum_{j=1}^{24} y_j$, $\sum_{j=1}^{24} (x_j - \bar{x}_{25}) y_j$, $\sum_{j=1}^{25} (x_j - \bar{x}_{25})^2$, and $\sum_{j=1}^{25} x_j$ calculated from Table 4, this becomes

$$\alpha_i = |y_i + (0.553 - 0.110 x_i) y_{25} - 0.498 x_i + 2.04| .$$

For $i = 1, \ldots, 24$, we can further evaluate this by substituting the values of x_i and y_i. For $i = 25$, we can substitute 6.8 for x_{25}. These substitutions produce the expressions of the form $|c_i y_{25} + d_i|$ listed in the last column of Table 4. We have made sure that c_i is always positive by multiplying by -1 within the absolute value when need be.

Table 5 shows calculations required to find the conformal prediction region. The task is to identify, for $i = 1, \ldots, 24$, the y for which $|c_i y + d_i| \geq |0.805 y - 1.345|$. We first find the solutions of

the equation $|c_i y + d_i| = |0.805y - 1.345|$, which are

$$-\frac{d_i + 1.345}{c_i - 0.805} \qquad \text{and} \qquad -\frac{d_i - 1.345}{c_i + 0.805}.$$

As it happens, $c_i < 0.805$ for $i = 1, \ldots, 24$, and in this case the y satisfying $|c_i y + d_i| \geq |0.805 - 1.345|$ form the interval between these two points. This interval is shown in the last column of the table.

In order to be in the 96% interval, y must be in at least one of the 24 intervals in the table; in order to be in the 92% interval, it must be in at least two of them. So the 96% interval is $[1.0, 2.4]$, and the 92% interval is $[1.0, 2.3]$.

An algorithm for finding conformal prediction intervals using a least-squares or ridge-regression nonconformity measure with an object space of any finite dimension is spelled out on pp. 32–33 of our book (Vovk et al., 2005).

4.4 Optimality

The predictions produced by the conformal algorithm are invariant with respect to the old examples, correct with the advertised probability, and nested. As we now show, they are optimal among all region predictors with these properties.

Here is a more precise statement of the three properties:

1. *The predictions are invariant with respect to the ordering of the old examples.* Formally, this means that the predictor γ is a function of two variables, the significance level ε and the bag B of old examples. We write $\gamma^\varepsilon(B)$ for the prediction, which is a subset of the example space $\mathbf{Z}$.

2. *The probability of a hit is always at least the advertised confidence level.* For every positive integer n and every probability distribution under which $z_1, \ldots, z_n$ are exchangeable,

$$\Pr\{z_n \in \gamma^\varepsilon(\lbag z_1, \ldots, z_{n-1} \rbag)\} \geq 1 - \varepsilon.$$

3. *The prediction regions are nested.* If $\varepsilon_1 \geq \varepsilon_2$, then $\gamma^{\varepsilon_1}(B) \subseteq \gamma^{\varepsilon_2}(B)$.

Conformal predictors satisfy these three conditions. Other region predictors can also satisfy them. But as we now demonstrate, any γ satisfying them can be improved on by a conformal predictor: there always exists a nonconformity measure A such that the predictor γ_A constructed from A by the conformal algorithm satisfies $\gamma_A^\varepsilon(B) \subseteq \gamma^\varepsilon(B)$ for all B and ε.

The key to the demonstration is the following lemma:

Lemma 1 *Suppose γ is a region predictor satisfying the three conditions, $\lbag a_1, \ldots, a_n \rbag$ is a bag of examples, and $0 < \varepsilon \leq 1$. Then $n\varepsilon$ or fewer of the n elements of the bag satisfy*

$$a_i \notin \gamma^\varepsilon(\lbag a_1, \ldots, a_n \rbag \setminus \lbag a_i \rbag). \tag{22}$$

Proof Consider the unique exchangeable probability distribution for $z_1, \ldots, z_n$ that gives probability 1 to $\lbag z_1, \ldots, z_n \rbag = \lbag a_1, \ldots, a_n \rbag$. Under this distribution, each element of $\lbag a_1, \ldots, a_n \rbag$ has an equal probability of being z_n, and in this case, (22) is a mistake. By the second condition, the probability of a mistake is ε or less. So the fraction of the bag's elements for which (22) holds is ε or less. ■

	$\alpha_i = \lvert c_i y_{25} + d_i \rvert$	$-\frac{d_i + 1.345}{c_i - 0.805}$	$-\frac{d_i - 1.345}{c_i + 0.805}$	y satisfying $\lvert c_i y + d_i \rvert \geq \lvert 0.805 - 1.345 \rvert$
z_1	$\lvert 0.003 y_{25} - 0.149 \rvert$	1.49	1.85	[1.49,1.85]
z_2	$\lvert 0.069 y_{25} + 0.050 \rvert$	1.90	1.48	[1.48,1.90]
z_3	$\lvert 0.014 y_{25} - 0.199 \rvert$	1.45	1.89	[1.45,1.89]
z_4	$\lvert 0.069 y_{25} + 0.050 \rvert$	1.90	1.48	[1.48,1.90]
z_5	$\lvert 0.008 y_{25} + 0.099 \rvert$	1.81	1.53	[1.53,1.81]
z_6	$\lvert 0.096 y_{25} - 0.603 \rvert$	1.05	2.16	[1.05,2.16]
z_7	$\lvert 0.003 y_{25} - 0.249 \rvert$	1.37	1.97	[1.37,1.97]
z_8	$\lvert 0.151 y_{25} - 0.154 \rvert$	1.82	1.57	[1.57,1.82]
z_9	$\lvert 0.184 y_{25} - 0.104 \rvert$	2.00	1.47	[1.47,2.00]
z_{10}	$\lvert 0.129 y_{25} - 0.453 \rvert$	1.32	1.93	[1.32,1.93]
z_{11}	$\lvert 0.008 y_{25} + 0.299 \rvert$	2.06	1.29	[1.29,2.06]
z_{12}	$\lvert 0.047 y_{25} - 0.050 \rvert$	1.71	1.64	[1.64,1.71]
z_{13}	$\lvert 0.003 y_{25} + 0.151 \rvert$	1.87	1.48	[1.48,1.87]
z_{14}	$\lvert 0.041 y_{25} + 0.248 \rvert$	2.09	1.30	[1.30,2.09]
z_{15}	$\lvert 0.003 y_{25} + 0.551 \rvert$	2.36	0.98	[0.98,2.36]
z_{16}	$\lvert 0.184 y_{25} - 0.404 \rvert$	1.52	1.77	[1.52,1.77]
z_{17}	$\lvert 0.085 y_{25} - 0.353 \rvert$	1.38	1.91	[1.38,1.91]
z_{18}	$\lvert 0.052 y_{25} + 0.498 \rvert$	2.45	0.99	[0.99,2.45]
z_{19}	$\lvert 0.085 y_{25} - 0.153 \rvert$	1.66	1.68	[1.66,1.68]
z_{20}	$\lvert 0.041 y_{25} + 0.248 \rvert$	2.09	1.30	[1.30,2.09]
z_{21}	$\lvert 0.008 y_{25} + 0.199 \rvert$	1.94	1.41	[1.41,1.94]
z_{22}	$\lvert 0.074 y_{25} - 0.502 \rvert$	1.15	2.10	[1.15,2.10]
z_{23}	$\lvert 0.047 y_{25} + 0.050 \rvert$	1.84	1.52	[1.52,1.84]
z_{24}	$\lvert 0.047 y_{25} - 0.050 \rvert$	1.71	1.64	[1.64,1.71]
z_{25}	$\lvert 0.805 y_{25} - 1.345 \rvert$			

Table 5: **Calculations with least-squares nonconformity scores.** The column on the right gives the values of y for which the example's nonconformity score will exceed that of the 25th example.

Given the region predictor γ, what nonconformity measure will give us a conformal predictor that improves on it? If

$$z \notin \gamma^{\delta}(B), \tag{23}$$

then γ is asserting confidence $1-\delta$ that z should not appear next because it is so different from B. So the largest $1-\delta$ for which (23) holds is a natural nonconformity measure:

$$A(B,z) = \sup\{1-\delta \,|\, z \notin \gamma^{\delta}(B)\}.$$

	Least-squares prediction with normal errors	Conformal prediction with two different nonconformity measures	
		NN	Least squares
96%	$[1.0, 2.3]$	$[0.8, 2.3]$	$[1.0, 2.4]$
92%	$[1.1, 2.2]$	$[1.2, 1.9]$	$[1.0, 2.3]$

Table 6: **Prediction intervals for the 25th plant's petal width, calculated by three different methods.** The conformal prediction intervals using the least-squares nonconformity measure are quite close to the standard intervals based on least-squares with normal errors. All the intervals contain the actual value, 1.4.

The conformal predictor γ_A obtained from this nonconformity measure, though it agrees with γ on how to rank different z with respect to their nonconformity with B, may produce tighter prediction regions if γ is too conservative in the levels of confidence it asserts.

To show that $\gamma_A^\varepsilon(B) \subseteq \gamma^\varepsilon(B)$ for every ε and every B, we assume that

$$z \in \gamma_A^\varepsilon(\{z_1, \ldots, z_{n-1}\}) \tag{24}$$

and show that $z \in \gamma^\varepsilon(\{z_1, \ldots, z_{n-1}\})$. According to the conformal algorithm, (24) means that when we provisionally set z_n equal to z and calculate the nonconformity scores

$$\alpha_i = \sup\{1-\delta \,|\, z_i \notin \gamma^\delta(\{z_1, \ldots, z_n\} \setminus \{z_i\})\}$$

for $i = 1, \ldots, n$, we find that strictly more than $n\varepsilon$ of these scores are greater than or equal to α_n. Because γ's prediction regions are nested (condition 3), it follows that if $z_n \notin \gamma^\varepsilon(\{z_1, \ldots, z_{n-1}\})$, then $z_i \notin \gamma^\varepsilon(\{z_1, \ldots, z_n\} \setminus \{z_i\})$ for strictly more than $n\varepsilon$ of the z_i. But by Lemma 1, $n\varepsilon$ or fewer of the z_i can satisfy this condition. So $z_n \in \gamma^\varepsilon(\{z_1, \ldots, z_{n-1}\})$.

There are sensible reasons to use region predictors that are not invariant. We may want to exploit possible departures from exchangeability even while insisting on validity under exchangeability. Or it may simply be more practical to use a predictor that is not invariant. But invariance is a natural condition when we want to rely only on exchangeability, and in this case our optimality result is persuasive. For further discussion, see §2.4 of our book (Vovk et al., 2005).

4.5 Examples Are Seldom Exactly Exchangeable

Although the assumption of exchangeability is weak compared to the assumptions embodied in most statistical models, it is still an idealization, seldom matched exactly by what we see in the world. So we should not expect conclusions derived from this assumption to be exactly true. In particular, we should not be surprised if a 95% conformal predictor is wrong more than 5% of the time.

We can make this point with the USPS data set so often used to illustrate machine learning methods. This data set consists of 9298 examples of the form (x, y), where x is a 16×16 gray-scale matrix and y is one of the ten digits $0, 1, \ldots, 9$. It has been used in hundreds of books and articles. In our book (Vovk et al., 2005), it is used to illustrate conformal prediction with a number of different nonconformity measures. It is well known that the examples in this data set are not perfectly exchangeable. In particular, the first 7291 examples, which are often treated as a training

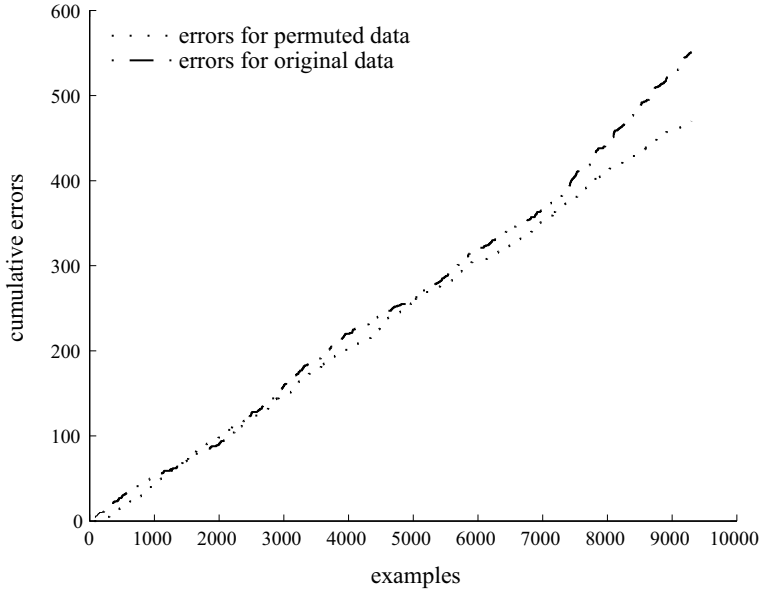


Figure 7: **Errors in** 95% **nearest-neighbor conformal prediction on the classical USPS data set.** When the 9298 examples are predicted in a randomly chosen order, so that the exchangeability assumption is satisfied for sure, the error rate is approximately 5% as advertised. When they are taken in their original order, first the 7291 in the training set, and then the 2007 in the test set, the error rate is higher, especially in the test set.

set, are systematically different in some respects from the remaining 2007 examples, which are usually treated as a test set.

Figure 7 illustrates how the non-exchangeability of the USPS data affects conformal prediction. The figure records the performance of the 95% conformal predictor using the nearest-neighbor nonconformity measure (10), applied to the USPS data in two ways. First we use the 9298 examples in the order in which they are given in the data set. (We ignore the distinction between training and test examples, but since the training examples are given first we do go through them first.) Working through the examples in this order, we predict each y_n using the previous examples and x_n. Second, we randomly permute all 9298 examples, thus producing an order with respect to which the examples are necessarily exchangeable. The law of large numbers works when we go through the examples in the permuted order: we make mistakes at a steady rate, about equal to the expected 5%. But when we go through the examples in the original order, the fraction of mistakes is less stable, and it worsens as we move into the test set. As Table 7 shows, the fraction of mistakes is 5%, as desired, in the first 7291 examples (the training set) but jumps to 8% in the last 2007 examples.

Non-exchangeability can be tested statistically, using conventional or game-theoretic methods (Vovk et al., 2005, §7.1). In the case of this data, any reasonable test will reject exchangeability decisively. Whether the deviation from exchangeability is of practical importance for prediction depends, of course, on circumstances. An error rate of 8% when 5% has been promised may or may not be acceptable.

	Original data			Permuted data		
	Training	Test	Total	Training	Test	Total
singleton hits	6798	1838	8636	6800	1905	8705
uncertain hits	111	0	111	123	0	123
total hits	6909	1838	8747	6923	1905	8828
empty	265	142	407	205	81	286
singleton errors	102	27	129	160	21	181
uncertain errors	15	0	15	3	0	3
total errors	382	169	551	368	102	470
total examples	7291	2007	9298	7291	2007	9298
% hits	95%	92%	94%	95%	95%	95%
total singletons	6900	1865	8765	6960	1926	8880
% hits	99%	99%	99%	98%	99%	98%
total uncertain	126	0	126	126	0	126
% hits	82%		82%	98%		98%
total errors	382	169	551	368	102	470
% empty	69%	85%	74%	57%	79%	61%

Table 7: **Details of the performance of** 95% **nearest-neighbor conformal prediction on the classical USPS data set.** Because there are 10 labels, the uncertain predictions, those containing more than one label, can be hits or errors.

5. On-Line Compression Models

In this section, we generalize conformal prediction from the exchangeability model to a whole class of models, which we call on-line compression models.

In the exchangeability model, we compress or summarize examples by omitting information about their order. We then look backwards from the summary (the bag of unordered examples) and give probabilities for the different orderings that could have produced it. The compression can be done on-line: each time we see a new example, we add it to the bag. The backward-looking probabilities can also be given step by step. Other on-line compression models compress more or less drastically but have a similar structure.

On-line compression models were studied in the 1970s and 1980s, under various names, by Per Martin-Löf (1974), Steffen Lauritzen (1988), and Eugene Asarin (1987; 1988). Different authors had different motivations. Lauritzen and Martin-Löf started from statistical mechanics, whereas Asarin started from Kolmogorov's thinking about the meaning of randomness. But the models they studied all summarize past examples using statistics that contain all the information useful for predicting future examples. The summary is updated each time one observes a new example, and the probabilistic content of the structure is expressed by Markov kernels that give probabilities for summarized examples conditional on the summaries.

In general, a Markov kernel is a mapping that specifies, as a function of one variable, a probability distribution for some other variable or variables. A Markov kernel for w given u, for example, gives a probability distribution for w for each value of u. It is conventional to write $P(w \mid u)$ for this

distribution. We are interested in Markov kernels of the form $P(z_1,\dots,z_n \mid \sigma_n)$, where σ_n summarizes the examples $z_1,\dots,z_n$. Such a kernel gives probabilities for the different $z_1,\dots,z_n$ that could have produced σ_n.

Martin-Löf, Lauritzen, and Asarin were interested in justifying widely used statistical models from principles that seem less arbitrary than the models themselves. On-line compression models offer an opportunity to do this, because they typically limit their use of probability to representing ignorance with a uniform distribution but lead to statistical models that seem to say something more. Suppose, for example, that Joe summarizes numbers $z_1,\dots,z_n$ by

$$\bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i \qquad \text{and} \qquad r^2 = \sum_{i=1}^{n} (z_i - \bar{z})^2$$

and gives these summaries to Bill, who does not know $z_1,\dots,z_n$. Bill might adopt a probability distribution for $z_1,\dots,z_n$ that is uniform over the possibilities, which form the $(n-1)$-dimensional sphere of radius r centered around $(\bar{z},\dots,\bar{z})$. As we will see in §5.3.2, this is an on-line compression model. It was shown, by Freedman and Smith (see Vovk et al., 2005, p. 217) and then by Lauritzen (1988, pp. 238–247), that if we assume this model is valid for all n, then the distribution of $z_1, z_2, \dots$ must be a mixture of distributions under which $z_1, z_2, \dots$ are independent and normal with a common mean and variance. This is analogous to de Finetti's theorem, which says that if $z_1,\dots,z_n$ are exchangeable for all n, then the distribution of $z_1, z_2, \dots$ must be a mixture of distributions under which $z_1, z_2, \dots$ are independent and identically distributed.

For our own part, we are interested in using an on-line compression model directly for prediction rather than as a step towards a model that specifies probabilities for examples more fully. We have already seen how the exchangeability model can be used directly for prediction: we establish a law of large numbers for backward-looking probabilities (§3.4), and we use it to justify confidence in conformal prediction regions (§4.2). The argument extends to on-line compression models in general.

For the exchangeability model, conformal prediction is optimal for obtaining prediction regions (§4.4). No such statement can be made for on-line compression models in general. In fact, there are other on-line compression models in which conformal prediction is very inefficient (Vovk et al., 2005, p. 220).

After developing the general theory of conformal prediction for on-line compression models (§5.1 and §5.2), we consider two examples: the exchangeability-within-label model (§5.3.1) and on-line Gaussian linear model (§5.3.2).

5.1 Definitions

A more formal look at the exchangeability model will suffice to bring the general notion of an on-line compression model into focus.

In the exchangeability model, we summarize examples simply by omitting information about their ordering; the ordered examples are summarized by a bag containing them. The backward-looking probabilities are equally simple; given the bag, the different possible orderings all have equal probability, as if the ordering resulted from drawing the examples successively at random from the bag without replacement. Although this picture is very simple, we can distinguish four distinct mathematical operations within it:

1. *Summarizing.* The examples $z_1,\dots,z_n$ are summarized by the bag $\wr z_1,\dots,z_n\int$. We can say that the summarization is accomplished by a *summarizing function* Σ_n that maps an n-tuple of examples $(z_1,\dots,z_n)$ to the bag containing these examples:

$$\Sigma_n(z_1,\dots,z_n) := \wr z_1,\dots,z_n\int.$$

   We write σ_n for the summary—that is, the bag $\wr z_1,\dots,z_n\int$.

2. *Updating.* The summary can be formed step by step as the examples are observed. Once you have the bag containing the first $n-1$ examples, you just add the nth. This defines an *updating function* $U_n(\sigma,z)$ that satisfies

$$\Sigma_n(z_1,\dots,z_n) = U_n(\Sigma_{n-1}(z_1,\dots,z_{n-1}),z_n).$$

   The top panel in Figure 8 depicts how the summary σ_n is built up step by step from $z_1,\dots,z_n$ using the updating functions $U_1,\dots,U_n$. First $\sigma_1 = U_1(\square,z_1)$, where $\square$ is the empty bag. Then $\sigma_2 = U_2(\sigma_1,z_2)$, and so on.

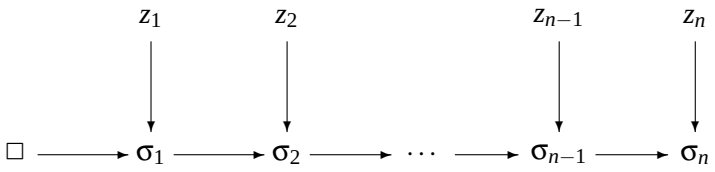
3. *Looking back all the way.* Given the bag σ_n, the $n!$ different orderings of the elements of the bag are equally likely, just as they would be if we ordered the contents of the bag randomly. As we learned in §3.2, we can say this with a formula that takes explicit account of the possibility of repetitions in the bag: the probability of the event $\{z_1 = a_1,\dots,z_n = a_n\}$ is

$$P_n(a_1,\dots,a_n \,|\, \sigma_n) = \begin{cases} \frac{n_1!\cdots n_k!}{n!} & \text{if } \wr a_1,\dots,a_n\int = \sigma_n \\ 0 & \text{if } \wr a_1,\dots,a_n\int \neq \sigma_n, \end{cases}$$

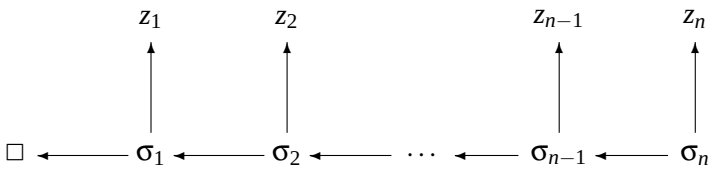
   where k is the number of distinct elements in σ_n, and $n_1,\dots,n_k$ are the numbers of times these distinct elements occur. We call $P_1,P_2,\dots$ the *full kernels*.

4. *Looking back one step.* We can also look back one step. Given the bag σ_n, what are the probabilities for z_n and σ_{n-1}? They are the same as if we drew z_n out of σ_n at random. In other words, for each z that appears in σ_n, there is a probability k/n, where k is the number of times z appears in σ_n, that (1) $z_n = z$ and (2) σ_{n-1} is the bag obtained by removing one instance of z from σ_n. The kernel defined in this way is represented by the two arrows backward from σ_n in the bottom panel of Figure 8. Let us designate it by R_n. We similarly obtain a kernel R_{n-1} backward from σ_{n-1} and so on. These are the *one-step kernels* for the model. We can obtain the full kernel P_n by combining the one-step kernels $R_n,R_{n-1},\dots,R_1$. This is most readily understood not in terms of formulas but in terms of a sequence of drawings whose outcomes have the probability distributions given by the kernels. The drawing from σ_n (which goes by the probabilities given by $R_n(\cdot\,|\,\sigma_n)$) gives us z_n and σ_{n-1}, the drawing from σ_{n-1} (which goes by the probabilities given by $R_{n-1}(\cdot\,|\,\sigma_{n-1})$) gives us z_{n-1} and σ_{n-2}, and so on; we finally obtain the whole random sequence $z_1,\dots,z_n$, which has the distribution $P_n(\cdot\,|\,\sigma_n)$. This is the meaning of the bottom panel in Figure 8.

All four operations are important. The second and fourth, updating and looking back one step, can be thought of as the most fundamental, because we can derive the other two from them. Summarization can be carried out by composing updates, and looking back all the way can be carried out by composing one-step look-backs. Moreover, the conformal algorithm uses the one-step back



Updating. We speak of "on-line" compression models because the summary can be updated with each new example. In the case of the exchangeability model, we obtain the bag σ_i by adding the new example z_i to the old bag σ_{i-1}.



Backward probabilities. The two arrows backwards from σ_i symbolize our probabilities, conditional on σ_i, for what example z_i and what previous summary σ_{i-1} were combined to produce σ_i. Like the diagram in Figure 3 that it generalizes, this diagram is a Bayes net.

Figure 8: **Elements of an on-line compression model.** The top diagram represents the updating functions $U_1,\ldots,U_n$. The bottom diagram represents the one-step kernels $R_1,\ldots,R_n$.

probabilities. But when we turn to particular on-line compression models, we will find it initially most convenient to describe them in terms of their summarizing functions and full kernels.

In general, an *on-line compression model* for an example space $\mathbf{Z}$ consists of a space $\mathbf{S}$, whose elements we call *summaries*, and two sequences of mappings:

- Updating functions $U_1, U_2, \ldots$. The function U_n maps a summary s and an example z to a new summary $U_n(s,z)$.

- One-step kernels $R_1, R_2, \ldots$. For each summary s, the kernel R_n gives a joint probability distribution $R_n(s', z \,|\, s)$ for an unknown summary s' and unknown example z. We require that $R_n(\cdot \,|\, s)$ give probability one to the set of pairs (s', z) such that $U_n(s', z) = s$.

We also require that the summary space $\mathbf{S}$ include the empty summary $\square$.

The recipes for constructing the summarizing functions $\Sigma_1, \Sigma_2, \ldots$ and the full kernels $P_1, P_2, \ldots$ are the same in general as in the exchangeability model:

- The summary $\sigma_n = \Sigma_n(z_1,\ldots,z_n)$ is built up step by step from $z_1,\ldots,z_n$ using the updating functions. First $\sigma_1 = U_1(\square, z_1)$, then $\sigma_2 = U_2(\sigma_1, z_2)$, and so on.

- We obtain the full kernel P_n by combining, backwards from σ_n, the random experiments represented by the one-step kernels $R_n, R_{n-1}, \ldots, R_1$. First we draw z_n and σ_{n-1} from $R_n(\cdot \mid \sigma_n)$, then we draw z_{n-1} and σ_{n-2} from $R_{n-1}(\cdot \mid \sigma_{n-1})$, and so on. The sequence $z_1, \ldots, z_n$ obtained in this way has the distribution $P_n(\cdot \mid \sigma_n)$.

On-line compression models are usually initially specified in terms of their summarizing functions Σ_n and their full kernels P_n, because these are usually easy to describe. One must then verify that these easily described objects do define an on-line compression model. This requires verifying two points:

1. $\Sigma_1, \Sigma_2, \ldots$ can be defined successively by means of updating functions:

$$\Sigma_n(z_1, \ldots, z_n) = U_n(\Sigma_{n-1}(z_1, \ldots, z_{n-1}), z_n).$$

   In words: σ_n depends on $z_1, \ldots, z_{n-1}$ only through the earlier summary σ_{n-1}.

2. Each P_n can be obtained as required using one-step kernels. One way to verify this is to exhibit the one-step kernels $R_1, \ldots, R_n$ and then to check that drawing z_n and σ_{n-1} from $R_n(\cdot \mid \sigma_n)$, then drawing z_{n-1} and σ_{n-2} from $R_{n-1}(\cdot \mid \sigma_{n-1})$, and so on produces a sequence $z_1, \ldots, z_n$ with the distribution $P_n(\cdot \mid \sigma_n)$. Another way to verify it, without necessarily exhibiting the one-step kernels, is to verify the conditional independence relations represented by Figure 8: z_n (and hence also σ_n) is probabilistically independent of $z_1, \ldots, z_{n-1}$ given σ_{n-1}.

5.2 Conformal Prediction

In the context of an on-line compression model, a *nonconformity measure* is an arbitrary real-valued function $A(\sigma, z)$, where σ is a summary and z is an example. We choose A so that $A(\sigma, z)$ is large when z seems very different from the examples that might be summarized by σ.

In order to state the conformal algorithm, we write $\tilde{\sigma}_{n-1}$ and $\tilde{z}_n$ for random variables with a joint probability distribution given by the one-step kernel $R(\cdot \mid \sigma_n)$. The algorithm using old examples alone can then be stated as follows:

The Conformal Algorithm Using Old Examples Alone

Input: Nonconformity measure A, significance level ε, examples $z_1, \ldots, z_{n-1}$, example z

Task: Decide whether to include z in $\gamma^\varepsilon(z_1, \ldots, z_{n-1})$.

Algorithm:

1. Provisionally set $z_n := z$.
2. Set $p_z := R_n(A(\tilde{\sigma}_{n-1}, \tilde{z}_n) \geq A(\sigma_{n-1}, z_n) \mid \sigma_n)$.
3. Include z in $\gamma^\varepsilon(z_1, \ldots, z_{n-1})$ if and only if $p_z > \varepsilon$.

To see that this reduces to the algorithm we gave for the exchangeability model on p. 385, recall that $\sigma_n = \wr z, \ldots, z_n \int$ and $\tilde{\sigma}_{n-1} = \wr z_1, \ldots, z_n \int \setminus \wr \tilde{z}_n \int$ in that model, so that

$$A(\tilde{\sigma}_{n-1}, \tilde{z}_n) = A(\wr z_1, \ldots, z_n \int \setminus \wr \tilde{z}_n \int, \tilde{z}_n) \tag{25}$$

and

$$A(\sigma_{n-1}, z_n) = A(\wr z_1, \ldots, z_{n-1} \int, z_n). \tag{26}$$

Under $R_n(\cdot \mid \{z_1,\dots,z_n\})$, the random variable $\tilde{z}_n$ has equal chances of being any of the z_i, so that the probability of (25) being greater than or equal to (26) is simply the fraction of the z_i for which

$$A(\{z_1,\dots,z_n\} \setminus \{z_i\}, z_i) \geq A(\{z_1,\dots,z_{n-1}\}, z_n),$$

and this is how p_z is defined on p. 385.

Our arguments for the validity of the regions $\gamma^\varepsilon(z_1,\dots,z_{n-1})$ in the exchangeability model generalize readily. The definitions of *n*-event and ε-rare generalize in an obvious way:

- An event E is an *n-event* if its happening or failing is determined by the value of z_n and the value of the summary σ_{n-1}.
- An *n*-event E is ε-*rare* if $R_n(E \mid \sigma_n) \leq \varepsilon$.

The event $z_n \notin \gamma^\varepsilon(z_1,\dots,z_{n-1})$ is an *n*-event, and it is ε-rare (the probability is ε or less that a random variable will take a value that it equals or exceeds with a probability of ε or less). So working backwards from the summary σ_N for a large value of N, Bill can still bet against the errors successively at rates corresponding to their probabilities under σ_n, which are always ε or less. This produces an exact analog to Informal Proposition 1:

Informal Proposition 2 *Suppose N is large, and the variables $z_1,\dots,z_N$ obey an on-line compression model. Suppose E_n is an ε-rare n-event for $n = 1,\dots,N$. Then the law of large numbers applies; with very high probability, no more than approximately the fraction ε of the events $E_1,\dots,E_N$ will happen.*

The conformal algorithm using features of the new example generalizes similarly:

The Conformal Algorithm

Input: Nonconformity measure A, significance level ε, examples $z_1,\dots,z_{n-1}$, object x_n, label y

Task: Decide whether to include y in $\Gamma^\varepsilon(z_1,\dots,z_{n-1},x_n)$.

Algorithm:

1. Provisionally set $z_n := (x_n, y)$.
2. Set $p_y := R_n(A(\tilde{\sigma}_{n-1}, \tilde{z}_n) \geq A(\sigma_{n-1}, z_n) \mid \sigma_n)$.
3. Include y in $\Gamma^\varepsilon(z_1,\dots,z_{n-1},x_n)$ if and only if $p_y > \varepsilon$.

The validity of this algorithm follows from the validity of the algorithm using old examples alone by the same argument as in the case of exchangeability.

5.3 Examples

We now look at two on-line compression models: the exchangeability-within-label model and the on-line Gaussian linear model.

The exchangeability-within-label model was first introduced in work leading up to our book (Vovk et al., 2005). It weakens the assumption of exchangeability.

The on-line Gaussian linear model, as we have already mentioned, has been widely studied. It overlaps the exchangeability model, in the sense that the assumptions for both of the models can

hold at the same time, but the assumptions for one of them can hold without the assumptions for the other holding. It is closely related to the classical Gaussian linear model. Conformal prediction in the on-line model leads to the same prediction regions that are usually used for the classical model. But the conformal prediction theory adds the new information that these intervals are valid in the sense of this article: they are right $1-\varepsilon$ of the time when used on accumulating data.

5.3.1 THE EXCHANGEABILITY-WITHIN-LABEL MODEL

The assumption of exchangeability can be weakened in many ways. In the case of classification, one interesting possibility is to assume only that the examples for each label are exchangeable with each other. For each label, the objects with that label are as likely to appear in one order as in another. This assumption leaves open the possibility that the appearance of one label might change the probabilities for the next label.

Suppose the label space has k elements, say $\mathbf{Y} = \{1,\dots,k\}$. Then we define the *exchangeability-within-label model* as follows:

Summarizing Functions The nth summarizing function is

$$\Sigma_n(z_1,\dots,z_n) := (y_1,\dots,y_n,B_1^n,\dots,B_k^n)\,,$$

where B_j^n is the bag consisting of the objects in the list $x_1,\dots,x_n$ that have the label j.

Full Kernels The full kernel $P_n(z_1,\dots,z_n \,|\, y_1,\dots,y_n,B_1^n,\dots,B_k^n)$ is most easily described in terms the random action for which it gives the probabilities: independently for each label j, distribute the objects in B_j^n randomly among the positions i for which y_i is equal to j.

To check that this is an on-line compression model, we exhibit the updating function and the one-step kernels:

Updating When (x_n,y_n) is observed, the summary

$$(y_1,\dots,y_{n-1},B_1^{n-1},\dots,B_k^{n-1})$$

is updated by inserting y_n after y_{n-1} and adding x_n to $B_{y_n}^{n-1}$.

One step back The one-step kernel R_n is given by

$$R_n(\text{summary},(x,y)\,|\,y_1,\dots,y_n,B_1^n,\dots,B_k^n) = \begin{cases} \frac{k}{|B_{y_n}^n|} & \text{if } y = y_n \\ 0 & \text{otherwise,} \end{cases}$$

where k is the number of xs in $B_{y_n}^n$. This is the same as the probability the one-step kernel for the exchangeability model would give for x on the basis of a bag of size $|B_{y_n}^n|$ that includes k xs.

Because the true labels are part of the summary, our imaginary bettor Bill can choose to bet just on those rounds of his game with Joe where the label has a particular value, and this implies that a 95% conformal predictor under the exchangeability-within-label model will make errors at no more than a 5% rate for examples with that label. This is not necessarily true for a 95% conformal predictor under the exchangeability model; although it can make errors no more than about 5% of

the time overall, its error rate may be higher for some labels and lower for others. As Figure 9 shows, this happens in the case of the USPS data set. The graph in the top panel of the figure shows the cumulative errors for examples with the label 5, which is particularly easy to confuse with other digits, when the nearest-neighbor conformal predictor is applied to that data in permuted form. The error rate for 5 is over 11%. The graph in the bottom panel shows the results of the exchangeability-within-label conformal predictor using the same nearest-neighbor nonconformity measure; here the error rate stays close to 5%. As this graph makes clear, the predictor holds the error rate down to 5% in this case by producing many prediction regions containing more than one label ("uncertain predictions").

As we explain in §4.5 and §8.4 of our book (Vovk et al., 2005), the exchangeability-within-label model is a *Mondrian model*. In general, a Mondrian model decomposes the space $\mathbf{Z} \times \mathbb{N}$, where $\mathbb{N}$ is set of the natural numbers, into non-overlapping rectangles, and it asks for exchangeability only within these rectangles. For each example z_i, it then records, as part of the summary, the rectangle into which (z_i, i) falls. Mondrian models can be useful when we need to weaken the assumption of exchangeability. They can also be attractive even if we are willing to assume exchangeability across the categories, because the conformal predictions they produce will be calibrated within categories.

5.3.2 THE ON-LINE GAUSSIAN LINEAR MODEL

Consider examples $z_1, \ldots, z_N$, of the form $z_n = (x_n, y_n)$, where y_n is a number and x_n is a row vector consisting of p numbers. For each n between 1 and N, set

$$X_n := \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad \text{and} \qquad Y_n := \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

Thus X_n is an $n \times p$ matrix, and Y_n is a column vector of length n.

In this context, the *on-line Gaussian linear model* is the on-line compression model defined by the following summarizing functions and full kernels:

Summarizing Functions The nth summarizing function is

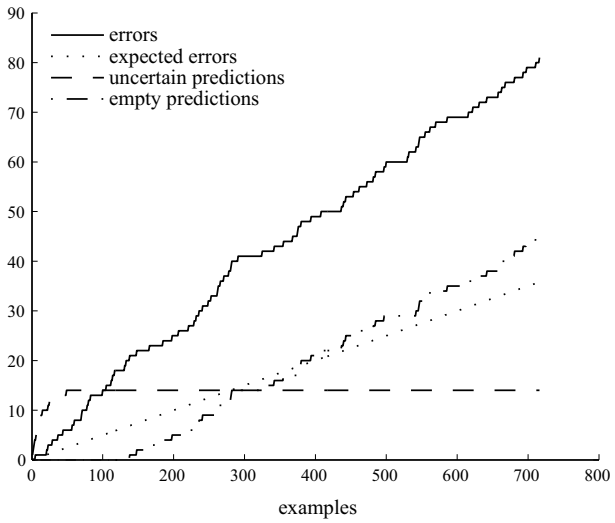
$$\begin{aligned} \Sigma_n(z_1, \ldots, z_n) &:= \left(x_1, \ldots, x_n, \sum_{i=1}^{n} y_i x_i, \sum_{i=1}^{n} y_i^2 \right) \\ &= \left(X_n, X_n' Y_n, Y_n' Y_n \right). \end{aligned}$$

Full Kernels The full kernel $P_n(z_1, \ldots, z_n \mid \sigma_n)$ distributes its probability uniformly over the set of vectors $(y_1, \ldots, y_n)$ consistent with the summary σ_n. (We consider probabilities only for $y_1, \ldots, y_n$, because $x_1, \ldots, x_n$ are fixed by σ_n.)

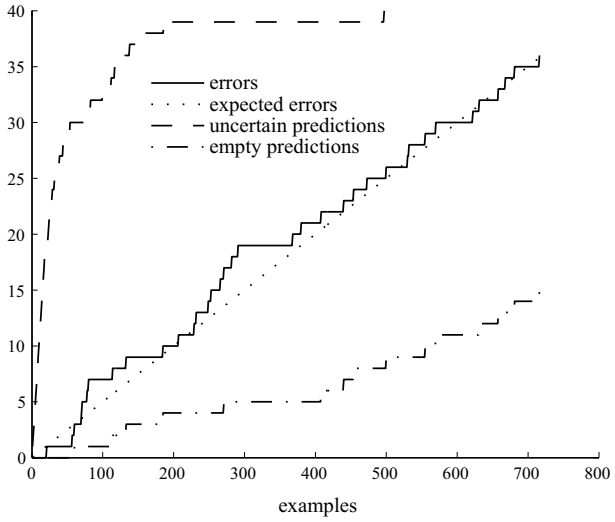
We can write $\sigma_n = (X_n, C, r^2)$, where C is a column vector of length p, and r is a nonnegative number. A vector $(y_1, \ldots, y_n)$ is consistent with σ_n if

$$\sum_{j=1}^{n} y_j x_j = C \qquad \text{and} \qquad \sum_{j=1}^{n} y_j^2 = r^2.$$

This is the intersection of a hyperplane with a sphere. Not being empty, the intersection is either a point (in the exceptional case where the hyperplane is tangent to the sphere) or a lower-dimensional



Exchangeability model



Exchangeability-within-label model

Figure 9: **Errors for** 95% **conformal prediction using nearest neighbors in the permuted USPS data when the true label is** 5**.** In both figures, the dotted line represents the overall expected error rate of 5%. The actual error rate for 5s with the exchangeability-within-label model tracks this line, but with the exchangeability model it is much higher. The exchangeability-within-label predictor keeps its error rate down by issuing more prediction regions containing more than one digit ("uncertain predictions").

sphere. (Imagine intersecting a plane and a 2-dimensional sphere; the result is a circle.) The kernel $P_n(\cdot \mid \sigma_n)$ puts all its probability on the point or distributes it uniformly over the the lower-dimensional sphere.

To see that the summarizing functions and full kernels define an on-line compression model, we must check that the summaries can be updated and that the full kernels have the required conditional independence property: conditioning $P_n(\cdot \mid \sigma_n)$ on $z_{i+1}, \ldots, z_n$ gives $P_i(\cdot \mid \sigma_i)$. (We do not condition on σ_i since it can be computed from $z_{i+1}, \ldots, z_n$ and σ_n.) Updating is straightforward; when we observe (x_n, y_n), we update the summary

$$\left(x_1, \ldots, x_{n-1}, \sum_{i=1}^{n-1} y_i x_i, \sum_{i=1}^{n-1} y_i^2 \right)$$

by inserting x_n after x_{n-1} and adding a term to each of the sums. To see that conditioning $P_n(\cdot \mid \sigma_n)$ on $z_{i+1}, \ldots, z_n$ gives $P_i(\cdot \mid \sigma_i)$, we note that conditioning the uniform distribution on a sphere on values $y_{i+1} = a_{i+1}, \ldots, y_n = a_n$ involves intersecting the sphere with the hyperplanes defined by these $n - i$ equations. This produces the uniform distribution on the possibly lower-dimensional sphere defined by

$$\sum_{j=1}^{i} y_j^2 = r^2 - \sum_{j=i+1}^{n} y_j^2 \qquad \text{and} \qquad \sum_{j=1}^{i} y_j x_j = C - \sum_{j=i+1}^{n} y_j x_j;$$

this is indeed $P_i(y_1, \ldots, y_i \mid \sigma_i)$.

The on-line Gaussian linear model is closely related to the *classical Gaussian linear model*. In the classical model,[5]

$$y_i = x_i \beta + e_i, \tag{27}$$

where the x_i are row vectors of known numbers, β is a column vector of unknown numbers (the regression coefficients), and the e_i are independent of each other and normally distributed with mean zero and a common variance. When $n - 1 > p$ and $\mathrm{Rank}(X_{n-1}) = p$, the theory of the classical model tells us the following:

- After observing examples $(x_1, y_1, \ldots, x_{n-1}, y_{n-1})$, estimate the vector of coefficients β by

  $$\hat{\beta}_{n-1} := (X'_{n-1} X_{n-1})^{-1} X'_{n-1} Y_{n-1}$$

  and after further observing x_n, predict y_n by

  $$\hat{y}_n := x_n \hat{\beta}_{n-1} = x_n (X'_{n-1} X_{n-1})^{-1} X'_{n-1} Y_{n-1}.$$

- Estimate the variance of the e_i by

  $$s_{n-1}^2 := \frac{Y'_{n-1} Y_{n-1} - \beta'_{n-1} X'_{n-1} Y_{n-1}}{n - p - 1}.$$

5. There are many names for the classical model. The name "classical Gaussian linear model" is used by Bickel and Doksum (2001, p. 366).

- The random variable

$$t_n := \frac{y_n - \hat{y}_n}{s_{n-1}\sqrt{1 + x_n'(X_{n-1}'X_{n-1})^{-1}x_n}} \tag{28}$$

has a t-distribution with $n-p-1$ degrees of freedom, and so

$$\hat{y}_n \pm t^{\varepsilon/2}_{n-p-1} s_{n-1}\sqrt{1 + x_n'(X_{n-1}'X_{n-1})^{-1}x_n} \tag{29}$$

has probability $1-\varepsilon$ of containing y_n (Ryan 1997, p. 127; Seber and Lee 2003, p. 132).

The assumption $\mathrm{Rank}(X_{n-1}) = p$ can be relaxed, at the price of complicating the formulas involving $(X_{n-1}'X_{n-1})^{-1}$. But the assumption $n-1 > \mathrm{Rank}(X_{n-1})$ is essential to finding a prediction interval of the type (29); when it fails there are values for the coefficients β such that $y_{n-1} = X_{n-1}\beta$, and consequently there is no residual variance with which to estimate the variance of the e_i.

We have already used two special cases of (29) in this article. Formula (1) in §2.1.1 is the special case with $p = 1$ and each x_i equal to 1, and formula (21) at the beginning of §4.3.2 is the special case with $p = 2$ and the first entry of each x_i equal to 1.

The relation between the classical and on-line models, fully understood in the theoretical literature since the 1980s, can be summarized as follows:

- If $z_1, \ldots, z_N$ satisfy the assumptions of the classical Gaussian linear model, then they satisfy the assumptions of the on-line Gaussian linear model. In other words, the assumption that the errors e_i in (27) are independent and normal with mean zero and a common variance implies that conditional on $X_n'Y_n = C$ and $Y_n'Y_n = r^2$, the vector Y_n is distributed uniformly over the sphere defined by C and r^2. This was already noted by R. A. Fisher in 1925.

- The assumption of the on-line Gaussian linear model, that conditional on $X_n'Y_n = C$ and $Y_n'Y_n = r^2$, the vector Y_n is distributed uniformly over the sphere defined by C and r^2, is sufficient to guarantee that (28) has the t-distribution with $n-p-1$ degrees of freedom (Dempster, 1969; Efron, 1969).

- Suppose $z_1, z_2, \ldots$ is an infinite sequence of random variables. Then $z_1, \ldots, z_N$ satisfy the assumptions of the on-line Gaussian linear model for every integer N if and only if the joint distribution of $z_1, z_2, \ldots$ is a mixture of distributions given by the classical Gaussian linear model, each model in the mixture possibly having a different β and a different variance for the e_i (Lauritzen, 1988).

A natural nonconformity measure A for the on-line Gaussian linear model is given, for $\sigma = (X, X'Y, Y'Y)$ and $z = (x, y)$, by

$$A(\sigma, z) := |y - \hat{y}|, \tag{30}$$

where $\hat{y} = x(X'X)^{-1}X'Y$.

Proposition 2 *When (30) is used as the nonconformity measure, the $1-\varepsilon$ conformal prediction region for y_n is (29), the interval given by the t-distribution in the classical theory.*

Proof When (30) is used as the nonconformity measure, the test statistic $A(\sigma_{n-1}, z_n)$ used in the conformal algorithm becomes $|y_n - \hat{y}_n|$. The conformal algorithm considers the distribution of this

statistic under $R_n(\cdot \mid \sigma_n)$. But when σ_n is fixed and t_n is given by (28), $|t_n|$ is a monotonically increasing function of $|y_n - \hat{y}_n|$ (see Vovk et al., 2005, pp. 202–203, for details). So the conformal prediction region is the interval of values of y_n for which $|t_n|$ does not take its most extreme values. Since t_n has the t-distribution with $n-p-1$ degrees of freedom under $R_n(\cdot \mid \sigma_n)$, this is the interval (29). ■

Together with Informal Proposition 2, Proposition 2 implies that when we use (29) for a large number of successive values of n, y_n will be in the interval $1-\varepsilon$ of the time. In fact, because the probability of error each time is exactly ε, we can say simply that the errors are independent and for this reason the classical law of large numbers applies.

In our example involving the prediction of petal width from sepal length, the exchangeability and Gaussian linear models gave roughly comparable results (see Table 6 in §4.3.2). This will often be the case. Each model makes an assumption, however, that the other does not make. The exchangeability model assumes that the *x*s, as well as the *y*s, are exchangeable. The Gaussian linear model assumes that given the *x*s, the *y*s are normally distributed.

Acknowledgments

This article is based on a tutorial lecture by Glenn Shafer at the Workshop on Information Theory and Applications at the University of California at San Diego on February 1, 2007. Glenn Shafer would like to thank Yoav Freund and Alon Orlitsky for the invitation to participate in this workshop. We thank Wei Wu for his assistance in computations. Many other intellectual debts are acknowledged in the preface of our book (Vovk et al., 2005).

Appendix A. Validity

The main purpose of this appendix is to formalize and prove the following informal proposition:

Informal Proposition 1 *Suppose N is large, and the variables $z_1, \ldots, z_N$ are exchangeable. Suppose E_n is an ε-rare n-event for $n = 1, \ldots, N$. Then the law of large numbers applies; with very high probability, no more than approximately the fraction ε of the events $E_1, \ldots, E_N$ will happen.*

We used this informal proposition in §3.4 to establish the validity of conformal prediction in the exchangeability model. As we promised then, we will discuss two different approaches to formalizing it: a classical approach and a game-theoretical approach. The classical approach shows that the E_n are mutually independent in the case where they are exactly ε-rare and then appeals to the classical weak law of large numbers for independent events. The game-theoretic approach appeals directly to the more flexible game-theoretic weak law of large numbers.

Our proofs will also establish the analogous Informal Proposition 2, which we used to establish the validity of conformal prediction in on-line compression models in general.

In §A.3, we return to R. A. Fisher's prediction interval for a normal random variable, which we discussed in §2.1.1. We show that this prediction interval's successive hits are independent, so that validity follows from the usual law of large numbers. Fisher's prediction interval is a special case of conformal prediction for the Gaussian linear model, and so it is covered by the general result

for on-line compression models. But the proof in §A.3, being self-contained and elementary and making no reference to conformal prediction, may be especially informative for many readers.

A.1 A Classical Argument for Independence

Recall the definitions we gave in §3.4 in the case where $z_1,\dots,z_N$ are exchangeable: An event E is an *n-event* if its happening or failing is determined by the value of z_n and the value of the bag $\{z_1,\dots,z_{n-1}\}$, and an n-event E is ε-*rare* if $\Pr(E \mid \{z_1,\dots,z_n\}) \le \varepsilon$. Let us further say that n-event E is *exactly* ε-*rare* if

$$\Pr(E \mid \{z_1,\dots,z_n\}) = \varepsilon. \tag{31}$$

The conditional probability in this equation is a random variable, depending on the random bag $\{z_1,\dots,z_n\}$, but the equation says that it is not really random, for it is always equal to ε. Its expected value, the unconditional probability of E, is therefore also equal to ε.

Proposition 3 *Suppose E_n is an exactly ε-rare n-event for $n = 1,\dots,N$. Then $E_1,\dots,E_N$ are mutually independent.*

Proof Consider (31) for $n = N-1$:

$$\Pr(E_{N-1} \mid \{z_1,\dots,z_{N-1}\}) = \varepsilon. \tag{32}$$

Given $\{z_1,\dots,z_{N-1}\}$, knowledge of z_N does not change the probabilities for z_{N-1} and $\{z_1,\dots,z_{N-2}\}$, and z_{N-1} and $\{z_1,\dots,z_{N-2}\}$ determine the $(N-1)$-event E_{N-1}. So adding knowledge of z_N will not change the probability in (32):

$$\Pr(E_{N-1} \mid \{z_1,\dots,z_{N-1}\} \ \&\ z_N) = \varepsilon.$$

Because E_N is determined by z_N once $\{z_1,\dots,z_{N-1}\}$ is given, it follows that

$$\Pr(E_{N-1} \mid \{z_1,\dots,z_{N-1}\} \ \&\ E_N) = \varepsilon,$$

and from this it follows that $\Pr(E_{N-1} \mid E_N) = \varepsilon$. The unconditional probability of E_{N-1} is also ε. So E_N and E_{N-1} are independent. Continuing the reasoning backwards to E_1, we find that the E_n are all mutually independent. ■

This proof generalizes immediately to the general case of on-line compression models (see p. 407); we simply replace $\{z_1,\dots,z_n\}$ with σ_n.

If N is sufficiently large, and E_n is an exactly ε-rare n-event for $n = 1,\dots,N$, then the law of large numbers applies; with very high probability, no more than approximately the fraction ε of the N events will happen. It is intuitively clear that this conclusion will also hold if we have an inequality instead of an equality in (31), because making the E_n even less likely to happen cannot reverse the conclusion that few of them will happen.

The preceding argument is less than rigorous on two counts. First, the proof of Proposition 3 does not consider the existence of the conditional probabilities it uses. Second, the argument from the case where (31) is an equality to that where it is merely an inequality, though entirely convincing, is only intuitive. A fully rigorous proof, which uses Doob's measure-theoretic framework to deal with the conditional probabilities and uses a randomization to bring the inequality up to an equality, is provided on pp. 211–213 of our book (Vovk et al., 2005).

A.2 A Game-theoretic Law of Large Numbers

As we explained in §3.3, the game-theoretic interpretation of exchangeability involves a backward-looking protocol, in which Bill observes first the bag $\{z_1,\dots,z_N\}$ and then successively z_N, z_{N-1}, and so on, finally observing z_1. Just before he observes z_n, he knows the bag $\{z_1,\dots,z_n\}$ and can bet on the value of z_n at odds corresponding to the probabilities the bag determines:

$$\Pr(z_n = a \mid \{z_1,\dots,z_n\} = B) = \frac{k}{n}, \tag{33}$$

where k is the number of times a occurs in B.

THE BACKWARD-LOOKING BETTING PROTOCOL
Players: Joe, Bill
$\mathcal{K}_N := 1$.
Joe announces a bag B_N of size N.
FOR $n = N, N-1, \dots, 2, 1$
Bill bets on z_n at odds set by (33).
Joe announces $z_n \in B_n$.
$\mathcal{K}_{n-1} := \mathcal{K}_n +$ Bill's net gain.
$B_{n-1} := B_n \setminus \{z_n\}$.

Bill's initial capital $\mathcal{K}_N$ is 1. His final capital is $\mathcal{K}_0$.

Given an event E, set

$$e := \begin{cases} 1 & \text{if } E \text{ happens} \\ 0 & \text{if } E \text{ fails.} \end{cases}$$

Given events $E_1,\dots,E_N$, set

$$\text{Freq}_N := \frac{1}{N}\sum_{j=1}^{N} e_j.$$

This is the fraction of the events that happen—the frequency with which they happen. Our game-theoretic law of large numbers will say that if each E_n is an ε-rare n-event, then it is very unlikely that Freq_N will substantially exceed ε.

In game-theoretic probability, what do we mean when we say an event E is "very unlikely"? We mean that the bettor, Bill in this protocol, has a betting strategy that guarantees

$$\mathcal{K}_0 \geq \begin{cases} C & \text{if } E \text{ happens} \\ 0 & \text{if } E \text{ fails,} \end{cases} \tag{34}$$

where C is a large positive number. Cournot's principle, which says that Bill will not multiply his initial unit capital by a large factor without risking bankruptcy, justifies our thinking E unlikely. The larger C, the more unlikely E. We call the quantity

$$\overline{\text{P}}E := \inf\left\{ \frac{1}{C} \;\middle|\; \text{Bill can guarantee (34)} \right\} \tag{35}$$

E's *upper probability*. An unlikely event is one with small upper probability.

Proposition 4 (Game-theoretic weak law of large numbers) *Suppose E_n is an ε-rare n-event, for $n = 1,\dots,N$. Suppose $\varepsilon < 1/2$, $\delta_1 > 0$, $\delta_2 > 0$, and $N \geq 1/\delta_1\delta_2^2$. Then*

$$\overline{\mathrm{P}}(\mathrm{Freq}_N \geq \varepsilon + \delta_2) \leq \delta_1.$$

In words: If N is sufficiently large, there is a small (less than δ_1) upper probability that the frequency will exceed ε substantially (by more than δ_2).

Readers familiar with game-theoretic probability will recognize Proposition 4 as a form of the game-theoretic weak law of large numbers (Shafer and Vovk, 2001, pp. 124–126). The bound it gives for the upper probability of the event $\mathrm{Freq}_N \geq \varepsilon + \delta_2$ is the same as the bound that Chebyshev's inequality gives for the probability of this event in classical probability theory when the E_n are independent and all have probability ε.

For the benefit of those not familiar with the concepts of game-theoretic probability used in the proof just cited, we now give an elementary and self-contained proof of Proposition 4.

Lemma 5 *Suppose, for $n = 1,\dots,N$, that E_n is an ε-rare n-event. Then Bill has a strategy that guarantees that his capital $\mathcal{K}_n$ will satisfy*

$$\mathcal{K}_n \geq \frac{n}{N} + \frac{1}{N}\left(\left(\sum_{j=n+1}^{N}(e_j - \varepsilon)\right)^+\right)^2 \tag{36}$$

for $n = 1,\dots,N$, where $t^+ := \max(t,0)$.

Proof When $n = N$, (36) reduces to $\mathcal{K}_N \geq 1$, and this certainly holds; Bill's initial capital $\mathcal{K}_N$ is equal to 1. So it suffices to show that if (36) hold for n, then Bill can bet on E_n in such a way that the corresponding inequality for $n-1$,

$$\mathcal{K}_{n-1} \geq \frac{n-1}{N} + \frac{1}{N}\left(\left(\sum_{j=n}^{N}(e_j - \varepsilon)\right)^+\right)^2, \tag{37}$$

also holds. Here is how Bill bets.

- If $\sum_{j=n+1}^{N}(e_j - \varepsilon) \geq \varepsilon$, then Bill buys $(2/N)\sum_{j=n+1}^{N}(e_j - \varepsilon)$ units of e_n. By assumption, he pays no more than ε for each unit. So we have a lower bound on his net gain, $\mathcal{K}_{n-1} - \mathcal{K}_n$:

$$\begin{aligned}
\mathcal{K}_{n-1} - \mathcal{K}_n &\geq \frac{2}{N}\left(\sum_{j=n+1}^{N}(e_j - \varepsilon)\right)(e_n - \varepsilon) \\
&\geq \frac{1}{N}\left(\sum_{j=n}^{N}(e_j - \varepsilon)\right)^2 - \frac{1}{N}\left(\sum_{j=n+1}^{N}(e_j - \varepsilon)\right)^2 - \frac{1}{N} \\
&\geq \frac{1}{N}\left(\left(\sum_{j=n}^{N}(e_j - \varepsilon)\right)^+\right)^2 - \frac{1}{N}\left(\left(\sum_{j=n+1}^{N}(e_j - \varepsilon)\right)^+\right)^2 - \frac{1}{N}.
\end{aligned} \tag{38}$$

Adding (38) and (36), we obtain (37).

- If $\sum_{j=n+1}^{N}(e_j-\varepsilon)<\varepsilon$, then Bill does not bet at all, and $\mathcal{K}_{n-1}=\mathcal{K}_n$. Because

$$\left(\left(\sum_{j=n}^{N}(e_j-\varepsilon)\right)^+\right)^2-\left(\left(\sum_{j=n+1}^{N}(e_j-\varepsilon)\right)^+\right)^2\le(\varepsilon+(e_n-\varepsilon))^2\le 1,$$

  we again obtain (37) from (36).

■

Proof of Proposition 4 The inequality $\mathrm{Freq}_N\ge\varepsilon+\delta_2$ is equivalent to

$$\frac{1}{N}\left(\left(\sum_{j=1}^{N}(e_j-\varepsilon)\right)^+\right)^2\ge N\delta_2^2. \tag{39}$$

Bill's strategy in Lemma 5 does not risk bankruptcy (it is obvious that $\mathcal{K}_n\ge 0$ for all n), and (36) says

$$\mathcal{K}_0\ge\frac{1}{N}\left(\left(\sum_{j=1}^{N}(e_j-\varepsilon)\right)^+\right)^2. \tag{40}$$

Combining (39) and (40) with the assumption that $N\ge 1/\delta_1\delta_2^2$, we see that when the event $\mathrm{Freq}_N\ge\varepsilon+\delta_2$ happens, $\mathcal{K}_0\ge 1/\delta_1$. So by (34) and (35), $\overline{\mathrm{P}}(\mathrm{Freq}_N\ge\varepsilon+\delta_2)\le\delta_1$. ■

A.3 The Independence of Hits for Fisher's Interval

Recall that if $z_1,\dots,z_n,z_{n+1}$ are independent normal random variables with mean 0 and standard deviation 1, the distribution of the ratio

$$\frac{z_{n+1}}{\sqrt{\sum_{i=1}^{n}z_i^2/n}} \tag{41}$$

is called the *t-distribution with n degrees of freedom*. The upper percentile points for this distribution, the points t_n^ε exceeded by (41) with probability exactly ε, are readily available from textbooks and standard computer programs.

Given a sequence of numbers $z_1,\dots,z_l$, where $l\ge 2$, we set

$$\bar{z}_l:=\frac{1}{l}\sum_{i=1}^{l}z_i\qquad\text{and}\qquad s_l^2:=\frac{1}{l-1}\sum_{i=1}^{l}(z_i-\bar{z}_l)^2.$$

As we recalled in §2.1.1, R. A. Fisher proved that if $n\ge 3$ and $z_1,\dots,z_n$ are independent and normal with a common mean and standard deviation, then the ratio t_n given by

$$t_n:=\frac{z_n-\bar{z}_{n-1}}{s_{n-1}}\sqrt{\frac{n-1}{n}} \tag{42}$$

has the t-distribution with $n-2$ degrees of freedom (Fisher, 1935). It follows that the event

$$\bar{z}_{n-1} - t_{n-2}^{\varepsilon/2}\, s_{n-1} \sqrt{\frac{n}{n-1}} \le z_n \le \bar{z}_{n-1} + t_{n-2}^{\varepsilon/2}\, s_{n-1} \sqrt{\frac{n}{n-1}} \tag{43}$$

has probability $1-\varepsilon$. We will now prove that the t_n for successive n are independent. This implies that the events (43) for successive values of n are independent, so that the law of large numbers applies: with very high probability approximately $1-\varepsilon$ of these events will happen. This independence was overlooked by Fisher and subsequent authors.

We begin with two purely arithmetic lemmas, which do not rely on any assumption about the probability distribution of $z_1, \dots, z_n$.

Lemma 6 *The ratio t_n given by (42) depends on $z_1, \dots, z_n$ only through the ratios among themselves of the differences*

$$z_1 - \bar{z}_n, \dots, z_n - \bar{z}_n.$$

Proof It is straightforward to verify that

$$z_n - \bar{z}_{n-1} = \frac{n}{n-1}\,(z_n - \bar{z}_n) \tag{44}$$

and

$$s_{n-1}^2 = \frac{(n-1)s_n^2}{n-2} - \frac{n(z_n-\bar{z}_n)^2}{(n-1)(n-2)}. \tag{45}$$

Substituting (44) and (45) in (42) produces

$$t_n = \frac{\sqrt{n(n-2)}(z_n-\bar{z}_n)}{\sqrt{(n-1)^2 s_n^2 - n(z_n-\bar{z}_n)^2}} \tag{46}$$

or

$$t_n = \frac{\sqrt{n(n-2)}(z_n-\bar{z}_n)}{\sqrt{(n-1)\sum_{i=1}^n (z_i-\bar{z}_n)^2 - n(z_n-\bar{z}_n)^2}}. \tag{47}$$

The value of (47) is unaffected if all the $z_i - \bar{z}_n$ are multiplied by a nonzero constant. ■

Lemma 7 *Suppose $\bar{z}_n$ and s_n are known. Then the following three additional items of information are equivalent, inasmuch as the other two can be calculated from any of the three:*

1. z_n
2. $\bar{z}_{n-1}$ *and* s_{n-1}
3. t_n

Proof Given z_n, we can calculate $\bar{z}_{n-1}$ and s_{n-1} from (44) and (45) and then calculate t_n from (42). Given $\bar{z}_{n-1}$ and s_{n-1}, we can calculate z_n from (44) or (45) and then t_n from (42). Given t_n, we can invert (46) to find z_n (when $\bar{z}_n$ and s_n are fixed, this equation expresses t_n as a monotonically increasing function of z_n) and then calculate $\bar{z}_{n-1}$ and s_{n-1} from (44) and (45). ■

Now we consider probability distributions for $z_1, \dots, z_n$.

Lemma 8 *If $z_1,\dots,z_n$ are independent and normal with a common mean and standard deviation, then conditional on $\bar{z}_n = w$ and $\sum_{i=1}^n (z_i - \bar{z}_n)^2 = r^2$, the vector $(z_1,\dots,z_n)$ is distributed uniformly over the $(n-2)$-dimensional sphere of vectors satisfying these conditions (the intersection of the hyperplane $\bar{z}_n = w$ with the $(n-1)$-dimensional sphere of radius r centered on the point $(w,\dots,w)$ in $\mathbb{R}^n$).*

Proof The logarithm of the joint density of $z_1,\dots,z_n$ is

$$-\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^n (z_i-\mu)^2 = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left(\sum_{i=1}^n (z_i-\bar{z}_n)^2 + n(\bar{z}_n-\mu)^2\right), \quad (48)$$

where μ and σ are the mean and standard deviation, respectively. Because this depends on $(z_1,\dots,z_n)$ only through $\bar{z}_n$ and $\sum_{i=1}^n (z_i-\bar{z}_n)^2$, the distribution of $(z_1,\dots,z_n)$ conditional on $\bar{z}_n = w$ and $\sum_{i=1}^n (z_i - \bar{z}_n)^2 = r^2$ is uniform over the set of vectors satisfying these conditions. ■

Lemma 9 *If the vector $(z_1,\dots,z_n)$ is distributed uniformly over the $(n-2)$-dimensional sphere defined by the conditions $\bar{z}_n = w$ and $\sum_{i=1}^n (z_i-\bar{z}_n)^2 = r^2$, then t_n has the t-distribution with $n-2$ degrees of freedom.*

Proof The distribution of t_n does not depend on w or r. This is because we can transform the uniform distribution over one $(n-2)$-dimensional sphere in $\mathbb{R}^n$ into a uniform distribution over another by adding a constant to all the z_i and then multiplying the differences $z_i - \bar{z}_n$ by a constant, and by Lemma 6, this will not change t_n.

Now suppose $z_1,\dots,z_n$ are independent and normal with a common mean and standard deviation. Lemma 8 says that conditional on $\bar{z}_n = w$ and $(n-1)s_n^2 = r^2$, the vector $(z_1,\dots,z_n)$ is distributed uniformly over the sphere of radius r centered on $w,\dots,w$. Since the resulting distribution for t_n does not depend on w or r, it must be the same as the unconditional distribution. ■

Lemma 10 *Suppose $(z_1,\dots,z_N)$ is distributed uniformly over the $N-2$-dimensional sphere defined by the conditions $\bar{z}_n = w$ and $\sum_{i=1}^n (z_i-\bar{z}_n)^2 = r^2$. Then $t_3,\dots,t_N$ are mutually independent.*

Proof It suffices to show that t_n still has the t-distribution with $n-2$ degrees of freedom conditional on $t_{n+1},\dots,t_N$. This will imply that t_n is independent of $t_{n+1},\dots,t_N$ and hence that all the t_n are mutually independent.

We start knowing $\bar{z}_N = r$ and $s_N = w$. So by Lemma 7, learning $t_{n+1},\dots,t_N$ is the same as learning $z_{n+1},\dots,z_N$. Geometrically, when we learn z_N we intersect our $(N-1)$-dimensional sphere in $\mathbb{R}^N$ with a hyperplane, reducing it to an $(N-2)$-dimensional sphere in $\mathbb{R}^{N-1}$. (Imagine, for example, intersecting a sphere in $\mathbb{R}^3$ with a plane: the result is a circle.) When we learn z_{N-1}, we reduce the dimension again, and so on. In each case, we obtain a uniform distribution on the lower-dimensional sphere for the remaining z_i. In the end, we find that $(z_1,\dots,z_n)$ is distributed uniformly over an $(n-1)$-dimensional sphere in $\mathbb{R}^n$, and so t_n has the t-distribution with $n-2$ degrees of freedom by Lemma 9. ■

Proposition 11 *Suppose $z_1, \ldots, z_N$ are independent and normal with a common mean and standard deviation. Then $t_3, \ldots, t_N$ are mutually independent.*

Proof By Lemma 10, $t_3, \ldots, t_N$ are mutually independent conditional on $\bar{z}_N = w$ and $s_N = r$, each t_n having the t-distribution with $n-2$ degrees of freedom. Because this joint distribution for $t_3, \ldots, t_N$ does not depend on w or r, it is also their unconditional joint distribution. ■

References

Eugene A. Asarin. Some properties of Kolmogorov δ-random finite sequences. *Theory of Probability and its Applications*, 32:507–508, 1987.

Eugene A. Asarin. On some properties of finite objects random in the algorithmic sense. *Soviet Mathematics Doklady*, 36:109–112, 1988.

Peter J. Bickel and Kjell A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics*. Volume I. Prentice Hall, Upper Saddle River, New Jersey, second edition, 2001.

Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, New York, 1999.

Emanuel Czuber. *Wahrscheinlichkeitsrechnung und ihre Anwendung auf Fehlerausgleichung, Statistik, und Lebensversicherung*. Volume I. Wahrscheinlichkeitstheorie, Fehlerausgleichung, Kollektivmasslehre. Teubner, Leipzig and Berlin, third edition, 1914.

Arthur P. Dempster. *Elements of Continuous Multivariate Analysis*. Addison-Wesley, Reading, Massachusetts, 1969.

Norman R. Draper and Harry Smith. *Applied Regression Analysis*. Wiley, New York, third edition, 1998.

Bradley Efron. Student's t-test under symmetry conditions. *Journal of the American Statistical Association*, 64:1278–1302, 1969.

Ronald A. Fisher. Applications of Student's distribution. *Metron*, 5:90–104, 1925.

Ronald A. Fisher. The fiducial argument in statistical inference. *Annals of Eugenics*, 6:391–398, 1935.

Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

Ronald A. Fisher. *Statistical Methods and Scientific Inference*. Hafner, New York, third edition, 1973. First edition, 1956.

Alex Gammerman and Vladimir Vovk. Hedging predictions in machine learning. *Computer Journal*, 50:151–163, 2007.

Edwin Hewitt and Leonard J. Savage. Symmetric measures on Cartesian products. *Transactions of the American Mathematical Society*, 80:470–501, 1955.

Steffen L. Lauritzen. *Extremal Families and Systems of Sufficient Statistics*. Volume 49 of *Lecture Notes in Statistics*. Springer, New York, 1988.

Erich L. Lehmann. *Testing Statistical Hypotheses*. Wiley, New York, second edition, 1986. First edition, 1959.

Per Martin-Löf. Repetitive structures and the relation between canonical and microcanonical distributions in statistics and statistical mechanics. In Ole E. Barndorff-Nielsen, Preben Blæsild, and Geert Schou, editors, *Proceedings of Conference on Foundational Questions in Statistical Inference*, pages 271–294, Aarhus, 1974. Memoirs 1.

Jerzy Neyman. Outline of a theory of statistical estimation based on the classical theory of probability. *Philosophical Transactions of the Royal Society, Series A*, 237:333–380, 1937.

Geoff K. Robinson. Some counterexamples to the theory of confidence intervals. *Biometrika*, 62: 155–161, 1975.

Thomas P. Ryan. *Modern Regression Methods*. Wiley, New York, 1997.

George A. F. Seber and Alan J. Lee. *Linear Regression Analysis*. Wiley, Hoboken, NJ, second edition, 2003.

Glenn Shafer. From Cournot's principle to market efficiency. In Jean-Philippe Touffut, editor, *Augustin Cournot, Economic Models and Rationality*. Edward Elgar, Cheltenham, 2007.

Glenn Shafer and Vladimir Vovk. *Probability and Finance: It's only a Game!* Wiley, New York, 2001.

Dylan S. Small, Joseph L. Gastwirth, Abba M. Krieger, and Paul R. Rosenbaum. R-estimates vs. GMM: A theoretical case study of validity and efficiency. *Statistical Science*, 21:363–375, 2006.

Student (William S. Gossett). On the probable error of a mean. *Biometrika*, 6:1–25, 1908.

John W. Tukey. Sunset salvo. *American Statistician*, 40:72–76, 1986.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, New York, 2005.

Journal of Machine Learning Research 9 (2008) 423-457 Submitted 6/07; Revised 12/07; Published 3/08

Theoretical Advantages of Lenient Learners: An Evolutionary Game Theoretic Perspective

Liviu Panait LIVIU@GOOGLE.COM
Google Inc.
Santa Monica, CA 90401, USA

Karl Tuyls K.TUYLS@MICC.UNIMAAS.NL
Maastricht University
MiCC-IKAT, The Netherlands

Sean Luke SEAN@CS.GMU.EDU
Department of Computer Science
George Mason University
Fairfax, VA 22030, USA

Editor: Leslie Pack Kaelbling

Abstract

This paper presents the dynamics of multiple learning agents from an evolutionary game theoretic perspective. We provide replicator dynamics models for cooperative coevolutionary algorithms and for traditional multiagent Q-learning, and we extend these differential equations to account for lenient learners: agents that forgive possible mismatched teammate actions that resulted in low rewards. We use these extended formal models to study the convergence guarantees for these algorithms, and also to visualize the basins of attraction to optimal and suboptimal solutions in two benchmark coordination problems. The paper demonstrates that lenience provides learners with more accurate information about the benefits of performing their actions, resulting in higher likelihood of convergence to the globally optimal solution. In addition, the analysis indicates that the choice of learning algorithm has an insignificant impact on the overall performance of multiagent learning algorithms; rather, the performance of these algorithms depends primarily on the level of lenience that the agents exhibit to one another. Finally, the research herein supports the strength and generality of evolutionary game theory as a backbone for multiagent learning.

Keywords: multiagent learning, reinforcement learning, cooperative coevolution, evolutionary game theory, formal models, visualization, basins of attraction

1. Introduction

Multiagent learning is a relatively new research area that has witnessed a significant research effort in the past decade. Work in this area is motivated by the wide-applicability of multiagent systems to a large set of complex real-world problem domains, as well as by the difficulty of hard-coding solutions for such distributed approaches (Panait and Luke, 2005a). Multiagent learning techniques have been successfully applied to domains such as multi-rover coordination (Tumer and Agogino, 2007; Knudson and Tumer, 2007), congestive games (Agogino and Tumer, 2006), air traffic control (Agogino and Tumer, 2007), and spacecraft power management (Airiau et al., 2006), to name but a few. Moreover, the formal analysis in Jansen and Wiegand (2003, 2004) demonstrated that these

techniques' strengths rely on the decomposition of the problem into simpler subproblems that can be explored separately, resulting at times in significant speedups over traditional learning algorithms. It is no surprise, then, that such techniques have also been successfully applied to problems outside of the multiagent systems realm, such as the discovery of cascade neural networks for classification tasks (Potter and Jong, 2000), the optimization of inventory control (Eriksson and Olsson, 1997), and the coevolution of neural networks for pole balancing (Gomez et al., 2006).

Multiagent learning is, however, a very challenging problem. Agents typically have limited knowledge of even which actions their teammates are currently performing, and must act not only in a constantly changing environment but one which may be actively *co-adapting* to the agents' actions, often in ways they cannot fully perceive (Panait and Luke, 2005a). Even when agents are notionally *cooperating* with one another, their limited perception capabilities can give rise to phenomena that hinder the agents' ability to coordinate effectively. Such dynamics are still not well understood formally.

Recent research has highlighted a tendency for multiagent learning algorithms to converge to suboptimal solutions, which is in marked contrast to the well-established convergence guarantees of their single-agent counterparts. For example, straightforward extensions of Q-learning to multi-agent systems fail to reach the optimal policy in fairly simple domains (Claus and Boutilier, 1998; Kapetanakis and Kudenko, 2002), despite Q-learning's convergence guarantees in single-agent domains (Sutton and Barto, 1998; Watkins and Dayan, 1992). Similarly, cooperative coevolutionary algorithms Potter and Jong (1994); Potter (1997) can not only converge but also be *attracted to* joint suboptima in the search space (Wiegand, 2004), despite traditional evolutionary algorithms being guaranteed to converge to the global optimum when given enough time and sufficient random exploration (Vose, 1999). Different reasons account for this: other agents are learning as well; the reinforcement an agent receives depends on the actions taken by the other agents; and not all information is observable. Such features make it very difficult to engineer learning algorithms capable of finding optimal solutions under these conditions.

This paper presents a theoretical foundation for multiagent learning in coordination games with no internal state; we only consider symmetric games where all agents receive equal reward. This foundation may be directly applied to two popular algorithms: cooperative coevolutionary algorithms (CCEAs) and multiagent Q-learning. These two techniques are representative of the two families (evolutionary algorithms and multiagent reinforcement learning) which form the bulk of the cooperative multiagent learning field (Panait and Luke, 2005a). In both of these algorithms it is common for an agent to base its assessment of a particular action on the expected reward received for that action. For example, a reinforcement learning agent usually updates its estimate for an action's utility every time[1] that action is performed in the environment, despite the dependence of the reward on the actions performed by the other agents. The formal analysis in this paper demonstrates that the reinforcement provided by the expected reward might be the primary cause for many observed problems with multiagent learning.

The paper applies a new theoretical framework to argue for a different approach to these problems, in which each agent tends to *ignore* lower rewards garnered by a particular action. We call this approach *lenience*: each agent shows lenience to its teammates by ignoring low rewards due to actions chosen by teammates that are poor matches to the agent's current action. For example, consider a simple scenario where agents A_1 and A_2 learn to play soccer together. Let's assume that they

1. This mechanism is essentially equivalent (over time) to using the expected reward.

do not have any *a priori* information about each other's skills. At some point, A_1 attempts a pass that should allow A_2 to receive the ball in a very favorable position (for example, alone with the goalie). However, A_2 is not able to receive the ball properly because he was not expecting the pass, or simply because A_2 has not learned the ball reception skill. Given the failure of their collaboration in this particular instance, should A_1 assume that the pass was bad in general and that it should be avoided in the future (due to the low reward the agents just received)? This is the approach usually taken by straightforward extensions of single-agent learning algorithms to multiagent domains. Alternatively, we argue that A_1 might benefit from showing lenience towards his teammate by completely ignoring the low reward observed following A_2's mistake, and only hope that A_2 will perform better once it learns to play better. In other words, A_1 might assume that the pass was good, and that the low reward is primarily due to A_2 not having learned the skill yet. With time, A_1 will be able to distinguish between good and bad passes based on A_2's future performance.

Lenience is particularly important early in the game, when the agents have not yet identified high-performing actions and so the average reward of a given action can be misleadingly low due to partnering with poor-performing joint actions. In previous work, we have demonstrated the efficacy of algorithms based on this notion: specifically, the Lenient Multiagent Q-learning algorithm (Panait et al., 2006) and the iCCEA algorithm (Panait and Luke, 2006). Here, we will establish a more formal justification for the effectiveness of lenience in multiagent learning, accompanied by an intuitive visualization of the impact that lenience has onto the tendency of these algorithms to converge to suboptimal solutions.

The remainder of this paper is structured as follows. Section 2 introduces basic formal models and background material, and describe the pathologies. In Section 3 we argue that lenience will benefit agents in overcoming such problems. Sections 4 and 5 present enhanced theoretical models of the multiagent learning paradigms. Finally, Section 6 concludes the paper and suggests directions for future research.

2. Background

This paper presents a formal analysis of two multiagent learning algorithms: cooperative coevolutionary algorithms and multiagent Q-learning. Sections 2.1.1 and 2.1.2 briefly describe each of them. Following, Sections 2.2.1 and 2.2.2 introduce the basics of the replicator equations both in discrete and continuous time. Sections 2.3.1 and 2.3.2 present two evolutionary game theory models that capture the dynamics of the two learning algorithms, that is, CCEA and multiagent Q-learning. These replicator dynamics (RD) models represent the foundation for our work and will be extended in Section 3 to account for lenience. More details on both models can be found in Wiegand (2004), Tuyls et al. (2006) and Tuyls et al. (2003).

2.1 Multiagent Learning Methods

This section provides a brief overview of two multiagent learning algorithms.

2.1.1 COOPERATIVE COEVOLUTIONARY ALGORITHMS

Evolutionary algorithms are beam search methods that employ and refine sets of candidate solutions to a given problem. The terminology in the field is borrowed from biology to reflect the main source of inspiration (Darwin's models for evolution) for these techniques; for example, candidate

solutions are termed *individuals*, sets of individuals are known as *populations*, and the creation of new solutions from previous ones is usually referred to as *breeding*.

An evolutionary algorithm begins with an initial population of random individuals. Each member of this population is then evaluated and assigned a *fitness* (a quality assessment). The algorithm then selects a set of fitter-than-average parents and alters them via *crossover* and *mutation* operations to produce a set of children, which are added to the population, replacing older, less fit individuals. One evaluation, selection, and breeding cycle is known as a *generation*. Successive generations continue to refine the population until time is exhausted, or until a sufficiently fit individual is discovered. For additional information, interested readers are referred to Jong (2006).

Coevolution is an intuitive extension of evolutionary algorithms for domains with multiple learning agents. Cooperative coevolutionary algorithms (CCEAs) are a popular approach for domains where agents succeed or fail together (Husbands and Mill, 1991; Potter and Jong, 1994). Here, each agent is given its own population[2] of individuals, each such individual representing a possible behavior for that agent. CCEAs usually assume that only the performance of the entire team can be assessed; as a consequence, one individual (behavior) for each of the team members is required for evaluation. Fitness assessment works as follows: for each individual i in an agent's population, one or more tuples are formed using i and individuals (termed *collaborators*) from the other agents' populations (either from the current generation, or from the previous one). These collaborators can be sampled randomly or selected based on performance (if the collaborators were evaluated in the previous generation, the better ones might be used preferentially). The fitness of i is then computed based on the performance obtained from evaluating the teams with the behaviors indicated by these tuples. As a consequence, the fitness assessment is context-sensitive and subjective. Aside from evaluation, the learning processes are independent of one another. The populations evolve concurrently (this is closer to the asynchronous nature of multiagent systems). Potter (1997), Potter and Jong (2000) and Wiegand (2004) provide comprehensive overviews of cooperative coevolutionary algorithms.

2.1.2 MULTIAGENT Q-LEARNING

Q-learning (Watkins, 1989; Sutton and Barto, 1998) is particularly useful in domains where reinforcement information (expressed as penalties or rewards) is stochastic and is observed after a sequence of actions has been performed. The algorithm associates a utility (or *Q*uality) Q with each (s,a) pair, where s is a state of the environment, and a is an action that the agent can perform when in state s. The agent updates the Q-values at each time step based on the reinforcement it has received. The update formula is

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left(r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right)$$

where the agent has just been in state s_t, has performed action a_t, has observed reward r_t, and has transitioned to the new (current) state s_{t+1}; α is the learning rate, and γ is a discount factor (bounded between 0 and 1) for incorporating future rewards into the utility estimate. Note that $Q(s_t,a_t)$ does not only depend on the immediate reward r_{t+1} observed by the agent, but also on future rewards (via the $\max_a Q(s_{t+1},a)$ term). The discount factor γ controls the impact of future versus immediate

2. Given this assumption that each agent has its own population, we sometimes use the term *population* instead of *agent* when the context deems it more appropriate.

rewards onto the *Q*uality terms: lower values of γ reduce the impact of future rewards onto the quality estimate for a state-action pair.

Action selection in Q-learning is usually based on a stochastic process; popular choices include ε-*greedy exploration* (select the best action with probability $1-\varepsilon$, or a random action otherwise) and *Boltzmann exploration* (the selection process further depends on a "temperature" parameter). As previously mentioned, Q-learning has certain guarantees of convergence under theoretical conditions, which include performing each action an infinite number of times in each state, as well as proper settings for the learning rate (Singh et al., 2000; Watkins and Dayan, 1992).

Multiagent Q-learning is a straightforward extension of Q-learning to domains involving multiple agents (Claus and Boutilier, 1998). Here, each agent is given its own Q-value function for (s,a) pairs. All agents choose their actions independently and concurrently, perform them in parallel, and observe the same reward associated with the joint action. In contrast to single-agent Q-learning, the information an agent observes depends on the actions chosen by other agents. The reward information is then used by each agent to update its Q-values. Agents usually cannot perceive which actions their teammates have chosen.

To simplify the theoretical analysis, we assume that agents choose actions by using the Boltzmann selection: an action a_k is chosen with probability

$$P(a_k) = \frac{e^{\frac{Q(s,a_k)}{\tau}}}{\sum_i e^{\frac{Q(s,a_i)}{\tau}}} \tag{1}$$

where τ is a temperature parameter used to balance exploration and exploitation (the agent tends to select actions associated with higher utilities when τ is low).

2.2 The Replicator Dynamics

John Maynard Smith introduced Evolutionary Game Theory (EGT) by applying traditional game theory to Biology (Smith, 1982; Maynard Smith and Price, 1973). EGT is particularly well-suited as a model of cooperative learning agents. In EGT, each agent holds a vector of proportions over possible actions, indicating the proportion of "individuals" or "replicators" in an infinite "population" (the agent) which have adopted a given action. EGT also commonly presumes one or more matrixes that represent the performance of actions in the context of actions of the other agents. Each timestep, the proportions of actions for each agent are changed based on the rewards in the matrix as well as on the current probabilities of other agents' choosing their actions.

Replicator dynamics (RD) is a key concept in EGT to express the adaptation of each population over time. RD can be represented either in discrete or continuous time. We elaborate on both forms of RD, starting with the discrete time version. In essence the replicator equations describe how a population of different actions evolves through time. An abstraction of an evolutionary process usually combines two basic elements: selection and mutation. Selection favors some population actions over others, while mutation provides variety in the population. The most basic form of RD only highlights the role of selection, that is, how the most fit actions in a population are selected.

2.2.1 DISCRETE TIME REPLICATOR DYNAMICS

Suppose there is a single population of actions (or replicators) and consider a discrete time process $t = 1,2,\ldots$. Let $A = (a_{ij})_{i,j=1}^{n}$ be the reward matrix ($a_{ij} \geq 0$ is the reward for the joint action (i,j),

and n is the total number of possible actions). Let us assume that the individuals in the population (considered infinite) represent different actions that the agent can perform. The state of the population can be described by a vector $x(t) = (x_1(t), ..., x_n(t))$, where $x_i(t)$ denotes the proportion of individual i in the population (it is also directly related to the probability that the agent will select action i).

At each time step t, the state $x(t)$ of the population changes according to the fitness values of the different individuals. More precisely, the expected number of offsprings for a single individual representing action i equals the ratio between the expected payoff for such an individual, $\sum_{j=1}^{n} a_{ij}x_j(t)$, and the average payoff $\sum_{k=1}^{n} x_k(t)\left(\sum_{j=1}^{n} a_{kj}x_j(t)\right)$ for all individuals in the population. Therefore, the ratio $x_i(t+1)$ of individuals representing action i at the next time step equals:

$$x_i(t+1) = x_i(t)\frac{\sum_{j=1}^{n} a_{ij}x_j(t)}{\sum_{k=1}^{n} x_k(t)\left(\sum_{j=1}^{n} a_{kj}x_j(t)\right)}.$$

Dividing both sides by $x_i(t)$ leads to:

$$\frac{x_i(t+1)}{x_i(t)} = \frac{\sum_{j=1}^{n} a_{ij}x_j(t)}{\sum_{k=1}^{n} x_k(t)\left(\sum_{j=1}^{n} a_{kj}x_j(t)\right)}.$$

A minor manipulation of the fractions leads to the general discrete time replicator dynamics:

$$\frac{\Delta x_i(t)}{x_i(t)} = \frac{\sum_{j=1}^{n} a_{ij}x_i(t) - x(t)Ax(t)}{x(t)Ax(t)}.$$

This system of equations expresses Darwinian selection as follows: the rate of change Δx_i of a particular action i is the difference between its average payoff, $\sum_{j=1}^{n} a_{ij}x_i(t)$, and the average payoffs for all actions, $x(t)Ax(t)$.

2.2.2 CONTINUOUS TIME REPLICATOR DYNAMICS

From a mathematical viewpoint, it is usually more convenient to work in continuous time. Therefore we now derive the continuous version of the above replicator equations. To do this we make time infinitesimal. More precisely, suppose that the amount of time that passes between two periods is given by δ with $\delta \in [0,1]$. We also assume that during a time period δ, only a fraction δ of the individuals die and generate offspring. Then the above equation can be rewritten as follows:

$$\frac{\Delta x_i(t)}{x_i(t)} = \frac{x_i(t+\delta) - x_i(t)}{x_i(t)} = \frac{\sum_{j=1}^{n} \delta a_{ij}x_i(t) - x(t)\delta Ax(t)}{x(t)\delta Ax(t) + (1-\delta)}.$$

Now, at the limit when δ approaches 0, the continuous replicator equations are:

$$\frac{dx_i}{dt} = \left[\sum_{j=1}^{n} a_{ij}x_i - x \cdot Ax\right] x_i$$

which can be rewritten as:

$$\frac{dx_i}{dt} = [(Ax)_i - x \cdot Ax]\, x_i. \tag{2}$$

In Equation 2, x_i represents the density of action i in the population, and A is the payoff matrix that describes the different payoff values that each individual replicator receives when interacting with other replicators in the population. The state x of the population can be described as a probability vector $x = (x_1, x_2, ..., x_n)$ which expresses the different densities of all the different types of replicators in the population. Hence $(Ax)_i$ is the payoff that replicator i receives in a population with state x and $x \cdot Ax$ describes the average payoff in the population. The growth rate $\frac{\frac{dx_i}{dt}}{x_i}$ of the proportion of action i in the population equals the difference between the action's current payoff and the average payoff in the population. Gintis (2001), Hofbauer and Sigmund (1998) and Weibull (1996) detail further information on this issues.

2.3 EGT Models for Multiagent Learning Algorithms

This paper is concerned with formal models of multiple agents that learn concurrently. For simplicity, the discussion focuses on only two such learning agents in a symmetric game (both agents receive the same payoff). As a result, we need two systems of differential equations, one for each agent. This setup corresponds to an RD for asymmetric games, where the available actions or strategies of the agents to belong to two different population. For example, consider the case of extending the continuous RD model to a multiagent learning scenario involving a two-player symmetric game with two agents, and consider that A^T is the payoff matrix that describes the reinforcements received by the second agent. Then, another equation similar to Equation 2 would emerge again to characterize the dynamics of the second learner.

This translates into the following replicator equations for the two populations:

$$\frac{dp_i}{dt} = [(Aq)_i - p \cdot Aq]p_i, \tag{3}$$

$$\frac{dq_i}{dt} = [(A^T p)_i - q \cdot A^T p]q_i. \tag{4}$$

Equations 3 and 4 indicate that the growth rate of the types in each population is additionally determined by the composition of the other population, in contrast to the single population (learner) case described by Equation 2.

Next, we present two EGT models for multiagent learning. First, Section 2.3.1 presents a model for cooperative coevolutionary algorithms, which is based on the discrete time RD model in Section 2.2.1. Then, Section 2.3.2 presents a model for multiagent Q-learning, which is based on the continuous time RD model in Section 2.2.2.

2.3.1 EGT MODEL FOR COOPERATIVE COEVOLUTIONARY ALGORITHMS

Wiegand (2004) describes an evolutionary game theoretic model for cooperative coevolutionary algorithms. The model applies to stateless domains involving only two agents, and where each agent has a finite set of actions to choose from. The two populations (one per agent) contain individuals, each of them representing an action that the agent might perform. The model further assumes that the two populations are infinite, and it computes the proportions of individuals in the populations at each generation. If the first agent has a finite number n of distinct actions to choose from, its population at each generation is an element of the set $\Delta^n = \{x \in [0,1]^n \mid \sum_{i=1}^n x_i = 1\}$. A higher proportion x_i indicates that the agent is more likely to choose action i. Given that the second agent might have a different set (of size m) of actions to choose from, its population is an element of the

set $\Delta^m = \left\{ y \in [0,1]^m \mid \sum_{j=1}^m y_j = 1 \right\}$. The fitness computation in the EGT model involves the game payoff matrix A, where the a_{ij} element represents the reward for the joint action (i, j). Assuming both agents are equally rewarded, A is used to compute the fitnesses in one population, and the transpose of A is used for the other population (as in Section 2.3). Equations 5–8 describe the EGT model for CCEAs, as expressed in Wiegand (2004):

$$u_i^{(t)} = \sum_{j=1}^{m} a_{ij} y_j^{(t)}, \tag{5}$$

$$w_j^{(t)} = \sum_{i=1}^{n} a_{ij} x_i^{(t)}, \tag{6}$$

$$x_i^{(t+1)} = \left(\frac{u_i^{(t)}}{\sum_{k=1}^{n} x_k^{(t)} u_k^{(t)}} \right) x_i^{(t)}, \tag{7}$$

$$y_j^{(t+1)} = \left(\frac{w_j^{(t)}}{\sum_{k=1}^{m} y_k^{(t)} w_k^{(t)}} \right) y_j^{(t)} \tag{8}$$

where $x^{(t)}$ and $y^{(t)}$ represent the proportions of genotypes (actions) in the two populations at generation t, and $x^{(t+1)}$ and $y^{(t+1)}$ represent the new proportions at the next generation $t+1$. For simplicity, the equations only model the dynamics of cooperative coevolutionary systems under the pressure of selection.

The EGT model can be separated into two phases. First, the fitness of each action is computed for the two populations (Equations 5 and 6). The fitness of action i is estimated as the mean payoff over pairwise collaborations with every action in the other population. Note that this computation uses the proportions of each action in the (infinite) populations. Second, the distributions of the two populations at the next generation are computed (Equations 7 and 8 model the effects of fitness proportional selection).

2.3.2 EGT MODEL FOR MULTIAGENT Q-LEARNING

This section introduces the RD model of Multiagent Q-learning. For a complete mathematical derivation of it, please refer to Tuyls et al. (2006, 2003). Basically, the derivation boils down to constructing a continuous time limit of the Q-learning model (in the same manner as when going from the discrete replicator equations to the continuous version), where Q-values are interpreted as Boltzmann probabilities for the action selection. For simplicity, we only consider games between two agents, and assume that the game is stateless: the reward that the agents receive depends solely on the actions they have currently performed in the environment. We admit up front that such scenarios are unrealistic for practical purposes, but they are complex enough to emphasize specific challenges for multiagent reinforcement learning algorithms. For this reason, several previous empirical investigations of these techniques have employed such domains (for example, by Claus and Boutilier 1998, Kapetanakis and Kudenko 2002 and Lauer and Riedmiller 2000).

Each agent has a probability vector over its action set, more precisely $x_1, \ldots, x_n$ over action set $a_1, \ldots, a_n$ for the first agent and $y_1, \ldots, y_m$ over $b_1, \ldots, b_m$ for the second agent. Equation 1 can also be rewritten as follows:

$$x_i(k) = \frac{e^{\frac{Qa_i(k)}{\tau}}}{\sum_{j=1}^{n} e^{\frac{Qa_j(k)}{\tau}}}$$

where $x_i(k)$ is the probability of playing action i at time step k and τ is the temperature. Now we can find an expression for, $x_i(k+1)$:

$$\begin{aligned}\frac{x_i(k+1)}{x_i(k)} &= \frac{e^{\frac{Qa_i(k+1)}{\tau}}\sum_j e^{\frac{Qa_j(k)}{\tau}}}{e^{\frac{Qa_i(k)}{\tau}}\sum_j e^{\frac{Qa_j(k+1)}{\tau}}} \\ &= \frac{e^{\frac{\Delta Qa_i(k)}{\tau}}}{\sum_j x_j(k) e^{\frac{\Delta Qa_j(k)}{\tau}}}\end{aligned}$$

from which follows that

$$x_i(k+1) = x_i(k)\frac{e^{\frac{\Delta Qa_i(k)}{\tau}}}{\sum_j x_j(k) e^{\frac{\Delta Qa_j(k)}{\tau}}}.$$

Considering the difference equation for x_i, we have that

$$\begin{aligned}x_i(k+1) - x_i(k) &= \frac{x_i(k) e^{\frac{\Delta Qa_i(k)}{\tau}}}{\sum_j x_j(k) e^{\frac{\Delta Qa_j(k)}{\tau}}} - x_i(k) \\ &= x_i(k)\left(\frac{e^{\frac{\Delta Qa_i(k)}{\tau}} - \sum_j x_j(k) e^{\frac{\Delta Qa_j(k)}{\tau}}}{\sum_j x_j(k) e^{\frac{\Delta Qa_j(k)}{\tau}}}\right).\end{aligned}$$

For the continuous time version, suppose that the amount of time that passes between two repetitions of the game is given by δ with $0 < \delta \leq 1$. The variable $x_i(k\delta)$ describes the x-values at time $k\delta = t$. Under these assumptions, it follows that

$$\begin{aligned}\frac{x_i(k\delta+\delta) - x_i(k\delta)}{\delta} &= \frac{x_i(k\delta)}{\delta\sum_j x_j(k\delta) e^{\frac{\Delta Qa_j(k\delta)}{\tau}}} \times \\ &\quad \left(e^{\frac{\Delta Qa_i(k\delta)}{\tau}} - \sum_j x_j(k\delta) e^{\frac{\Delta Qa_j(k\delta)}{\tau}}\right).\end{aligned}$$

We are interested in the limit with $\delta \to 0$. At a given time $t \geq 0$, the state of the limit process is computed by taking the limit of $x_i(k\delta)$ with $\delta \to 0$ and $k\delta \to t$.

$$\begin{aligned}\lim_{\delta\to 0}\frac{\Delta x_i(k\delta)}{\delta} &= \lim_{\delta\to 0}\frac{x_i(k\delta)}{\sum_j x_j(k\delta) e^{\frac{\Delta Qa_j(k\delta)}{\tau}}} \times \\ &\quad \lim_{\delta\to 0}\left(\frac{e^{\frac{\Delta Qa_i(k\delta)}{\tau}}}{\delta} - \frac{\sum_j x_j(k\delta) e^{\frac{\Delta Qa_j(k\delta)}{\tau}}}{\delta}\right). \qquad (9)\end{aligned}$$

The first limit equals $x_i(t)$ (or x_i for a shorter notation):

$$\lim_{\delta\to 0}\frac{x_i(k\delta)}{\sum_j x_j(k\delta) e^{\frac{\Delta Qa_j(k\delta)}{\tau}}} = x_i(t) \qquad (10)$$

because the exponent term becomes 1 ($\Delta Q_{a_j}(k\delta)$ becomes 0) and $\sum_j x_j(k\delta)$ equals 1(sum of all probabilities equals 1). Therefore,

$$\lim_{\delta\to 0}\frac{\Delta x_i(k\delta)}{\delta} = x_i \times \lim_{\delta\to 0}\left(\frac{e^{\frac{\Delta Q_{a_i}(k\delta)}{\tau}}}{\delta} - \frac{\sum_j x_j(k\delta)e^{\frac{\Delta Q_{a_j}(k\delta)}{\tau}}}{\delta}\right).$$

The second limit is undefined (this is a $\frac{0}{0}$ situation), which allows us to use L'Hôpital's rule. The second term now equals

$$\lim_{\delta\to 0}\left(\frac{e^{\frac{\Delta Q_{a_i}(k\delta)}{\tau}}}{\delta} - \frac{\sum_j x_j(k\delta)e^{\frac{\Delta Q_{a_j}(k\delta)}{\tau}}}{\delta}\right) = \frac{dQ_{a_i}(t)}{\tau dt} - \sum_j x_j(t)\frac{dQ_{a_j}(t)}{\tau dt}. \quad (11)$$

The total limit now becomes (by combining Equations 9 – 11),

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{1}{\tau}\left(\frac{dQ_{a_i}}{dt} - \sum_j \frac{dQ_{a_j}}{dt}x_j\right). \quad (12)$$

This completes the derivation of the continuous time model for the Q-learning process. As can be seen in Equation 12, we need an expression for $\frac{dQ_{a_i}(t)}{dt}$. This can be derived the differential equation for the Q-function by performing the following steps.

The Q-learning update rule for the first agent can be written as follows:

$$Q_{a_i}(k+1) = Q_{a_i}(k) + \alpha(r_{a_i}(k+1) + \gamma\max_{a_i} Q - Q_{a_i}(k))$$

which implies,

$$\Delta Q_{a_i}(k) = \alpha(r_{a_i}(k+1) + \gamma\max_{a_i} Q - Q_{a_i}(k)).$$

This expression is the difference equation for the Q-function. To make this equation infinitesimal, going from discrete steps to a continuous version, we suppose that the amount of time that passes between two repetitions of updates of the Q-values is given by δ with $0 < \delta \leq 1$. The variable $Q_{a_i}(k\delta)$ describes the Q-values at time $k\delta$. It follows that

$$\Delta Q_{a_i}(k\delta) = \alpha(r_{a_i}((k+1)\delta) + \gamma\max_{a_i} Q - Q_{a_i}(k\delta))((k+1)\delta - k\delta)$$

which is the same as writing

$$\Delta Q_{a_i}(k\delta) = \alpha(r_{a_i}((k+1)\delta) + \gamma\max_{a_i} Q - Q_{a_i}(k\delta))\delta.$$

We are interested in the limit $\delta \to 0$. The state of the limit process at some time $t \geq 0$ (which we keep fixed) can be computed by taking the limit of $Q_{a_i}(k\delta)$ with $\delta \to 0$ and $k\delta \to t$. Bringing δ to the left side, and taking the limit for $\delta \to 0$, it follows that

$$\frac{dQ_{a_i}}{dt} = \alpha(r_{a_i} + \gamma\max_{a_i} Q - Q_{a_i}). \quad (13)$$

Now, substituting Equation 13 in Equation 12 yields

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{1}{\tau}\alpha\left(r_{a_i} - \sum_j x_j r_{a_j} - Q_{a_i} + \sum_j Q_{a_j} x_j\right)$$

because $\sum_j x_j = 1$, this expression becomes

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{\alpha}{\tau}\left(r_{a_i} - \sum_j x_j r_{a_j} + \sum_j x_j (Q_{a_j} - Q_{a_i})\right).$$

Given that $\frac{x_j}{x_i}$ equals $\frac{e^{\frac{Q_{a_j}}{\tau}}}{e^{\frac{Q_{a_i}}{\tau}}}$, it follows that

$$\alpha \sum_j x_j ln \frac{x_j}{x_i} = \frac{\alpha}{\tau} \sum_j x_j (Q_{a_j} - Q_{a_i})$$

which gives us

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{\alpha}{\tau}\left(r_{a_i} - \sum_j x_j r_{a_j}\right) + \alpha \sum_j x_j ln\left(\frac{x_j}{x_i}\right).$$

Let us put this equation in the context of the two concurrent learning agents. For clarity, we use u_i (instead of r_{a_i}) to indicate the expected reward that the first agent expects for performing action i, and w_j to indicate the expected reward that the second agent expects for performing action j. Remember that the payoff matrix for the second agent is simply the transposed payoff matrix used by the first agent.

Thus $u_i = \sum_k a_{ik} y_k$ is the expected reward that would be observed by the first agent, given the other agent's current probabilities of selecting among its actions (and similarly $w_j = \sum_k a_{kj} x_k$). The RD model above therefore has the simpler form,

$$u_i = \sum_k a_{ik} y_k, \tag{14}$$

$$w_j = \sum_k a_{kj} x_k, \tag{15}$$

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{\alpha}{\tau}\left(u_i - \sum_k x_k u_k\right) + \alpha \sum_k x_k \ln \frac{x_k}{x_i}, \tag{16}$$

$$\frac{\frac{dy_j}{dt}}{y_j} = \frac{\alpha}{\tau}\left(w_j - \sum_k y_k w_k\right) + \alpha \sum_k y_k \ln \frac{y_k}{y_j}. \tag{17}$$

2.4 The Relative Overgeneralization Pathology

Wiegand et al. (2002) used the EGT model in Equations 5–8 to provide a formal analysis for the properties of cooperative coevolutionary algorithms. Note that the model described is deterministic: given the composition of the two populations, the model can be used to compute the precise distributions of genotypes in the populations at the next generation (the stochastic nature of evolutionary algorithms is compensated for by the use of infinite populations). Using this model, Wiegand et al.

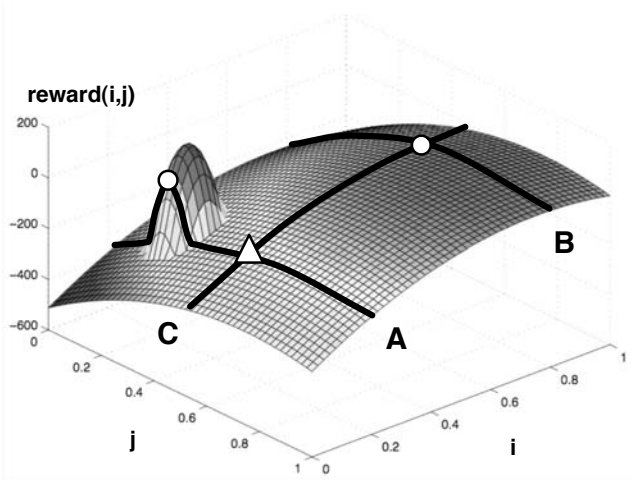


Figure 1: The relative generalization pathology in multiagent cooperative learning. The axes i and j are the various actions afforded to two different agents, and the axis $reward(i, j)$ is the reward received from a given joint action (higher is better).

(2002) showed that pure Nash equilibria are stable, attracting fixed points. This implies that starting from certain initial configurations and iterating the model until convergence may result in convergence to suboptimal Nash equilibria. A suboptimal Nash equilibrium is one that is Pareto-dominated by at least one other Nash equilibrium. This pathology, termed *relative overgeneralization* in Wiegand (2004), indicates that CCEAs have a tendency to move not to points of optimum reward, but rather to points which are "jacks of all trades but masters of none." This argument is taken further in Wiegand and Potter (2006) to suggest that cooperative coevolutionary algorithms might not necessarily search for solutions of high quality, but rather for robust solutions.

To illustrate how this can occur, consider a trivial scenario involving two agents, each with a set of actions. Both agents receive the same reward based on the joint action selected, and the objective of the two agents is to arrive on the joint action that optimizes their joint reward. Let i and j be the spaces of actions for the two agents. Figure 1 shows one possible joint space of actions $\langle i, j \rangle$ and their resulting reward. The optimum is at the top of the small peak; but it has been shown that a dynamical system such as in Equations 5 – 8 will tend to converge to the top of the broad suboptimal peak (Wiegand, 2004). The reason is as follows. Line A represents the possible rewards for the first population following action i_A when paired with various actions j from the other population. Some of these rewards are near the optimum; but most of them are below the suboptimal rewards that action i_B (line B) receives. If the utility of an action is based on *average reward*, the utility for i_B is higher that that of i_A, even though i_A has the potential to form better solutions (the circle on the higher peak in Figure 1). As a result, the system moves towards solutions more in line with i_B. This is the relative overgeneralization pathology.

Similar issues were reported in the multiagent Q-learning literature as well. For example, Fulda and Ventura (2007) attribute the convergence to suboptimal solutions to what they term *action shadowing*. Action i shadows action j if both of the following conditions apply. First, the rewards ob-

served by the agent upon performing action i over a period of time are higher than the ones observed for action j during the same period of time. This results in an expected higher utility for performing action i, as opposed to performing action j. Second, the reward obtained for j when the teammate selects a specific action k is higher than the reward obtain for i with any action selected by the teammate. For example, the domain illustrated in Figure 1 presents a scenario where action i_B shadows action i_A. This occurs because of low rewards received for i_A with poorly-matched actions (such as j_C) selected by the teammate. The circles represent how good actions i_A and i_B could be (when matched by specific actions of the teammate), while the triangle illustrates the low rewards due to action j_C that contribute to action shadowing (action shadowing is the result of many such actions).

Note informally that both relative overgeneralization and action shadowing might be solved by using the same approach: basing an action's utility on the *maximum* reward over some large number N of interactions with actions chosen by the other player. As N grows, the utility of an action approaches the joint reward it would produce were it paired with its optimum collaborating action; and thus each population could select from its actions knowing the true optimum potential of each.

This general notion of using the maximum is at the center of what we call *lenient learning*. The remainder of the paper demonstrates the advantages of being lenient in a multiagent learning system.

3. Lenient Learners

Lenient learning attempts to address the primary pathologies described in Section 2.4 by allowing agents to ignore the lower rewards received for a given action, notionally due to poorly-matched actions from the teammate, and instead concentrate on the higher rewards seen so far for that action.

Consider the Climb and Penalty domains introduced in Claus and Boutilier (1998) and used in previous investigations of multiagent Q-learning (Kapetanakis and Kudenko, 2002; Lauer and Riedmiller, 2000) and cooperative coevolution (Panait et al., 2003). The joint payoff matrices for these coordination games are defined as:

$$Climb: \begin{bmatrix} 11 & -30 & 0 \\ -30 & 7 & 6 \\ 0 & 0 & 5 \end{bmatrix}, \qquad Penalty: \begin{bmatrix} 10 & 0 & -10 \\ 0 & 2 & 0 \\ -10 & 0 & 10 \end{bmatrix}.$$

Observe that Climb has two Nash equilibria, the optimum $(1,1)$ (both agents play their first strategy: the indices of the matrix) and the suboptimum $(2,2)$ (both agents play their second strategy), and that it is strongly deceptive. Suboptimal meaning that $(2,2)$ is Pareto-dominated by $(1,1)$. Penalty has three Nash equilibria: $(1,1)$, $(2,2)$, and $(3,3)$. Of them, $(2,2)$ is suboptimal (Pareto-dominated), but is more forgiving if one or the other population deviates from the Nash equilibrium. This means that $(2,2)$ risk dominates $(1,1)$ and $(3,3)$, because playing action 2 will provide a higher expected payoff when an agent is uncertain about the other agent's action,.

Now consider two agents learning the Climb coordination game. Given that the agents have just started to learn and have no knowledge about the structure of the payoff matrix, we may assume that the agents choose their actions randomly. The expected reward for each action an agent might perform is:

$$\begin{aligned}
ExpectedReward(a_1) = &\quad 11*\tfrac{1}{3} - 30*\tfrac{1}{3} + 0*\tfrac{1}{3} &= -6.33\\
ExpectedReward(a_2) = &\quad -30*\tfrac{1}{3} + 7*\tfrac{1}{3} + 6*\tfrac{1}{3} &= -5.67\\
ExpectedReward(a_3) = &\quad 0*\tfrac{1}{3} + 0*\tfrac{1}{3} + 5*\tfrac{1}{3} &= 1.67
\end{aligned}$$

Observe that action a_1 has the lowest expected reward. The agent might thus become less likely to choose action a_1 due to the lower rewards it receives at early stages of learning. This is problematic, however, as a_1 corresponds to the optimal Nash equilibrium. The same applies to the Penalty coordination problem:

$$\begin{aligned}
ExpectedReward(a_1) = &\quad 10*\tfrac{1}{3} + 0*\tfrac{1}{3} - 10*\tfrac{1}{3} &= 0\\
ExpectedReward(a_2) = &\quad 0*\tfrac{1}{3} + 2*\tfrac{1}{3} + 0*\tfrac{1}{3} &= 0.67\\
ExpectedReward(a_3) = &\quad -10*\tfrac{1}{3} + 0*\tfrac{1}{3} + 10*\tfrac{1}{3} &= 0
\end{aligned}$$

One solution to remedy this problem is to allow agents to show lenience towards their teammates: the agents can *ignore* lower rewards observed upon performing their actions, and only update the utilities of actions based on the higher rewards. This can be achieved if the learners compare the observed reward with the estimated utility for an action, and update the utility only if it is lower than the reward. In some sense, such an approach changes the learning focus from performing as well as possible in the context of the current (likely mediocre) teammate, to performing as well as possible in the context of improved behaviors for its teammate (as the teammate learns, it likely will not choose those actions that resulted in lower rewards!).

Panait et al. (2006) presents evidence on the advantages of such an approach. The paper introduced a time-dependent degree of lenience: the agents start by ignoring most of the low rewards that they observe, but as learning progresses, agents tend to explore certain "better" actions and also ignore fewer low rewards. The paper also presented empirical evidence that several multiagent learning paradigms can significantly benefit from agents being lenient to one another. Similarly, cooperative coevolutionary algorithms often use agents that ignore lower rewards (for example, in Wiegand et al., 2001), and we previously demonstrated the advantages of a time-dependent degree of lenience for cooperative coevolution (Panait and Luke, 2005b). Although such algorithms have shown their strengths in empirical analysis, formal models of multiagent learning have always been one step behind.

This paper focuses on the mathematical foundations of a simpler approach to lenient learners: each agent collects the rewards it receives for performing actions. Upon observing N rewards for an action, only the maximum of these N rewards is used to update the utility associated with that action. The set of observed rewards for that action is cleared, and the learning process continues. In other words, the agents employ a fixed degree of lenience (which is determined by N) to their teammates. The more low rewards the agents ignore (the higher N is), the more lenient the agents can be said to be. Although this approach does not model any existing multiagent learning algorithm, its formal analysis has two major contributions. First, it provides a valuable addition to our set of tools to study such multiagent learning algorithms, and also strengthens evolutionary game theory as a primary framework for such analysis. Second, it adds to our understanding of the causes for multiagent learning algorithms drifting away from globally optimal solutions.

Next, we extend the EGT models from Section 2.3 such that each learner uses only the maximum of N rewards (it ignores the lower $N-1$ rewards) to improve its utility estimate. The following theorem establishes a key result for the formal model of lenient learners.

Theorem 1 *Let $(a_{ij})_{i,j=1}^{n}$ be the payoff matrix (a_{ij} is the payoff received by the agents when one performs action i and the other performs action j), and let $(p_j)_{j\in 1..n}$ be the probability that the teammate selects action j. The expected maximum payoff for i over N pairwise combinations with actions $j_1 \ldots j_N$ chosen with replacement according to $(p_j)_{j\in 1..n}$ is*

$$\sum_{j=1}^{n} a_{ij} \frac{p_j}{\sum_{k:a_{ik}=a_{ij}} p_k} \left(\left(\sum_{k:a_{ik}\leq a_{ij}} p_k \right)^N - \left(\sum_{k:a_{ik}<a_{ij}} p_k \right)^N \right).$$

Proof

The expected maximum reward of action i over N pairwise combinations can be expressed as a weighted sum of all possible rewards a_{ij} that action i can receive with different actions j for the teammate. The weight of each term a_{ij} equals the probability that a_{ij} is the maximum reward observed over N trials. This probability can be computed based on the difference between the probability of receiving N rewards that are no higher than a_{ij}, which equals $\left(\sum_{k:a_{ik}\leq a_{ij}} p_k\right)^N$, minus the probability of receiving N rewards that are strictly lower than a_{ij}, which equals $\left(\sum_{k:a_{ik}<a_{ij}} p_k\right)^N$. Observe that an action i might receive the same reward in combination with different actions of the teammate, and as a result, there is an extra weight $\frac{p_j}{\sum_{k:a_{ik}=a_{ij}} p_k}$ that computes the probability of observing the reward a_{ij} in combination with action j of the teammate, out of all the other actions the teammate might have selected and would have resulted in the same reward. ■

Using Theorem 1, let us compute the expected reward for a lenient learner that ignores the lower of two observed rewards for each of its actions. We assume again that the teammate selects actions at random. The expected rewards in the Climb coordination problem are:

$$\begin{aligned}
ExpectedReward(a_1) = &\quad 11*\tfrac{5}{9} - 30*\tfrac{1}{9} + 0*\tfrac{1}{3} &&= 2.78\\
ExpectedReward(a_2) = &\quad -30*\tfrac{1}{9} + 7*\tfrac{5}{9} + 6*\tfrac{1}{3} &&= 2.56\\
ExpectedReward(a_3) = &\quad 0*\tfrac{2}{9} + 0*\tfrac{2}{9} + 5*\tfrac{5}{9} &&= 2.78
\end{aligned}$$

Observe that a_1 has the highest reward (tied with that of a_3). Were the agent to be more lenient (for example, if it ignored the lower two of three observed rewards), the expected reward for a_1 would be significantly higher than those for a_2 and a_3. Similarly, the expected rewards in the Penalty domain (with learners ignoring the lower of every two rewards) are:

$$\begin{aligned}
ExpectedReward(a_1) = &\quad 10*\tfrac{5}{9} + 0*\tfrac{1}{3} - 10*\tfrac{1}{9} &&= 4.44\\
ExpectedReward(a_2) = &\quad 0*\tfrac{2}{9} + 2*\tfrac{5}{9} + 0*\tfrac{2}{9} &&= 1.11\\
ExpectedReward(a_3) = &\quad -10*\tfrac{1}{9} + 0*\tfrac{1}{3} + 10*\tfrac{5}{9} &&= 4.44
\end{aligned}$$

Note that the Penalty game has two global optima, and that the lenient agents might have difficulties choosing both the same optimum. As we will show in this paper, lenience can help multiagent learning algorithms converge to the global optimum in domains with a unique such global optimum, as well as in some domains with multiple optima.

The results above support the hypothesis that lenient learners might benefit from estimating the expected reward for an action based on how good that action might be once the teammate has learned a better behavior. However, the analysis so far relied on the assumption that the teammate only chooses random actions. The EGT frameworks for multiagent learning, as described in Section 2.3, clearly eliminate this assumption by explicitly keeping track of the agents' probabilities of selecting each action over time, and by using that information to describe the learning process of each agent. The paper continues with extensions of these models to account for lenience: Section 4 demonstrates the benefits of lenience for cooperative coevolutionary algorithms, while Section 5 presents evidence for the advantages of lenience in multiagent Q-learning algorithms.

4. Enhanced Evolutionary Game Theory Model for Cooperative Coevolutionary Algorithms

As described in Section 2.3.1, the EGT model for CCEAs updates the utility of an action based on the expected reward for that action in combination with all possible actions that the teammate might select (Equations 5 – 6). Next, we use Theorem 1 to enhance this model to account for lenient learners.

Assessing fitness as the maximum of multiple rewards has been used before in practical cooperative coevolutionary algorithms (for example, in Wiegand et al., 2001). The research here contributes a formal analysis of this approach, which has so far been lacking.

Definition 1 *The EGT model in Equations 18 – 21 represents a theoretical model for a cooperative coevolutionary algorithm where the fitness of an individual is computed as the maximum reward with N collaborators.*

$$u_i^{(t)} = \sum_{j=1}^{m} a_{ij} \frac{y_j^{(t)}}{\sum_{k:a_{ik}=a_{ij}} y_k^{(t)}} \left(\left(\sum_{k:a_{ik}\leq a_{ij}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{ik}<a_{ij}} y_k^{(t)} \right)^N \right), \tag{18}$$

$$w_j^{(t)} = \sum_{i=1}^{n} a_{ij} \frac{x_i^{(t)}}{\sum_{k:a_{kj}=a_{ij}} x_k^{(t)}} \left(\left(\sum_{k:a_{kj}\leq a_{ij}} x_k^{(t)} \right)^N - \left(\sum_{k:a_{kj}<a_{ij}} x_k^{(t)} \right)^N \right), \tag{19}$$

$$x_i^{(t+1)} = \left(\frac{u_i^{(t)}}{\sum_{k=1}^{n} x_k^{(t)} u_k^{(t)}} \right) x_i^{(t)}, \tag{20}$$

$$y_j^{(t+1)} = \left(\frac{w_j^{(t)}}{\sum_{k=1}^{m} y_k^{(t)} w_k^{(t)}} \right) y_j^{(t)}. \tag{21}$$

Note that the enhanced model is equivalent to Wiegand's standard EGT model in Equations 5 – 8 when a single collaborator is used ($N = 1$). As demonstrated next, CCEAs are expected to achieve better performance as N increases. The proof that is the case relies on the following lemmas which

establish bounds on the probabilities that the initial populations have extremely large or small proportions of certain genotypes.

Lemma 1 *Assume the populations for the EGT model are initialized at random based on a uniform distribution over all possible initial populations. Then, for any* $\varepsilon > 0$*, there exists* $\theta_\varepsilon > 0$ *such that*

$$P\left(\min_{i=1..n} x_i^{(0)} \leq \theta_\varepsilon\right) < \varepsilon, \tag{22}$$

$$P\left(\max_{i=1..n} x_i^{(0)} \geq 1-\theta_\varepsilon\right) < \varepsilon, \tag{23}$$

$$P\left(\min_{j=1..m} y_j^{(0)} \leq \theta_\varepsilon\right) < \varepsilon, \tag{24}$$

$$P\left(\max_{j=1..m} y_j^{(0)} \geq 1-\theta_\varepsilon\right) < \varepsilon. \tag{25}$$

Proof One method to sample the simplex Δ^n uniformly is described in Devroye (1986) (pages 568 – 569): take $n-1$ uniformly distributed numbers in $[0,1]$, then sort them, and finally use the differences between consecutive numbers (also, the difference between the smallest number and 0, and that between 1 and the largest number) as the coordinates for the point.

It follows that $\min_{i=1..n} x_i^{(0)}$ is the smallest distance between two such numbers (and possibly the boundaries 0 and 1). Similarly, $\max_{i=1..n} x_i^{(0)}$ is the largest distance between two numbers.

Suppose $\gamma > 0$ is a small number. We iterate over the $n-1$ uniformly-distributed random numbers that are needed to generate an initial population $\left(x_i^{(0)}\right)_{i=1..n}$. The probability that the first number is not within γ of the boundaries 0 and 1 is $1-2\gamma$. The probability that the second number is not within γ of the boundaries or of the first number is less than or equal to $1-4\gamma$. In general, the probability that the kth number is not within γ of the boundaries or of the first $k-1$ numbers is less than or equal to $1-2k\gamma$. Given that the numbers are generated independently of one another, the probability that the closest pair of points (considering the boundaries) is farther apart than γ is equal to

$$\begin{aligned} P\left(\min_{i=1..n} x_i^{(0)} \leq \gamma\right) &= 1 - P\left(\min_{i=1..n} x_i^{(0)} > \gamma\right) \\ &= 1 - \prod_{i=1}^{n-1} (1-2i\gamma) \quad \leq \quad 1-(1-2(n-1)\gamma)^{n-1}. \end{aligned}$$

Inequalities 22 and 24 are symmetric, and as such, the proof that

$$P\left(\min_{j=1..m} y_j^{(0)} \leq \gamma\right) \leq 1-(1-2(m-1)\gamma)^{m-1}$$

is very similar. Given that

$$\begin{aligned} \lim_{\gamma\to 0} 1-(1-2(n-1)\gamma)^{n-1} &= 0, \\ \lim_{\gamma\to 0} 1-(1-2(m-1)\gamma)^{m-1} &= 0 \end{aligned}$$

it follows that for any $\varepsilon > 0$ there exists $\theta_\varepsilon > 0$ such that $P\left(\min_{i=1..n} x_i^{(0)} \leq \theta_\varepsilon\right) < \varepsilon$ and $P\left(\min_{j=1..m} y_j^{(0)} \leq \theta_\varepsilon\right) < \varepsilon$.

To prove Inequality 23, consider that $\max_{i=1..n} x_i^{(0)} \geq 1-\theta_\varepsilon$ implies that all other x_i ratios except for the maximum are smaller to θ_ε, which, as proven above, occurs with probability smaller than ε. The proof for Inequality 25 is similar. ■

Lemma 1 basically proved that the probability that an agent starts with an extremely low or an extremely high probability of selecting an action is very low, given a random initialization of the multiagent learning process. Following, Lemma 2 derives a complementary result, namely that each agent should have a fair likelihood of selecting each of its actions when the learning process starts.

Lemma 2 *Assume the populations for the EGT model are initialized at random based on a uniform distribution over all possible initial populations. Then, for any* $\varepsilon > 0$*, there exists* $\eta_\varepsilon > 0$ *such that*

$$P\left(\min_{i=1..n} x_i^{(0)} > \eta_\varepsilon \wedge \max_{i=1..n} x_i^{(0)} < 1-\eta_\varepsilon \wedge \min_{j=1..m} y_j^{(0)} > \eta_\varepsilon \wedge \max_{j=1..m} y_j^{(0)} < 1-\eta_\varepsilon\right) \geq 1-\varepsilon.$$

Proof We apply Lemma 1 for $\frac{1-\sqrt{1-\varepsilon}}{2}$, which is greater than 0. The specific value of η_ε for this proof equals the value of $\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}$ from Lemma 1. It follows that:

$$\begin{aligned}
&P\left(\min_{i=1..n} x_i^{(0)} > \eta_\varepsilon \wedge \max_{i=1..n} x_i^{(0)} < 1-\eta_\varepsilon \wedge \min_{j=1..m} y_j^{(0)} > \eta_\varepsilon \wedge \max_{j=1..m} y_j^{(0)} < 1-\eta_\varepsilon\right) \\
&\quad = P\left(\min_{i=1..n} x_i^{(0)} > \theta_{\frac{1-\sqrt{1-\varepsilon}}{2}} \wedge \max_{i=1..n} x_i^{(0)} < 1-\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right) \times \\
&\qquad P\left(\min_{j=1..m} y_j^{(0)} > \theta_{\frac{1-\sqrt{1-\varepsilon}}{2}} \wedge \max_{j=1..m} y_j^{(0)} < 1-\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right) \\
&\quad = \left(1-P\left(\min_{i=1..n} x_i^{(0)} \leq \theta_{\frac{1-\sqrt{1-\varepsilon}}{2}} \vee \max_{i=1..n} x_i^{(0)} \geq 1-\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right)\right) \times \\
&\qquad \left(1-P\left(\min_{j=1..m} y_j^{(0)} \leq \theta_{\frac{1-\sqrt{1-\varepsilon}}{2}} \vee \max_{j=1..m} y_j^{(0)} \geq 1-\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right)\right) \\
&\quad \geq \left(1-\left(P\left(\min_{i=1..n} x_i^{(0)} \leq \theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right)+P\left(\max_{i=1..n} x_i^{(0)} \geq 1-\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right)\right)\right) \times \\
&\qquad \left(1-\left(P\left(\min_{j=1..m} y_j^{(0)} \leq \theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right)+P\left(\max_{j=1..m} y_j^{(0)} \geq 1-\theta_{\frac{1-\sqrt{1-\varepsilon}}{2}}\right)\right)\right) \\
&\quad \geq \left(1-2\frac{1-\sqrt{1-\varepsilon}}{2}\right) \times \left(1-2\frac{1-\sqrt{1-\varepsilon}}{2}\right) \\
&\quad = 1-\varepsilon.
\end{aligned}$$

■

Next, we prove that the revised EGT model described by Equations 18−21 converges to the global optimum with an arbitrarily-high probability.

Theorem 2 *Given a joint reward matrix A with a unique global optimum $a_{i^\star j^\star}$, for any $\varepsilon > 0$ there exists $N_\varepsilon \geq 1$ such that the theoretical CCEA model in Equations 18−21 converges to the global optimum with probability greater than $(1-\varepsilon)$ for any number of collaborators N such that $N \geq N_\varepsilon$.*

Proof

ε only serves as a guarantee for the worst case scenario for the proportions of individuals in the initial populations. From Lemma 2, it follows that there exists $\eta_\varepsilon > 0$ such that with probability at least $1-\varepsilon$, it is the case that $\eta_\varepsilon < x_i^{(0)} < 1-\eta_\varepsilon$ and $\eta_\varepsilon < y_j^{(0)} < 1-\eta_\varepsilon$ for $i, j \in \{1,2,3\}$. In other words, with probability ε, the initial populations will not have any proportion of individuals that cover more than $1-\eta_\varepsilon$, nor cover less than η_ε of the entire population.

We will prove that there exists $N_\varepsilon \geq 0$ such that the EGT model converges to the global optimum for any $N \geq N_\varepsilon$ and for all initial populations that satisfy $\eta_\varepsilon < x_{i^\star}^{(0)} < 1-\eta_\varepsilon$ and $\eta_\varepsilon < y_{j^\star}^{(0)} < 1-\eta_\varepsilon$. To this end, let α be the second highest element of the joint payoff matrix A ($\alpha < a_{i^\star j^\star}$). It follows that $u_i^{(t)} \leq \alpha$ for all $i \neq i^\star$, and similarly $w_j^{(t)} \leq \alpha$ for all $j \neq j^\star$. Given the symmetry, proving only the first (by refining Equation 18) is sufficient:

$$\begin{aligned} u_i^{(t)} &\leq \sum_{j=1}^{m} \alpha \frac{y_j^{(t)}}{\sum_{k:a_{ik}=a_{ij}} y_k^{(t)}} \left(\left(\sum_{k:a_{ik}\leq a_{ij}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{ik}<a_{ij}} y_k^{(t)} \right)^N \right) \\ &\leq \alpha \sum_{j=1}^{m} \left(\left(\sum_{k:a_{ik}\leq a_{ij}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{ik}<a_{ij}} y_k^{(t)} \right)^N \right) \leq \alpha. \end{aligned}$$

Next, we identify a lower bound for $u_{i^\star}^{(t)}$:

$$
\begin{aligned}
u_{i^\star}^{(t)} &= \sum_{j=1}^{m} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(\left(\sum_{k:a_{i^\star k}\le a_{i^\star j}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{i^\star k}< a_{i^\star j}} y_k^{(t)} \right)^N \right) \\
&= a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \\
&\quad \sum_{j\neq j^\star} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(\left(\sum_{k:a_{i^\star k}\le a_{i^\star j}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{i^\star k}< a_{i^\star j}} y_k^{(t)} \right)^N \right) \\
&= a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \\
&\quad \sum_{j\neq j^\star \wedge a_{i^\star j}\ge 0} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(\left(\sum_{k:a_{i^\star k}\le a_{i^\star j}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{i^\star k}< a_{i^\star j}} y_k^{(t)} \right)^N \right) + \\
&\quad \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(\left(\sum_{k:a_{i^\star k}\le a_{i^\star j}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{i^\star k}< a_{i^\star j}} y_k^{(t)} \right)^N \right) \\
&\ge a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \\
&\quad \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(\left(\sum_{k:a_{i^\star k}\le a_{i^\star j}} y_k^{(t)} \right)^N - \left(\sum_{k:a_{i^\star k}< a_{i^\star j}} y_k^{(t)} \right)^N \right) \\
&\ge a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(\sum_{k:a_{i^\star k}\le a_{i^\star j}} y_k^{(t)} \right)^N .
\end{aligned}
$$

Given that $\sum_{k=1}^{m} y_k^{(t)} = 1$, the previous inequality can be further refined:

$$
\begin{aligned}
u_{i^\star}^{(t)} &\ge a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \frac{y_j^{(t)}}{\sum_{k:a_{i^\star k}=a_{i^\star j}} y_k^{(t)}} \left(1 - y_{j^\star}^{(t)} \right)^N \\
&\ge a_{i^\star j^\star} \left(1 - \left(1 - y_{j^\star}^{(t)} \right)^N \right) + \left(1 - y_{j^\star}^{(t)} \right)^N \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \\
&= a_{i^\star j^\star} - \left(1 - y_{j^\star}^{(t)} \right)^N \left(a_{i^\star j^\star} - \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \right). \qquad (26)
\end{aligned}
$$

The inequalities $\eta_\varepsilon < y_{j^\star}^{(0)} < 1-\eta_\varepsilon$ hold for the initial populations, as inferred earlier from Lemma 2. It follows from Equation 26 that

$$
u_{i^\star}^{(0)} \ge a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j\neq j^\star \wedge a_{i^\star j}< 0} a_{i^\star j} \right). \qquad (27)
$$

However,

$$\lim_{N\to\infty} a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j\neq j^\star \wedge a_{i^\star j}<0} a_{i^\star j} \right) \quad = \quad a_{i^\star j^\star} \tag{28}$$

Given that $a_{i^\star j^\star} > \alpha$, Equation 28 implies that there exists $N_1 \geq 1$ such that

$$a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j\neq j^\star \wedge a_{i^\star j}<0} a_{i^\star j} \right) > \frac{a_{i^\star j^\star}}{2} + \frac{\alpha}{2} > \alpha \tag{29}$$

for all $N \geq N_1$. From Equations 26 and 29, it follows that

$$u_{i^\star}^{(0)} > \frac{a_{i^\star j^\star}}{2} + \frac{\alpha}{2} > \alpha \tag{30}$$

for all $N \geq N_1$. Observe that N_1 does not depend on the initial populations $x^{(0)}$ and $y^{(0)}$. Similarly, it is straightforward to prove that

$$w_{j^\star}^{(t)} \geq a_{i^\star j^\star} - \left(1 - x_{i^\star}^{(t)}\right)^N \left(a_{i^\star j^\star} - \sum_{i\neq i^\star \wedge a_{ij^\star}<0} a_{ij^\star} \right), \tag{31}$$

$$w_{j^\star}^{(0)} \geq a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{i\neq i^\star \wedge a_{ij^\star}<0} a_{ij^\star} \right) \tag{32}$$

and that there exists $N_2 \geq 1$ such that

$$a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{i\neq i^\star \wedge a_{ij^\star}<0} a_{ij^\star} \right) > \frac{a_{i^\star j^\star}}{2} + \frac{\alpha}{2} > \alpha$$

which implies that

$$w_{j^\star}^{(0)} > \frac{a_{i^\star j^\star}}{2} + \frac{\alpha}{2} > \alpha$$

for all $N \geq N_2$.

Let $N_\varepsilon = max(N_1, N_2)$, and let $N \geq N_\varepsilon$. Next, we show by induction by t (the number of iterations of the model, that is, the number of generations) that the following four inequalities hold (note that the last two imply a monotonic increase in the actions that comprise the global optimum):

$$u_{i^\star}^{(t)} \geq a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j\neq j^\star \wedge a_{i^\star j}<0} a_{i^\star j} \right), \tag{33}$$

$$w_{j^\star}^{(t)} \geq a_{i^\star j^\star} - (1-\eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{i\neq i^\star \wedge a_{ij^\star}<0} a_{ij^\star} \right),$$

$$x_{i^\star}^{(t+1)} \geq x_{i^\star}^{(t)}, \tag{34}$$

$$y_{j^\star}^{(t+1)} \geq y_{j^\star}^{(t)}. \tag{35}$$

At the first generation ($t = 0$), the first two inequalities hold (Equations 27 and 32). Combining these with the definition of N, it follows that $u_{i^\star}^{(0)} > u_i^{(0)}$ for all $i \neq i^\star$ (and similarly, $w_{j^\star}^{(0)} > w_j^{(0)}$ for all $j \neq j^\star$). As a consequence, Equation 20 implies that

$$\begin{aligned} x_{i^\star}^{(1)} &= \left(\frac{u_{i^\star}^{(0)}}{\sum_{k=1}^{n} x_k^{(0)} u_k^{(0)}} \right) x_{i^\star}^{(0)} \\ &> \left(\frac{u_{i^\star}^{(0)}}{\sum_{k=1}^{n} x_k^{(0)} u_{i^\star}^{(0)}} \right) x_{i^\star}^{(0)} \\ &= x_{i^\star}^{(0)}. \end{aligned}$$

and Equation 21 similarly implies that

$$\begin{aligned} y_{j^\star}^{(1)} &= \left(\frac{w_{j^\star}^{(0)}}{\sum_{k=1}^{m} y_k^{(0)} w_k^{(0)}} \right) y_{j^\star}^{(0)} \\ &> \left(\frac{w_{j^\star}^{(0)}}{\sum_{k=1}^{m} y_k^{(0)} w_{j^\star}^{(0)}} \right) y_{j^\star}^{(0)} \\ &= y_{j^\star}^{(0)}. \end{aligned}$$

To prove the inductive step, it follows from Equation 26 and from the inductive hypothesis that

$$\begin{aligned} u_{i^\star}^{(t+1)} &\geq a_{i^\star j^\star} - \left(1 - y_{j^\star}^{(t+1)}\right)^N \left(a_{i^\star j^\star} - \sum_{j \neq j^\star \wedge a_{i^\star j} < 0} a_{i^\star j} \right) \\ &\geq a_{i^\star j^\star} - \left(1 - y_{j^\star}^{(t)}\right)^N \left(a_{i^\star j^\star} - \sum_{j \neq j^\star \wedge a_{i^\star j} < 0} a_{i^\star j} \right) \\ &\cdots \\ &\geq a_{i^\star j^\star} - \left(1 - y_{j^\star}^{(0)}\right)^N \left(a_{i^\star j^\star} - \sum_{j \neq j^\star \wedge a_{i^\star j} < 0} a_{i^\star j} \right) \\ &\geq a_{i^\star j^\star} - (1 - \eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{j \neq j^\star \wedge a_{i^\star j} < 0} a_{i^\star j} \right). \end{aligned}$$

Given the definitions of N and α, this also implies that $u_{i^\star}^{(t+1)} > \alpha > u_i^{(t+1)}$ for all $i \neq i^\star$. As a consequence,

$$\begin{aligned} x_{i^\star}^{(t+1)} &= \left(\frac{u_{i^\star}^{(t)}}{\sum_{k=1}^{n} x_k^{(t)} u_k^{(t)}}\right) x_{i^\star}^{(t)} \\ &> \left(\frac{u_{i^\star}^{(t)}}{\sum_{k=1}^{n} x_k^{(t)} u_{i^\star}^{(t)}}\right) x_{i^\star}^{(t)} \\ &= x_{i^\star}^{(t)}. \end{aligned}$$

Similarly, Equation 31 and the inductive hypothesis imply that

$$\begin{aligned} w_{j^\star}^{(t+1)} &\geq a_{i^\star j^\star} - \left(1 - x_{i^\star}^{(t+1)}\right)^N \left(a_{i^\star j^\star} - \sum_{i \neq i^\star \wedge a_{ij^\star} < 0} a_{ij^\star}\right) \\ &\geq a_{i^\star j^\star} - \left(1 - x_{i^\star}^{(t)}\right)^N \left(a_{i^\star j^\star} - \sum_{i \neq i^\star \wedge a_{ij^\star} < 0} a_{ij^\star}\right) \\ &\cdots \\ &\geq a_{i^\star j^\star} - (1 - \eta_\varepsilon)^N \left(a_{i^\star j^\star} - \sum_{i \neq i^\star \wedge a_{ij^\star} < 0} a_{ij^\star}\right). \end{aligned}$$

Again, given the definition of N and α, this implies $w_{j^\star}^{(t+1)} > \alpha > w_j^{(t+1)}$ for all $j \neq j^\star$. As a consequence,

$$\begin{aligned} y_{j^\star}^{(t+1)} &= \left(\frac{w_{j^\star}^{(t)}}{\sum_{k=1}^{m} y_k^{(t)} w_k^{(t)}}\right) y_{j^\star}^{(t)} \\ &> \left(\frac{w_{j^\star}^{(t)}}{\sum_{k=1}^{m} y_k^{(t)} w_{j^\star}^{(t)}}\right) y_{j^\star}^{(t)} \\ &= y_{j^\star}^{(t)} \end{aligned}$$

Having shown that both $\left(x_{i^\star}^{(t)}\right)_t$ and $\left(y_{j^\star}^{(t)}\right)_t$ are monotonically increasing (Equations 34–35), and given that they are bounded between 0 and 1, it follows that they converge to some value. Let $\bar{x} = \lim_{t\to\infty} x_{i^\star}^{(t)}$. We used Lemma 2 to define $\eta_\varepsilon > 0$ such that $\eta_\varepsilon < x_i^{(0)}$ (among other inequalities). This clearly implies $\bar{x} > 0$, as $\left(x_{i^\star}^{(t)}\right)_t$ is monotonically increasing. We also have that

$$\begin{aligned} \frac{x_{i^\star}^{(t+1)}}{x_{i^\star}^{(t)}} &= \frac{u_{i^\star}^{(t)}}{\sum_{k=1}^{n} u_k^{(t)} x_k^{(t)}} \\ &\geq \frac{u_{i^\star}^{(t)}}{u_{i^\star}^{(t)} x_{i^\star}^{(t)} + \alpha\left(1 - x_{i^\star}^{(t)}\right)} \\ &= 1 + \frac{\left(u_{i^\star}^{(t)} - \alpha\right)\left(1 - x_{i^\star}^{(t)}\right)}{u_{i^\star}^{(t)} x_{i^\star}^{(t)} + \alpha\left(1 - x_{i^\star}^{(t)}\right)}. \end{aligned}$$

But $\lim_{t\to\infty} \frac{x_{i^\star}^{(t+1)}}{x_{i^\star}^{(t)}} = 1$, and therefore

$$\lim_{t\to\infty} \frac{\left(u_{i^\star}^{(t)} - \alpha\right)\left(1 - x_{i^\star}^{(t)}\right)}{u_{i^\star}^{(t)} x_{i^\star}^{(t)} + \alpha\left(1 - x_{i^\star}^{(t)}\right)} = 0$$

which implies

$$\lim_{t\to\infty} \left(u_{i^\star}^{(t)} - \alpha\right)\left(1 - x_{i^\star}^{(t)}\right) = 0.$$

Finally, Equations 30 and 33 imply that $u_{i^\star}^{(t)} - \alpha > \frac{a_{i^\star j^\star}}{2} - \frac{\alpha}{2} > 0$. As a consequence $\lim_{t\to\infty} x_{i^\star}^{(t)} = 1$. The proof that $y_{j^\star}^{(t)}$ also converges to 1 is similar and is omitted for brevity. It follows that $x_i^{(t)}$ and $y_j^{(t)}$ converge to 0 for all $i \neq i^\star$ and for all $j \neq j^\star$. ■

Theorem 2 shows that CCEAs will converge to the global optimum in domains with a unique such optimum, if set appropriately and if given enough resources. This proves that the earlier claims of uncontrolled drift to suboptimal solutions, as reported in Wiegand (2004), were entirely due to the use of a formal model that employed a simplified fitness assessment mechanism. The term "enough" is used here to indicate two types of resources that CCEAs need: large populations to explore the space (the populations are in fact assumed to be infinite in our formal model, for simplicity), and large numbers of collaborators to provide accurate fitness estimates for all populations. This paper does not go into great depths with respect to the requirement for large populations. As for the second requirement, Section 4.1 analyzes the impact of the number of collaborators onto the performance of the CCEA.

Observe also that Theorem 2 does not cover domains with multiple optima, such as the Penalty domain discussed in Section 3. Next, we use a visualization technique to study the impact of lenience onto cooperative coevolutionary algorithms as applied to domains with one optimum, as well as to domains with multiple global optima.

4.1 Basins of Attraction due to Estimations

This section describes an intuitive way to visualize the impact of improved estimates on the performance of the system. Given specific initial populations, the EGT model is completely deterministic. As shown by Wiegand (2004), the populations are expected to converge to Nash equilibria in the payoff matrix. As a result, it is straightforward to provide two-dimensional visualizations of the basins of attraction for different Nash equilibria when each population contains only two genotypes: the two-dimensional location of a pixel uniquely encodes the initial ratios of one of the genotypes for each population, and the color of the pixel specifies the equilibrium to which the system converges after repeated iteration (see Panait et al., 2004, for details).

However, such simple problem domains present a limited range of challenges to cooperative coevolutionary algorithms. This section will instead illustrate the basins of attraction of Nash equilibria in two 3×3 coordination games: Climb and Penalty. This domains can stress the challenges posed by poor estimates of fitness, as discussed in Section 3.

Given that fitness proportional selection works properly only when all individuals have positive fitness, we add 30 to all values in the Climb payoff matrix, and we add 10 to all values in the Penalty

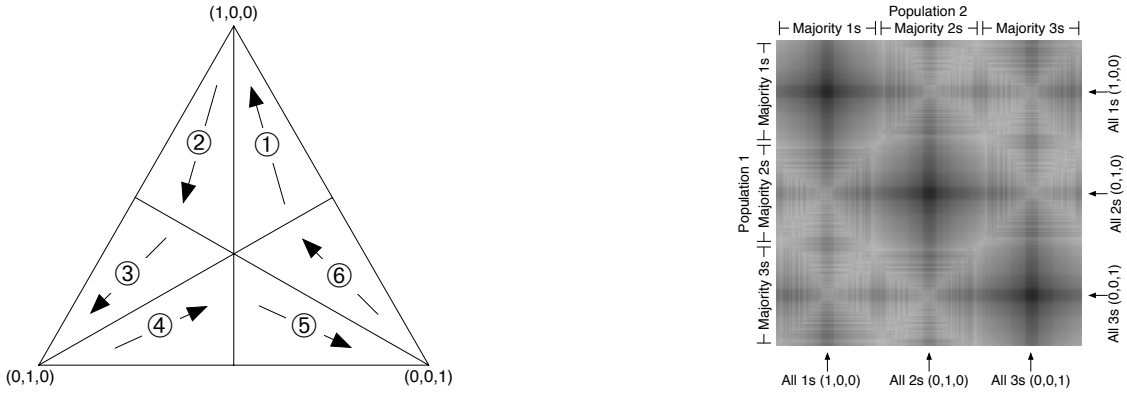


Figure 2: The projection of $\Delta^3 \times \Delta^3$ to $[0,1]^2$. (a): The projection divides the simplex Δ^3 into six equal-area triangles; arrows show the direction for sorting points in each area. (b): Visualization of the cartesian product of two simplexes.

payoff matrix. In contrast, in the next Section (Section 5), all experiments will employ the original payoff matrices.

We apply the EGT model in Equations $18-21$ to each of these two problem domains. The model is iterated for 100000 generations or until the proportion of one action in each population exceeds a threshold of $1-10^{-10}$. A specific color identifies each of the possible end results (Nash equilibria). For consistency, black dots always indicate convergence to a suboptimal solution, while white and grey dots indicate convergence to global optima. The projection of the $\Delta^3 \times \Delta^3$ is detailed next.

The search space (Δ^3) of the first population is projected along the vertical axis, while that of the second population is projected along the horizontal axis. The objective of this process is to group together regions of the space of initial populations that are expected to converge to the same Nash equilibrium. The projection of Δ^3 to one dimension starts by dividing it into six equal-area triangles, as shown in Figure 2a. Initial populations in areas $1-2$ have a majority of 1s in the population, and similarly areas $3-4$ and $5-6$ have majorities of 2s and 3s. If $i < j$, all initial populations in area i are projected before those in area j. Inside each area, initial populations are ordered lexicographically in the direction of the arrow. More specifically, in regions $1-2$, the sorting is done primarily on p_1, and secondarily on p_2; for $3-4$, p_2 and p_3; for $5-6$, p_3 and p_1. Even-numbered regions are sorted ascending and odd-numbered regions are sorted descending.

We sample 216 initial populations in the simplex: the six areas in Figure 2a are each divided into six triangles, and each of them is further divided into six more triangles. The center of each resulting triangle corresponds to an initial population. We add random noise distributed uniformly between -0.00005 and 0.00005 to reduce certain artifacts due to identical distributions of the two populations (see the discussion at the end of this section about the thin diagonal line in Figure 4). The sampling also does not cover initial populations on the edges or vertexes of the simplex, but the probability that an evolutionary algorithm starts from those initial populations is negligibly small.

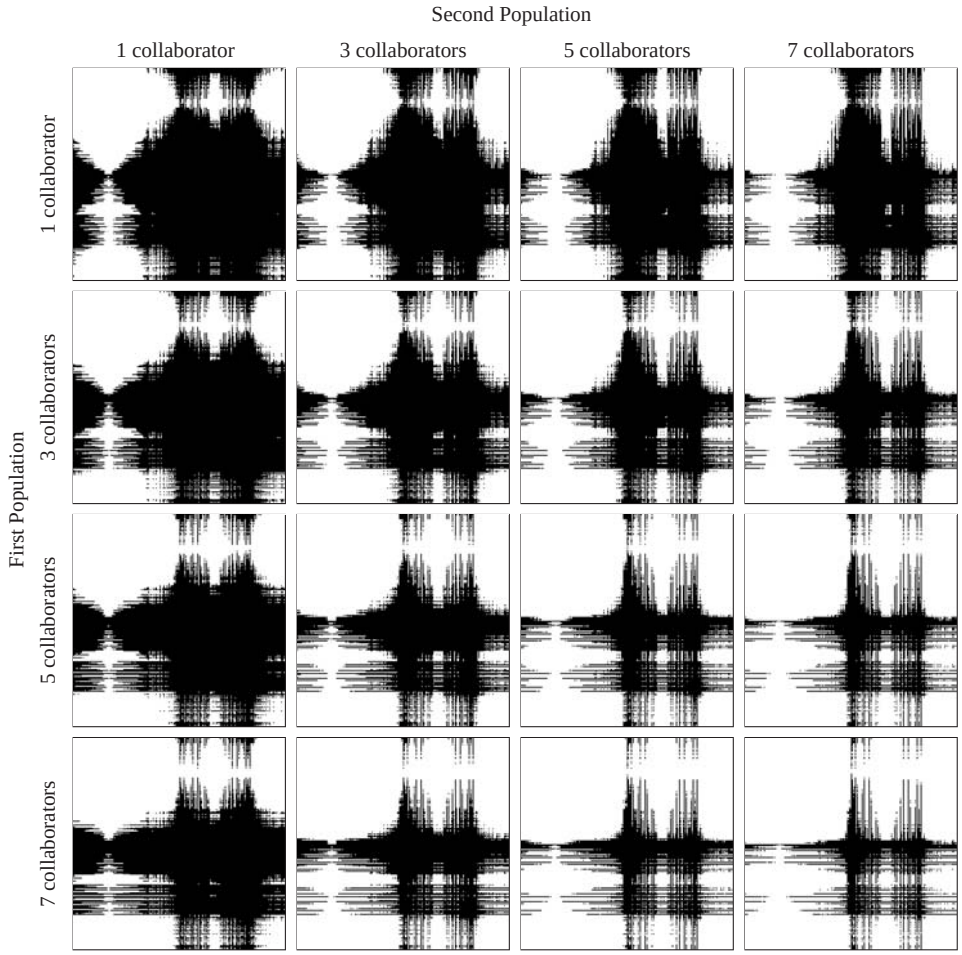


Figure 3: Basins of attraction for CCEA in the Climb problem domain for cooperative coevolution with 1, 3, 5 and 7 collaborators per population. White and black mark the basins of attraction for the (1,1) and (2,2) equilibria, respectively.

The right image in Figure 2 is an example of the resulting projection of $(\Delta^3)^2$ onto 2-D. Thanks to the sorting described above, certain regions reflect majority-1, majority-2, and majority-3 regions; and borders between those regions are the mixture of the two regions. Dark lines in this figure show locations that have high ratios of 1s, 2s, or 3s in one or the other population.

First, Figure 3 visualizes the impact of improved estimates due to increased numbers of collaborators for each population (to parallel Theorem 2). The images shows the basins of attraction in the Climb coordination game. Observe that the difficulty of the problem domain decreases as each population is provided with more accurate estimates for the fitness of individuals. When using a single collaborator, it appears that the coevolutionary search will find the optimum if at least one of the populations starts with a large number of 1s. Even in this case, the system is most likely to converge to the global optimum if the ratio of 2s is relatively low. As each population gets a

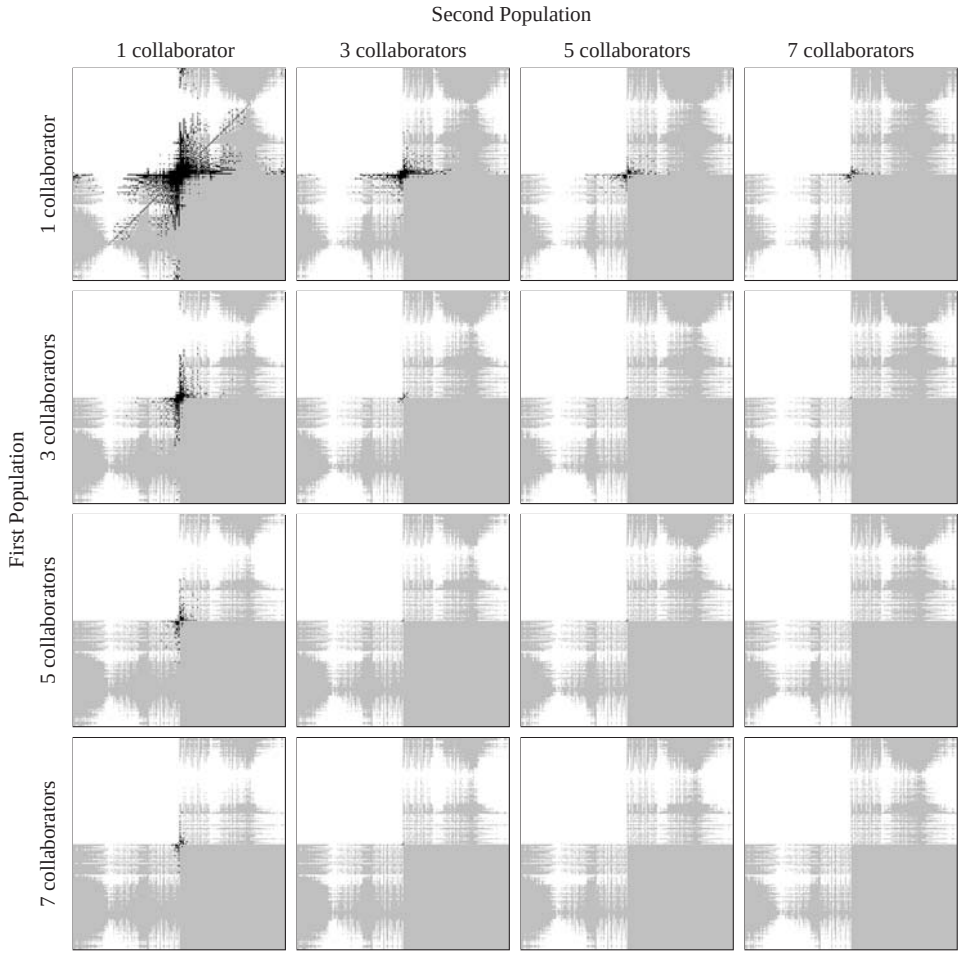


Figure 4: Basins of attraction for CCEA in the Penalty problem domain for cooperative coevolution with 1, 3, 5 and 7 collaborators per population. White, black, and light grey mark the basins of attraction for the (1,1), (2,2), and (3,3) equilibria, respectively.

better estimate of fitness (via an increased number of collaborators), the basin of attraction for the suboptimal equilibrium reduces to areas where at least one of the initial populations has a very large proportion of 2s or 3s: the more collaborators are used, the larger the proportion of 2s or 3s required to still converge to the sub-optimum.

Figure 4 presents the basins of attraction in the Penalty game. Note that the Penalty game has two global optima and thus does not fit the hypothesis of the formal proof in Theorem 2. The basins of attraction illustrated in Figure 4 indicate that lenient learners exhibit good performance in some games with multiple global optima as well. What precise characteristics of the coordination game pose difficulties to lenient learners is the focus of ongoing analysis.

Observe that the two global optima cover most of the space in Figure 4 even when a single collaborator is used; the suboptimal equilibria covers mainly areas where at least one of the pop-

ulation started with a high percentage of 2s, and the other population has the 1s and 3s equally distributed—this increases the percentage of miscoordinations. As the number of collaborators is increased, the basin of attraction for the (2,2) point reduces to only areas where *both* populations start with almost solely 2s. The visualization of the basins of attraction suggests that Penalty is a much easier coordination game than Climb. Note also the thin diagonal line in the top-left graph of Figure 4. Interestingly, this is due to the fact that if the proportion of 1s in one population is about equal to the proportion of 3s in the other population, there are frequent miscoordinations that impact on the expected reward for these actions as estimated by the learners, and the system converges to the suboptimal (2, 2) equilibrium.

The visualization of the basins of attraction to Nash equilibria provided an intuitive approach to grasping the impact that lenience has onto cooperative coevolutionary algorithms. Figures 3 – 4 showed that the number of trajectories for CCEAs that converge to suboptimal solutions decreases significantly across both problem domains, as the level of lenience increases. Next, we apply this visualization technique to show that lenience is not specific to only cooperative coevolution, but it can be used to improve the performance of other multiagent learning algorithms as well.

5. Evolutionary Game Theory Models for Lenient Multiagent Q-Learning

The RD model in Equations 14–17 assumes that the agent will update the utility of an action based on the average reward it expects to receive for that action. This approach is therefore similar to the one used by the EGT model of cooperative coevolutionary algorithms in Equations 5–8. As argued in Section 3 and demonstrated in Section 4 (for cooperative coevolutionary algorithms only), using the average reward usually results in poor estimates for the quality of actions, causing potential attraction to towards suboptimal solutions.

To remedy this situation, we extend the RD model such that each learner ignores lower rewards to improve its estimate. The key to this extension is in Theorem 1 in Section 3. The following RD model is a straightforward combination of these previous results:

$$u_i = \sum_{j=1}^{m} \frac{a_{ij} y_j \left(\left(\sum_{k:a_{ik} \leq a_{ij}} y_k \right)^N - \left(\sum_{k:a_{ik} < a_{ij}} y_k \right)^N \right)}{\sum_{k:a_{ik}=a_{ij}} y_k}, \tag{36}$$

$$w_j = \sum_{i=1}^{n} \frac{a_{ij} x_i \left(\left(\sum_{k:a_{kj} \leq a_{ij}} x_k \right)^N - \left(\sum_{k:a_{kj} < a_{ij}} x_k \right)^N \right)}{\sum_{k:a_{kj}=a_{ij}} x_k}, \tag{37}$$

$$\frac{\frac{dx_i}{dt}}{x_i} = \frac{\alpha}{\tau} \left(u_i - \sum_k x_k u_k \right) + \alpha \sum_k x_k \ln \frac{x_k}{x_i}, \tag{38}$$

$$\frac{\frac{dy_j}{dt}}{y_j} = \frac{\alpha}{\tau} \left(w_j - \sum_k y_k w_k \right) + \alpha \sum_k y_k \ln \frac{y_k}{y_j}. \tag{39}$$

Note that the two equations describing the update rule for the two learners have not changed. What has changed, however, is the expected reward that is used to update the utilities of actions Equations 36 and 37). As expected, observe that the two RD models described by Equations 14–17 and 36–39 are equivalent when $N = 1$ (that is, when the agents do not ignore any low rewards). For

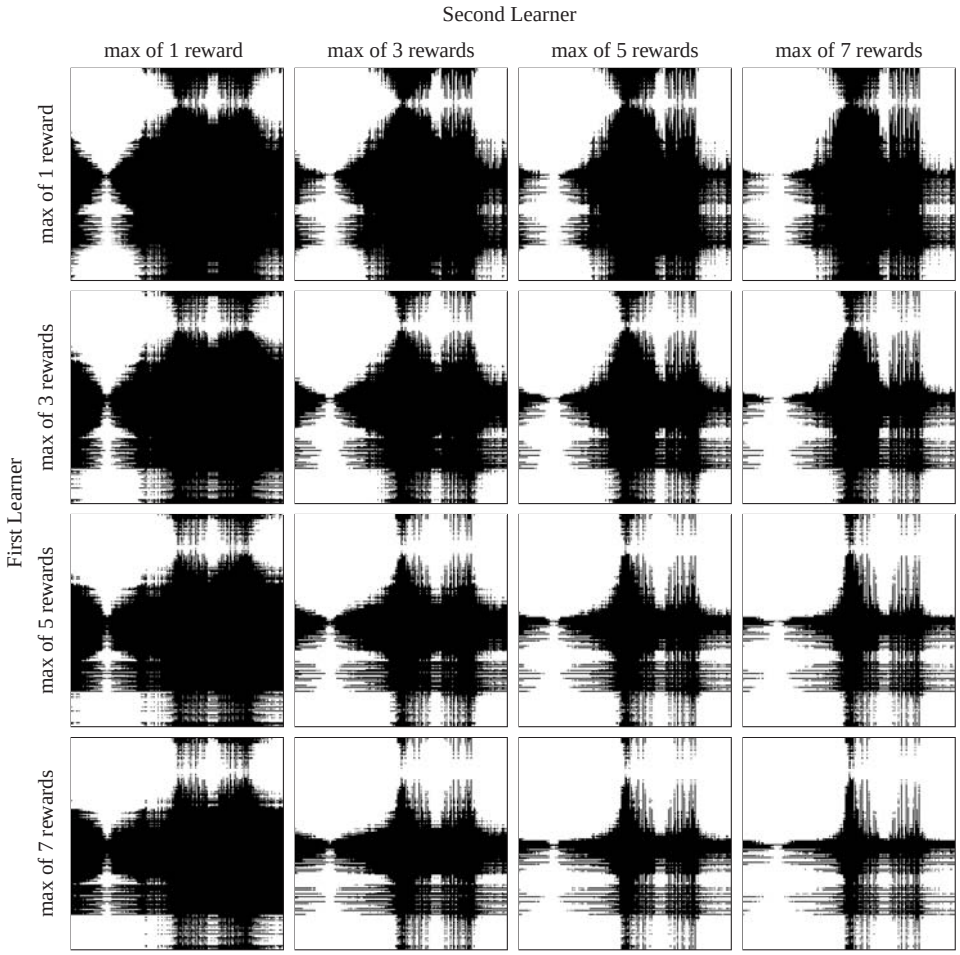


Figure 5: Basins of attraction in the Climb problem domain for multiagent Q-learning that updates the utility of an action based on the maximum of 1, 3, 5 and 7 of the rewards received when selecting that action. White and black mark the basins of attraction for the (1,1) and (2,2) equilibria.

this reason, the setting $N = 1$ constitutes a benchmark representing the formal model for traditional Q-learners.

5.1 Basins of Attraction to Nash Equilibria for Multiagent Q-Learning

Given that the RD model provides differential equations, the next state of the concurrent learning system can be approximated by assuming the variations in the derivative are small over short periods of time θ:

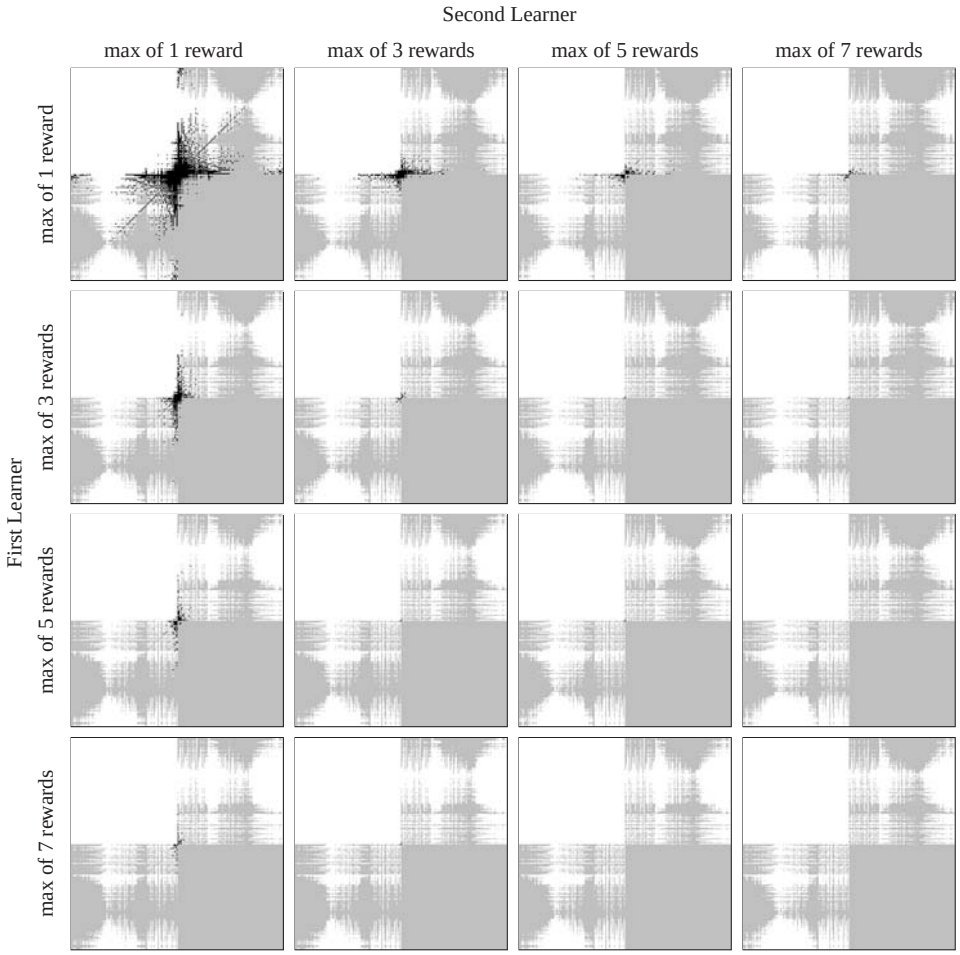


Figure 6: Basins of attraction in the Penalty problem domain for multiagent Q-learning that updates the utility of an action based on the maximum of 1, 3, 5 and 7 of the rewards received when selecting that action. White, black, and light grey mark the basins of attraction for the (1,1), (2,2), and (3,3) equilibria, respectively.

$$\begin{aligned} x_i' &= x_i + \theta \frac{dx_i}{dt}, \\ y_j' &= y_j + \theta \frac{dy_j}{dt}. \end{aligned}$$

Experiments used the following settings: $\theta = 0.001$, the learning rate $\alpha = 0.1$, and the exploration parameter $\tau = 0.01$. We iterate the RD model 100000 times, or until both agents have a probability exceeding $1 - 10^{-10}$ to select one of their actions over the other two.

The basins of attraction for the optimal $(1,1)$ (both play their first action) and suboptimal $(2,2)$ (both play their second action) equilibria in the Climb domain are visualized in Figure 5. Given

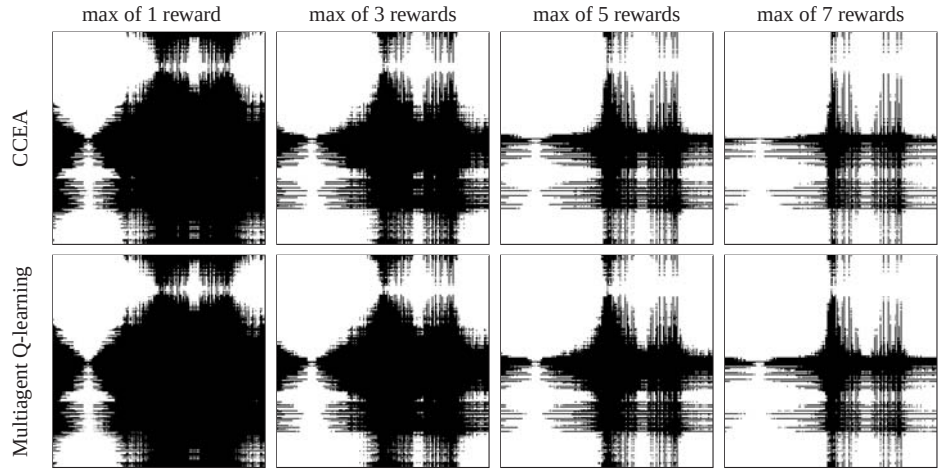


Figure 7: Basins of attraction in the Climb problem domain for cooperative coevolution and multiagent reinforcement learning, both ignoring all but the maximum of 1, 3, 5, and 7 rewards. White and black mark the basins of attraction for the (1,1) and (2,2) equilibria.

that the new RD model reduces to the one described in Tuyls et al. (2006) when $N = 1$, the basin of attraction in the top-left graph suggests that a straightforward extension of Q-learning to multiagent domains may converge to the suboptimal solution when started from many initial conditions. As the learners ignore more of the lower rewards, they improve their estimates for the quality of their actions, and are thus more likely to converge to the global optimum. The same behavior can be observed in the Penalty domain, as illustrated in Figure 6.

Note that multiagent Q-learning improves with more lenience in nearly an identical (but not *quite* identical) fashion to coevolution. Figure 5 shows the basins of attraction in the Climb coordination game. Once again, the images show that the difficulty of the problem domain decreases as each population is provided with more accurate estimates for the utilities of actions. Similarly, Figure 6 presents the basins of attraction in the Penalty game. As was the case for coevolution, increasing numbers of collaborators are able to essentially eliminate convergence to the suboptimal equilibrium. Also, Q-learning likewise has a thin diagonal line of suboptimal convergence in the top-left graph of Figure 6, which once again is due to one agent selecting action 1 in equal proportions to the second agent selecting action 3, and vice-versa.

6. Discussion and Conclusions

At first glance one would think that coevolution and multiagent Q-learning, being fairly different algorithms, with fairly different settings and aimed at different kinds of problems, would respond differently in these scenarios. However, not only do they perform nearly identically in the "classic" case (no lenience), but they also respond in nearly the same fashion when subjected to varying degrees of lenience as well. To highlight this similarity, Figures 7 and 8 sample the diagonal graphs in Figures 3 – 5 and Figures 4 – 6: these are the cases where both agents ignore the same number of lower rewards. This is not really that surprising however: as the theoretical analysis has shown,

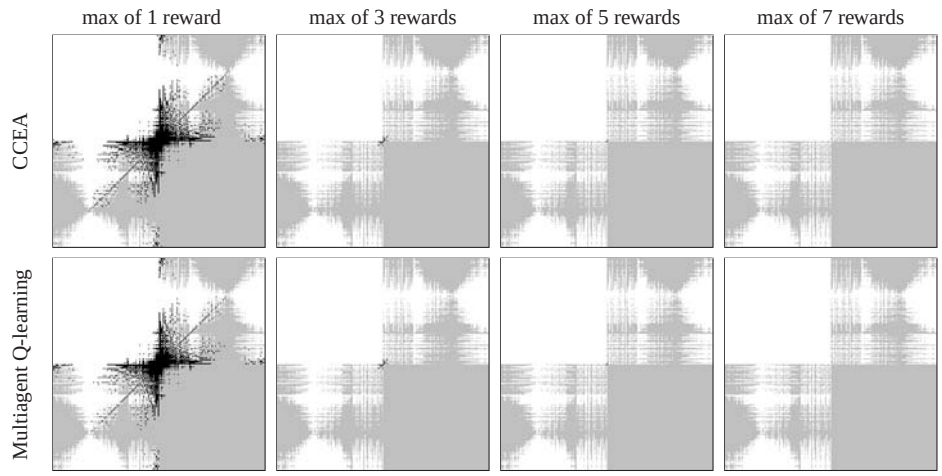


Figure 8: Basins of attraction in the Penalty problem domain for cooperative coevolution and multiagent reinforcement learning, both ignoring all but the maximum of 1, 3, 5, and 7 rewards. White, black, and light grey mark the basins of attraction for the (1,1), (2,2), and (3,3) equilibria, respectively.

the dynamics of these equations is quite similar; so much so, indeed, that each of these respective communities would do well to more closely examine the existing work of the other.

Keep in mind that both evolutionary algorithms and Q-learning are guaranteed to converge to the global optimum in single-agent scenarios, if properly set and if given enough resources. We have argued that the attraction of these algorithms towards suboptimal solutions is caused primarily by poor estimates, and that it can be dealt with if learners simply ignore lower rewards. Figures 7 and 8 suggest that if the multiagent learning algorithm is heading towards suboptimal solutions, it is usually caused by the estimation procedure for the quality of an action of the learning algorithm. These algorithms do not start to suffer from undesirable pathologies just because they are used in multiagent scenarios. Rather, poor estimates are the primary cause for the suboptimal solutions. Ignoring lower rewards improves the learners' estimate for the quality of their behaviors, resulting in an increased probability that the multiagent learning system will converge to the global optimum.

In summary, this paper presented an extended formal model for a new class of multiagent learning algorithms, namely those involving lenient agents that ignore low rewards observed upon performing actions in the environment. The paper also illustrated the basins of attraction to different optimal and suboptimal Nash equilibria, and showed how intuitive graphs might reveal valuable information about the properties of multiagent learning algorithms that was lacking in the summarizations of empirical results. The paper provided theoretical support for previous reports that lenience helps learners achieve higher rates of convergence to optimal solutions, and also proved that properly-set lenient learners are guaranteed to converge to the Pareto-optimal Nash equilibria in coordination games. Finally, the results strengthened the use of Evolutionary Game Theory to study the properties of multiagent reinforcement learning algorithms.

This work has also opened many avenues of future research. We plan to extend these formal models to help analyze the properties of multiagent learning algorithms in more complex domains,

such as those involving more actions and players, stochasticity, multiple states, and partial observability. We hope that this formal analysis will provide the foundation for new learning algorithms guaranteed to converge to optimal solutions.

Acknowledgments

The authors would like to thank R. Paul Wiegand, Dana Richards, Keith Sullivan, Dominic Mazzoni, and the anonymous reviewers for helpful discussions and suggestions.

References

Adrian K. Agogino and Kagan Tumer. Evolving distributed agents for managing air traffic. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1888 – 1895, 2007.

Adrian K. Agogino and Kagan Tumer. Handling communication restrictions and team formation in congestion games. *Journal of Autonomous Agents and Multi Agent Systems*, 13(1):97–115, 2006.

Stephane Airiau, Kagan Tumer, and Adrian K. Agogino. Learning agents for distributed and robust spacecraft power management. In *AAMAS-06 Workshop on Adaptation and Learning in Autonomous Agents and Multi Agent Systems*, 2006.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multi-agent systems. In *Proceedings of National Conference on Artificial Intelligence (AAAI-98)*, pages 746–752, 1998.

Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

Roger Eriksson and Björn Olsson. Cooperative coevolution in inventory control optimisation. In G. Smith, N. Steele, and R. Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference, ICANNGA97*, 1997.

Nancy Fulda and Dan Ventura. Predicting and preventing coordination problems in cooperative learning systems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007.

Herbert Gintis. *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction*. Princeton University Press, 2001.

Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. Efficient non-linear control through neuroevolution. In *Lecture Notes in Computer Science (LNCS 4212)*, pages 654–662. 2006.

Josef Hofbauer and Karl Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998.

Phil Husbands and Frank Mill. Simulated coevolution as the mechanism for emergent planning and scheduling. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 264–270, 1991.

Thomas Jansen and R. Paul Wiegand. Exploring the explorative advantage of the CC (1+1) EA. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 310–321, 2003.

Thomas Jansen and R. Paul Wiegand. The cooperative coevolutionary (1+1) ea. *Evolutionary Computation*, 12(4):405–434, 2004.

Kenneth De Jong. *Evolutionary Computation: A Unified Approach*. MIT Press, 2006.

Spiros Kapetanakis and Daniel Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-02)*, 2002.

Matt Knudson and Kagan Tumer. Effective policies for resource limited agents. In *Proceedings of the AAMAS-07 Workshop on Adaptive and Learning Agents Workshop (ALAg-07)*, 2007.

Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 535–542, 2000.

John Maynard Smith and George R. Price. The logic of animal conflict. *Nature*, 146:15–18, 1973.

Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005a.

Liviu Panait and Sean Luke. Time-dependent collaboration schemes for cooperative coevolutionary algorithms. In *Proceedings of the 2005 AAAI Fall Symposium on Coevolutionary and Coadaptive Systems*, 2005b.

Liviu Panait and Sean Luke. Selecting informative actions improves cooperative multiagent learning. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems – AAMAS-2006*, 2006.

Liviu Panait, R. Paul Wiegand, and Sean Luke. Improving coevolutionary search for optimal multiagent behaviors. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.

Liviu Panait, R. Paul Wiegand, and Sean Luke. A visual demonstration of convergence properties of cooperative coevolution. In *Parallel Problem Solving from Nature – PPSN-2004*, pages 892–901, 2004.

Liviu Panait, Keith Sullivan, and Sean Luke. Lenience towards teammates helps in cooperative multiagent learning. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems – AAMAS-2006*, 2006.

Mitchell Potter. *The Design and Analysis of a Computational Model of Cooperative CoEvolution*. PhD thesis, George Mason University, Fairfax, Virginia, 1997.

Mitchell Potter and Kenneth De Jong. A cooperative coevolutionary approach to function optimization. In *Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN III)*, pages 249–257, 1994.

Mitchell Potter and Kenneth De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

Satinder P. Singh, Michael J. Kearns, and Yishay Mansour. Nash convergence of gradient dynamics in general-sum games. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 541–548, 2000.

John Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

Kagan Tumer and Adrian K. Agogino. Coordinating multi-rover systems: Evaluation functions for dynamic and noisy environments. In *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 371–388. Springer, 2007.

Karl Tuyls, Katja Verbeeck, and Tom Lenaerts. A selection-mutation model for q-learning in multi-agent systems. In *The Second International Joint Conference on Autonomous Agents and Multi-Agent Systems. ACM Press, Melbourne, Australia*, 2003.

Karl Tuyls, Pieter Jan 't Hoen, and Bram Vanschoenwinkel. An evolutionary dynamical analysis of multi-agent learning in iterated games. *The Journal of Autonomous Agents and Multi-Agent Systems*, 12:115–153, 2006.

Michael Vose. *The Simple Genetic Algorithm*. MIT Press, 1999.

Christopher J. Watkins. *Models of Delayed Reinforcement Learning*. PhD thesis, Psychology Department, Cambridge University, Cambridge, United Kingdom, 1989.

Christopher J. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

Jorgen W. Weibull. *Evolutionary Game Theory*. MIT Press, 1996.

R. Paul Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University, Fairfax, Virginia, 2004.

R. Paul Wiegand and Mitchell Potter. Robustness in cooperative coevolution. In *GECCO '06: Proceedings of the 8th annual Conference on Genetic and Evolutionary Computation*, pages 369–376, 2006.

R. Paul Wiegand, William Liles, and Kenneth De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1235–1242, 2001.

R. Paul Wiegand, William Liles, and Kenneth De Jong. Modeling variation in cooperative coevolution using evolutionary game theory. In *Foundations of Genetic Algorithms VII*, pages 231–248, 2002.

Journal of Machine Learning Research 9 (2008) 459-483 Submitted 9/07; Revised 1/08; Published 3/08

A Recursive Method for Structural Learning of Directed Acyclic Graphs

Xianchao Xie XXIE@FAS.HARVARD.EDU
Zhi Geng ZGENG@MATH.PKU.EDU.CN
School of Mathematical Sciences, LMAM
Peking University
Beijing 100871, China

Editor: Marina Meila

Abstract

In this paper, we propose a recursive method for structural learning of directed acyclic graphs (DAGs), in which a problem of structural learning for a large DAG is first decomposed into two problems of structural learning for two small vertex subsets, each of which is then decomposed recursively into two problems of smaller subsets until none subset can be decomposed further. In our approach, search for separators of a pair of variables in a large DAG is localized to small subsets, and thus the approach can improve the efficiency of searches and the power of statistical tests for structural learning. We show how the recent advances in the learning of undirected graphical models can be employed to facilitate the decomposition. Simulations are given to demonstrate the performance of the proposed method.

Keywords: Bayesian network, conditional independence, decomposition, directed acyclic graph, structural learning

1. Introduction

Directed acyclic graphs (DAGs), also known as Bayesian networks, are frequently used to represent independencies, conditional independencies and causal relationships in a complex system with a large number of random variables (Lauritzen, 1996; Cowell et al., 1999; Pearl, 2000; Spirtes et al., 2000). Structural learning of DAGs from data is very important in applications to various fields, such as medicine, artificial intelligence and bioinformatics (Jordan, 2004; Engelhardt et al., 2006).

There have been two primary methods for learning the structures of DAGs from data. The search-and-score method defines a score for each possible structure based on the goodness-of-fit of the structure to data and the complexity of the structure, and then it tries to search the best structure over all possible structures (Cooper and Herskovits, 1992; Heckerman et al., 1995; Chickering, 2002; Friedman and Koller, 2003). The constraint-based method evaluates the presence or absence of an edge by testing conditional independencies among variables from data. The tests are usually done by using statistical or information-theoretic measures (Pearl, 2000; Spirtes et al., 2000; Cheng et al., 2002). There have also been hybrid methods. For example, Tsamardinos et al. (2006) takes advantage of both approaches. In a constraint-based method, search for separators of vertex pairs is a key issue for orientation of edges and for recovering DAG structures and causal relationships among variables. To recover structures of DAGs, Verma and Pearl (1990) presented the inductive causation (IC) algorithm which searches for a separator S of two variables, say u and v, from all possible

variable subsets such that u and v are independent conditionally on S. A systematic way of searching for separators in increasing order of cardinality was proposed by Spirtes and Glymour (1991). The PC algorithm limits possible separators to vertices that are adjacent to u and v (Pearl, 2000; Spirtes et al., 2000). Kalisch and Bühlmann (2007) showed that the PC algorithm is asymptotically consistent even when the number of vertices in a DAG grows at a certain rate as the sample size increases.

In this paper, we propose a recursive algorithm in which a problem of structural learning for a large DAG is split recursively into problems of structural learning for small vertex subsets. Our algorithm can be depicted as a binary tree whose top node is the full set of all vertices or variables and whose other nodes are proper subsets of the vertex set at its parent node. The algorithm mainly consists of two steps: the top-down step and the bottom-up step. First at the top-down step, the full set of all variables at the top is decomposed into two small subsets, each of which is decomposed recursively into two smaller subsets until each node cannot be decomposed further at the bottom of the tree. At each step, the decomposition is achieved by learning an undirected graph known as independence graph for a variable subset. Next at the bottom-up step, subgraphs (called skeletons) of leaf nodes are first constructed, and then a pair of child subgraphs are combined together into a large subgraph at their parent node until the entire graph is constructed at the top of the tree. In the algorithm, search for separators in a large graph is localized to small subgraphs. Statistical test is used to determine a skeleton as in the IC algorithm (Verma and Pearl, 1990) and the PC algorithm (Spirtes, 2000). By recursively decomposing the full variable set into small subsets, this algorithm can improve the efficiency of search for separators in structural learning, and it can also make statistical tests more powerful. We also discuss that several methods of learning undirected graphical models (Castelo and Roverato, 2006; Schmidt et al., 2007) can be used to facilitate the decomposition. Finally, we provide simulation results to show the performance of our method.

Section 2 gives notation and definitions. In Section 3, we first present the main theoretical results and then discuss the realization of the algorithm in detail, and we also introduce how the recent advances in various related fields can be used to improve the proposed method. In Section 4, we first use an example to illustrate our approach for learning the equivalence class of a DAG in detail, then we give numerical evaluations of its performance for several networks, and finally we discuss the computational complexity of our recursive algorithm. Conclusion is given in Section 5. The proofs of our main results are presented in Appendix.

2. Notation and Definitions

Let $\vec{G}_V = (V, \vec{E}_V)$ denote a DAG where $V = \{X_1, \ldots, X_n\}$ is the vertex set and $\vec{E}_V$ the set of directed edges. A directed edge from a vertex u to a vertex v is denoted by $\langle u, v \rangle$. We assume that there is no directed loop in $\vec{G}_V$. We say that u is a parent of v and v is a child of u if there is a directed edge $\langle u, v \rangle$, and denote the set of all parents of a vertex v by $pa(v)$ and the set of all children of v by $ch(v)$. We say that two vertices u and v are adjacent in $\vec{G}_V$ if there is an edge connecting them. A path l between two distinct vertices u and v is a sequence of distinct vertices in which the first vertex is u, the last one is v and two consecutive vertices are connected by an edge, that is, $l = (c_0 = u, c_1, \ldots, c_{m-1}, c_m = v)$ where $\langle c_{i-1}, c_i \rangle$ or $\langle c_i, c_{i-1} \rangle$ is contained in $\vec{E}_V$ for $i = 1, \ldots, m$ ($m \geq 1$), and $c_i \neq c_j$ for all $i \neq j$. We say that u is an ancestor of v and v is a descendant of u if there is a path between u and v in $\vec{G}_V$ and all edges on this path point at the direction toward v. The set of ancestors of v is denoted as $an(v)$, and we define $An(v) = an(v) \cup \{v\}$. A path l is said to be d-separated by a set of vertices Z if

(1) l contains a "chain": $u \to v \to w$ or a "fork" $u \leftarrow v \to w$ where v is in Z, or

(2) l contains a "collider" $u \to v \leftarrow w$ where v is not in Z and no descendant of v is in Z.

Two disjoint sets X and Y of vertices are d-separated by a set Z if Z d-separates every path from any vertex in X to any vertex in Y; We call Z a d-separator of X and Y. In $\vec{G}_V$, a collider $u \to v \leftarrow w$ is called a v-structure if u and w are non-adjacent in $\vec{G}_V$.

Let $\bar{G}_V = (V, \bar{E}_V)$ denote an undirected graph where $\bar{E}_V$ is a set of undirected edges. An undirected edge between two vertices u and v is denoted by (u,v). An undirected graph is called complete if any pair of vertices is connected by an edge. Define a moral graph $\bar{G}^m_V$ for a DAG $\vec{G}_V$ to be an undirected graph $\bar{G}^m_V = (V, \bar{E}_V)$ whose vertex set is V and whose edge set is constructed by marrying parents and dropping directions, that is, $\bar{E}_V = \{(u,v) : \langle u,v\rangle \text{ or } \langle v,u\rangle \in \vec{E}_V\} \cup \{(u,v) : (u,w,v) \text{ forms a } v\text{-structure}\}$ (Lauritzen, 1996). An undirected edge added for marrying parents is called a moral edge.

For an undirected graph, we say that vertices u and v are separated by a set of vertices Z if each path between u and v passes through Z. We say that two disjoint vertex sets X and Y are separated by Z if Z separates every pair of vertices u and v for any $u \in X$ and $v \in Y$. We call (A,B,C) a decomposition of $\bar{G}_V$ if

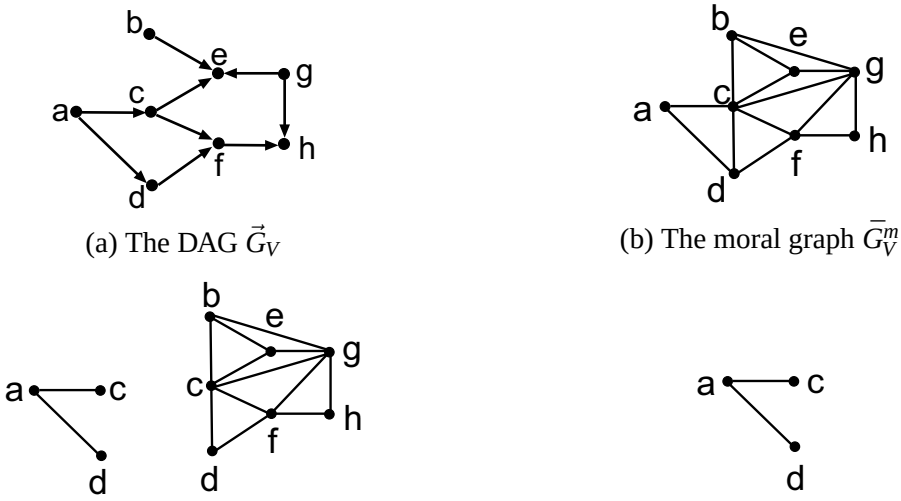
(1) $A \cup B \cup C = V$, and

(2) C separates A and B in $\bar{G}_V$.

Note that the above decomposition does not require that the separator C is complete, which is required for weak decomposition defined by Lauritzen (1996).

For a set $K \subseteq V$, we say that an undirected graph $\bar{G}_K$ is an undirected independence graph for a DAG $\vec{G}_V$ if that a set Z separates X and Y in $\bar{G}_K$ implies that Z d-separates X and Y in $\vec{G}_V$. An undirected independence graph is minimal if the proper subgraph obtained by deleting any edge is no longer an undirected independence graph. The moral graph $\bar{G}^m_V$ is the minimal undirected independence graph for $\vec{G}_V$ with $K = V$ (Lauritzen, 1996). It can also be obtained by connecting each vertex u with all vertices in its Markov blanket $Mb(u)$, which is the minimal set by which u are d-separated from the remaining set in V (that is, $V \setminus [Mb(u) \cup \{u\}]$). For a subset $K \subseteq V$, the Markov blanket for a vertex $u \in K$ can be defined similarly, that is, it is the minimum set that is contained in K and d-separates u from the remaining set in K. When $K = V$, it is easy to verify $Mb(u) = pa(u) \cup ch(u) \cup pa(ch(u))$. Define the local skeleton for a variable set $K \subseteq V$ with respect to $\vec{G}_V$ as an undirected graph $\bar{L}_K(K,E)$ where K is the vertex set and $E = \{(u,v) : \text{ no subset } S \text{ of } K \text{ } d\text{-separates } u \text{ and } v \text{ in } \vec{G}_V\}$ is the edge set. Note that though both minimal undirected independence graphs and local skeletons are undirected graphs and defined on the same vertex subset, they may be different. According to the definition of a minimal undirected independence graph, the absence or presence of an edge between u and v in the minimal undirected independence graph over $K \subseteq V$ depends on whether its two vertices are d-separated by the remaining set $K \setminus \{u,v\}$ in $\vec{G}_V$, while an edge between u and v in the local skeleton is determined by whether there exists a subset of K that can d-separate u and v in $\vec{G}_V$. Thus the edge set of the minimal undirected independence graph contains the edge set of the local skeleton.

The global skeleton is an undirected graph obtained by dropping the directions of the edges in a DAG, which coincides the local skeleton for $K = V$. Two DAGs over the same variable set



(a) The DAG $\vec{G}_V$

(b) The moral graph $\bar{G}_V^m$

(c) One decomposition based on $\bar{G}_V^m$

(d) A local skeleton

Figure 1: A directed graph, a moral graph, a decomposition and a local skeleton.

are called Markov equivalent if they induce the same conditional independence restrictions. Two DAGs are Markov equivalent if and only if they have the same global skeleton and the same set of *v*-structures (Verma and Pearl, 1990). An equivalence class of DAGs consists of all DAGs which are Markov equivalent, and it is represented as a partially directed graph (PDAG) where the directed edges represent arrows that are common to every DAG in the Markov equivalence class, while an undirected edge represents that the edge is oriented one way in some member of the Markov equivalence class, and is oriented the other way in some other member. Therefore the goal of structural learning is to construct a PDAG to represent the equivalence class.

Example 1. Consider the DAG in Figure 1 (a). $b \to e \leftarrow c$, $b \to e \leftarrow g$, $c \to f \leftarrow d$, $c \to e \leftarrow g$ and $f \to h \leftarrow g$ are *v*-structures. A path $l = (c,a,d)$ is d-separated by vertex a, while the path $l' = (c,f,h,g)$ is d-separated by an empty set. We have $an(e) = \{a,b,c,g\}$ and $An(e) = \{a,b,c,g,e\}$. The Markov blanket of c is $Mb(c) = \{a,b,d,e,f,g\}$, which d-separates c and the remaining set $\{h\}$. The moral graph $\bar{G}_V^m$ is given in Figure 1 (b), where edges (b,c), (b,g), (c,g), (c,d) and (f,g) are moral edges. Note that the set $\{c,d\}$ separates $\{a\}$ and $\{b,e,f,g,h\}$ in G_V^m, thus $(\{a\},\{b,e,f,g,h\},\{c,d\})$ forms a decomposition of the undirected graph G_V^m, the decomposed undirected independence subgraphs for $\{a,c,d\}$ and $\{b,c,d,e,f,g,h\}$ are shown in Figure 1 (c). The graph in Figure 1 (d) is the local skeleton $L_K(K,E)$ for $K = \{a,c,d\}$ because we have c and d are d-separated by $\{a\}$ in $\vec{G}_V$. Note that the minimal undirected independence graph for $\{a,c,d\}$ in Figure 1(c) coincides with its local skeleton in Figure 1 (d), which does not hold in general. For example, the local skeleton for $K = \{c,e,g\}$ does not have the edge (c,g), while the corresponding minimal undirected independence graph is complete.

Given a DAG $\vec{G}_V$, a joint distribution or density of variables $X_1,\ldots,X_N$ is

$$P(x_1,\cdots,x_N) = \prod_{i=1}^{N} P(x_i|pa_i),$$

where $P(x_i|pa_i)$ is the conditional probability or density of X_i given $pa(X_i) = pa_i$. The DAG $\vec{G}_V$ and the joint distribution P are said to be compatible (Pearl, 2000) and P obeys the global directed Markov property of $\vec{G}_V$ (Lauritzen, 1996). Let $X \perp\!\!\!\perp Y$ denote the independence of X and Y, and $X \perp\!\!\!\perp Y|Z$ the conditional independence of X and Y given Z. In this paper, we assume that all independencies of a probability distribution of variables in V can be checked by d-separations of $\vec{G}_V$, called the faithfulness assumption (Spirtes et al., 2000), which means that all independencies and conditional independencies among variables can be represented by $\vec{G}_V$. As a consequence, we also use $\perp\!\!\!\perp$ to denote the d-separation in DAGs.

3. A Recursive Method for Structural Learning of a DAG

In this section, we first present theoretical results in this paper and then we apply these results to structural learning of a DAG and show how the problem of searching for d-separators over the full set of all vertices can be recursively split into the problems of searching for d-separators over smaller subsets of vertices. We also discuss how to learn from data the undirected independence graphs which are used to achieve the recursive decomposition at each recursive step.

3.1 Theoretical Results and Recursive Algorithm for Structural Learning

Below we first give two theorems based on which we propose the recursive algorithm for structural learning of DAGs.

Theorem 1. Suppose that $A \perp\!\!\!\perp B|C$ in a DAG $\vec{G}_V$. Let $u \in A$ and $v \in A \cup C$. Then u and v are d-separated by a subset of $A \cup B \cup C$ if and only if they are d-separated by a subset of $A \cup C$.

According to Theorem 1, we can see that all edges falling in A or crossing A and C in the local skeleton $\overline{L}(K,E)$ with $K = A \cup C \cup B$ can be validly recovered from the marginal distribution of variables in $A \cup C$. Note that such a local skeleton over K can be used to recover the entire DAG over V even if there may not exist a marginalized DAG over K (Richardson and Spirtes, 2002).

Theorem 2. Suppose that $A \perp\!\!\!\perp B|C$ in a DAG $\vec{G}_V$. Let u and v be two vertices both of which are contained in the separator C. Then u and v are d-separated by a subset of $A \cup B \cup C$ if and only if they are d-separated by a subset of $A \cup C$ or by a subset of $B \cup C$.

According to Theorem 2, the existence of an edge falling into the separator C in the local skeleton $\overline{L}(K,E)$ with $K = A \cup C \cup B$ can be determined from the marginal distribution of $A \cup C$ or the marginal distribution of $B \cup C$.

Note that the union set $K = A \cup B \cup C$ in Theorems 1 and 2 may be a subset of the full set V (that is, $K \subseteq V$), and they are more general results than Theorem 1 presented in Xie et al. (2006), which requires that the union set K equals V (that is, $K = A \cup B \cup C = V$). These two theorems can guarantee that, for any partition (A,B,C) of a vertex set $K \subseteq V$ that satisfies $A \perp\!\!\!\perp B|C$, two non-adjacent vertices u and v in K are d-separated by a subset S of K in $\vec{G}_V$ if and only if they are d-separated by a subset S' of either $A \cup C$ or $B \cup C$ in $\vec{G}_V$. Therefore, we have the following result.

Theorem 3. Suppose that $A \perp\!\!\!\perp B|C$ in a DAG $\vec{G}_V$. Then the local skeleton $\overline{L}_K = (K, E_K)$ can be constructed by combining local skeletons $\overline{L}_{A\cup C} = (A \cup C, E_{A\cup C})$ and $\overline{L}_{B\cup C} = (B \cup C, E_{B\cup C})$ as follows:

(1) the vertex set $K = A \cup C \cup B$ and

(2) the edge set $E_K = (E_{A\cup C} \cup E_{B\cup C}) \setminus \{(u,v) : u,v \in C \text{ and } (u,v) \notin E_{A\cup C} \cap E_{B\cup C}\}$.

Based on these theorems, we propose a recursive algorithm for learning the structure of a DAG. Our algorithm has a series of operations on a binary tree. The top node of the tree is the full set of all variables, the leaves of the tree are subsets of variables which cannot be decomposed, and the variable set of each parent node in the binary tree is decomposed into two variable sets of its two children. Our algorithm consists of two steps: the top-down step for decomposing the full set of all variables into subsets as small as possible, and the bottom-up step for combining local skeletons into the global skeleton. At the top-down step, a variable set is decomposed into two subsets whenever a conditional independence $A \perp\!\!\!\perp B|C$ is found, and this decomposition is repeated until no new decomposition can be found. The decomposition at each step is done by learning an undirected independence graph over the vertex subset at the tree node, which will be discussed in Subsection 3.3. At the bottom-up step, two small skeletons are combined together to construct a larger skeleton, and the combination is repeated until the global skeleton is obtained. The entire process is formally described in the following algorithm.

Main Algorithm (*The recursive decomposition for structural learning of DAGs*)

1. Input: a target variable set V; observed data D.
2. Call DecompRecovery (V, $\bar{L}_V$) to get the global skeleton $\bar{L}_V$ and a separator list $\mathcal{S}$.
3. For each d-separator S_{uv} in the separator list $\mathcal{S}$, orient the local skeleton $u - w - v$ as a v-structure $u \rightarrow w \leftarrow v$ if $u - w - v$ (Note no edge between u and v) appears in the global skeleton and w is not contained in the separator S_{uv}.
4. Apply Meek's rule (Meek, 1995) to obtain a DAG in the Markov equivalence class: we orient other edges if each opposite of them creates either a directed cycle or a new v-structure. The Markov equivalence class can be obtained by collecting all possible DAGs.
5. Output: the equivalence class of DAGs.

PROCEDURE DecompRecovery (K, $\bar{L}_K$)

1. Construct an undirected independence graph $\bar{G}_K$;
2. If $\bar{G}_K$ has a decomposition (A, B, C)

   Then

   - For each pair (u, v) of $u \in A$ and $v \in B$, save $(u, v, S_{uv} = C)$ to the d-separator list $\mathcal{S}$;
   - DecompRecovery ($A \cup C$, $\bar{L}_{A\cup C}$);
   - DecompRecovery ($B \cup C$, $\bar{L}_{B\cup C}$);
   - Set $\bar{L}_K$ = CombineSubgraphs ($\bar{L}_{A\cup C}$, $\bar{L}_{B\cup C}$)

   Else

   - Construct the local skeleton $\bar{L}_K$ directly (such as using the IC algorithm):
     Start with a complete undirected graph over K.
     For any vertex pair (u, v) in the set K, if there exists a subset S_{uv} of $K \setminus \{u, v\}$ such that $u \perp\!\!\!\perp v|S_{uv}$, then delete the edge (u, v) and save (u, v, S_{uv}) to the d-separator list $\mathcal{S}$.

3. RETURN ($\bar{L}_K$).

FUNCTION CombineSubgraphs ($\bar{L}_U$, $\bar{L}_V$)

1. Combine $\bar{L}_U = (U, E_U)$ and $\bar{L}_V = (V, E_V)$ into an undirected graph $\bar{L}_{U\cup V} = (U \cup V, E_{U\cup V})$ where
$$E_{U\cup V} = (E_U \cup E_V) \setminus \{(u,v) : u,v \in U \cap V \text{ and } (u,v) \notin E_U \cap E_V\};$$
2. Return ($\bar{L}_{U\cup V}$).

As shown in the main algorithm, the equivalence class of $\vec{G}_V$ can be constructed by first calling DecompRecovery (V, $\bar{L}_V$) to get the skeleton, then recover all v-structures using the d-separator list $\mathcal{S}$ to orient the edges in $\bar{L}_V$, and finally orient other edges as much as possible using the rule in Meek (1995). Since a decomposition (A,B,C) of the undirected independence graph $\bar{G}_K$ implies $A \perp\!\!\!\perp B|C$, it is obvious by Theorems 1 and 2 that our algorithm is correct.

A binary decomposition tree is used in DecompRecovery to describe our algorithm simply and clearly. In our implementation, we use a junction tree to decompose a graph into several subgraphs simultaneously and to find the corresponding separators. It is known that the junction tree may not be unique, and thus we may have multiple decompositions. In theory, we prefer to use the junction tree with the minimum tree width. However, this is known to be an NP hard problem (Arnborg et al., 1987); therefore, we may use some sub-optimal method to consruct a junction tree for an undirected graph (Jensen and Jensen, 1994; Becker and Geiger, 2001). For example, two most well-known algorithms are the lexicographic search (Rose et al., 1976) and the maximum cardinality search (Tarjan and Yannakakis, 1984), whose computational expenses are $O(ne)$ and $O(n+e)$ respectively, where e is the number of edges in the graph. Especially, the latter method is used in our implementation. According to our experiences, the junction tree obtained by either method usually leads to very efficient decompositions.

In the recursive algorithm, statistical tests are used only at the top-down step but not at the bottom-up step. Thus the data sets used for statistical tests can be reduced into marginal data sets with decomposition of graphs. In this way, we only need to pass through small marginal data sets for statistical tests of subgraphs and need not pass through the full data set for every statistical test. Other algorithms (such as the PC algorithm) can be used to replace the IC algorithm to improve the performance of constructing the local skeleton $\bar{L}_K$ in DecompRecovery.

3.2 Tests of Conditional Independence

Conditional independence test of two variables u and v given a set C of variables is required at Step 1 and the 'Else' part of Step 2 of Procedure DecompRecovery to construct an undirected independence graph and a local skeleton respectively. Null hypothesis H_0 is $u \perp\!\!\!\perp v|C$ and alternative H_1 is that H_0 may not hold. Generally 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 more detailed discussion on this.

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 a 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.

For discrete data, the size of conditional variable sets cannot be so large that independence tests become inefficient. Thus the algorithm restricts the cardinality of conditioning sets. There are many methods that can be used to find a small conditioning set, such as a forward selection of variables. With the recursive decomposition, independence tests are localized to smaller and smaller subsets of variables, and thus the recursive algorithm has higher power for statistical tests.

3.3 Constructing Undirected Independence Graphs

In this subsection, we discuss how to construct undirected independence graphs at Step 1 of Procedure DecompRecovery. At first we call DecompRecovery with the full set V as the input argument, and construct an undirected independence graph $\bar{G}_V$ at Step 1. Then at each recursive calling, to construct a local undirected independence graph $\bar{G}_K$ with a subset K (say $K = A \cup C$) as the input argument, we shall present a theoretical result based on which we only need to check edges over the separator C without need of testing conditional independencies between any pair of variables in A and between any pair of variables crossing A and C.

To construct an undirected independence graph $\bar{G}_V$, we start with a complete undirected graph, and then we check an edge between each pair of vertices u and v. The edge (u,v) is removed if u and v are independent conditionally on the set of all other variables. For linear Gaussian models, the undirected graph can be constructed by removing an edge (u,v) if and only if the corresponding entry in the inverse covariance matrix is zero (Dempster, 1972; Whittaker, 1990). After decomposing a graph $\bar{G}_{A \cup B \cup C}$ into two subsets $A \cup C$ and $B \cup C$, we need to construct a local undirected independence graph $\bar{G}_K$ (say $\bar{G}_{A \cup C}$) at Step 1 of Procedure DecompRecovery. We show in the following theoretical result that an initial $\bar{G}_{A \cup C}$ can be constructed by using all undirected edges contained

by $A \cup C$ in the previous graph $\bar{G}_{A \cup B \cup C}$ and then only pairs of vertices contained in C need to be checked via conditional independence tests.

Theorem 4. Suppose that the distribution of $V = A \cup B \cup C$ is positive and has the conditional independence $A \perp\!\!\!\perp B | C$. Then for any u in A and any v in $A \cup C$, we have that $u \perp\!\!\!\perp v | [(A \cup C) \setminus \{u, v\}]$ if and only if $u \perp\!\!\!\perp v | [(A \cup B \cup C) \setminus \{u, v\}]$.

Note that Theorems 1 and 4 are different. The former is used to determine an edge in a DAG, and the latter is used to determine an edge in an undirected independence graph. According to this theorem, there exists an edge (u, v) in the minimal undirected independence graph $\bar{G}_{A \cup C}$ for u in A and v in $A \cup C$ if and only if there exists an edge (u, v) in the minimal undirected independence graph $\bar{G}_{A \cup B \cup C}$. Thus given an undirected independence graph $\bar{G}_{A \cup B \cup C}$ obtained in the preceding step, an undirected independence graph $\bar{G}_{A \cup C}$ has the same set of edges as $\bar{G}_{A \cup B \cup C}$ each of which has at least one vertex in A, but all of possible edges within the separator C need to be checked for $\bar{G}_{A \cup C}$.

When there is a large number of variables and a small sample size, it is infeasible or statistically unstable to test an independence between two variables conditionally on all other variables, and this problem is more serious when variables are discrete. Many current methods for learning undirected graphical models can also be used in our algorithm. For example, procedures based on *limited-order partial correlations* (Wille and Bühlmann, 2004; Castelo and Roverato, 2006) are rather suitable and can be even used in the case where the number of variables is larger than the number of samples. Another way of learning undirected independence graphs is to apply current available Markov blanket learning algorithms. By connecting each vertex with those in its Markov blanket, an independence graph is then obtained. Indeed, it is neither new nor uncommon to learn the Markov blanket as either an initial step for learning a DAG or as a special problem of interest. Koller and Sahami (1996) developed a method for feature selection which employs the concept of Markov blanket. Margaritis and Thrun (1999) proposed a two-phase algorithm to first identify a Markov blanket for each variable and then obtain a DAG by connecting vertices in a maximally consistent way. Tsamardinos et al. (2003) proposed a method that can soundly identify all Markov blankets and scale-up to a graph with thousands of variables.

Another particular method for learning the undirected independence graph may use Lasso-type estimators (Tibshirani, 1996; Meinshausen and Bühlmann, 2006; Zhao and Yu, 2006; Wainwright et al., 2006). We can apply Lasso method to select a neighborhood set of a vertex which contains the Markov blanket of the vertex. Schmidt et al. (2007) developed a new method of learning structure of a DAG. Note that it is not necessary to learn neighborhoods exactly in our algorithm, and there may be extra edges in our undirected independence graph.

4. Illustration and Evaluation of the Recursive Algorithm

In this section, we first illustrate the recursive algorithm step by step via a concrete example and then show simulation results to evaluate its performance.

4.1 Illustration of the Recursive Algorithm

In this subsection, we illustrate our recursive algorithm using a concrete example. We suppose in the following example that conditional independencies can be implemented correctly, that is, each conditional independence is checked by using the underlying DAG. Therefore the purpose of the example is simply to illustrate the overall scheme of the recursive algorithm presented in Section 3.1. The performance of conditional independence tests is discussed in the next subsection. We

compare the recursive algorithm with the decomposition algorithm proposed in Xie et al. (2006), in which an entire undirected independence graph is first constructed and then it is decomposed into many small subgraphs at one step instead of recursive steps. We show that, in our algorithm, search for separators is localized to smaller vertex subsets than those obtained by using the decomposition algorithm.

Example 1. (Continued) Consider again the DAG $\vec{G}_V = (V, \vec{E}_V)$ in Figure 1 (a). We call Procedure DecompRecovery to construct the global skeleton over V. At the top-down step (that is, at the 'Then' part of Step 2 in DecompRecovery), we construct the binary tree shown in Figure 2. At the top of the binary tree, the first decomposition is done by splitting the full vertex set V in G_1 (that is, the moral graph) into two subsets $\{a,c,d\}$ and $\{b,c,\ldots,h\}$ with the separator $\{c,d\}$. Next we learn the undirected independence graphs G_2 and G_3 for the two subsets separately. To construct the subgraphs G_2 and G_3, by Theorem 5, we only need to check the edge (c,d) in the separator $\{c,d\}$, and other edges in G_2 and G_3 can be obtained directly from G_1. Repeat this procedure until no further decomposition is possible. Finally we get the entire binary tree T as shown in Figure 2, where each leaf node is a complete graph and cannot be decomposed further.

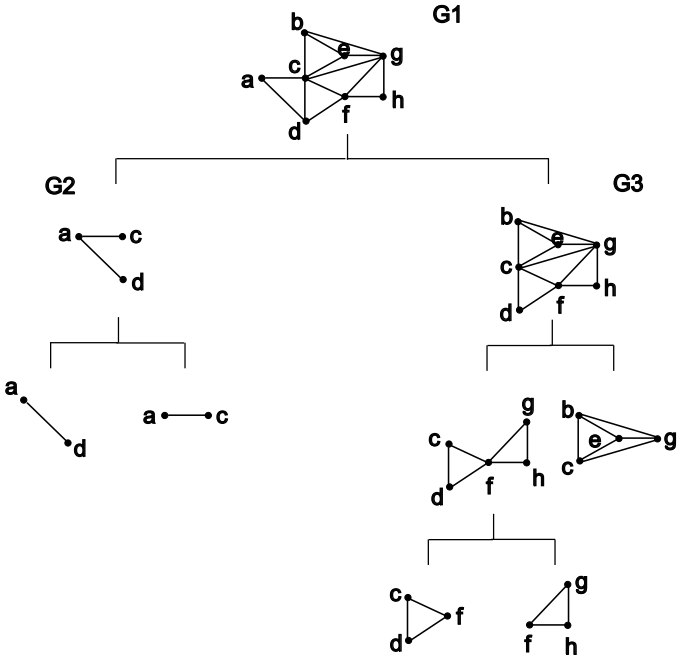


Figure 2: The binary tree T obtained at the top-down step (at 'Then' of Step 2).

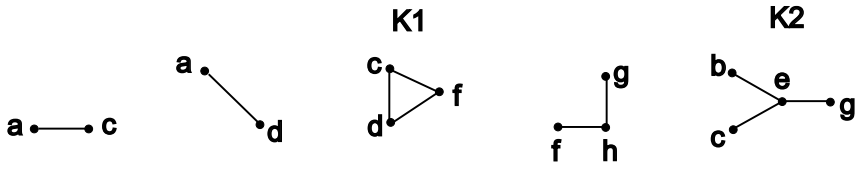


Figure 3: The local skeletons obtained at 'Else' of Step 2.

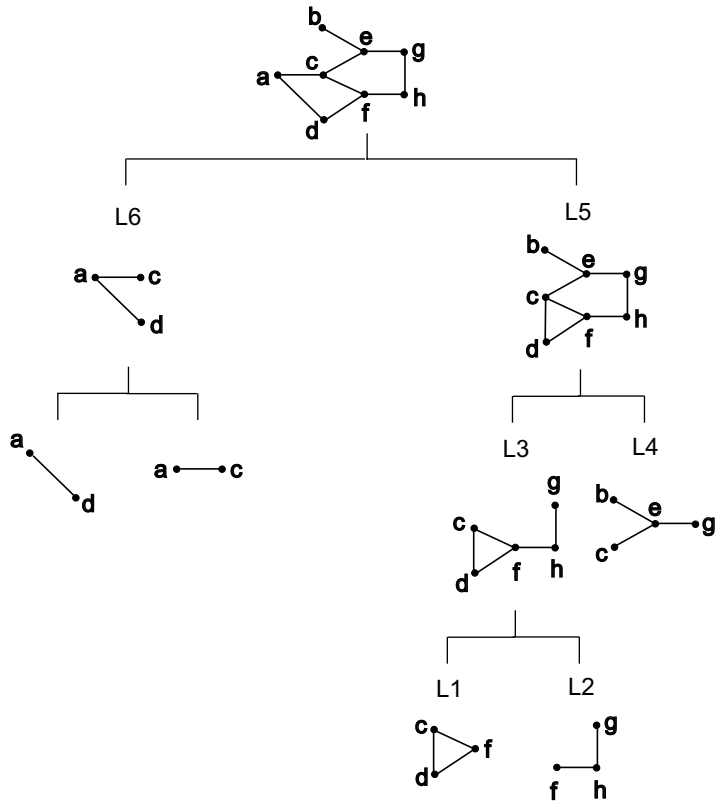


Figure 4: Combinations of local skeletons in Procedure CombineSubgraphs.

Before the bottom-up step (that is, the 'Else' part of Step 2 in Procedure DecompRecovery), for each leaf node K, we construct a local skeleton over K. For each vertex pair (u,v) in K, we search a separator set S_{uv} in all possible subsets of $K \setminus \{u,v\}$ to construct the local skeleton. All local skeletons of leaf nodes are shown in Figure 3. For example, the vertices c and d are adjacent in the local skeleton K_1 since no vertex set in K_1 d-separates them, whereas b and g are non-adjacent in the local skeleton K_2 since an empty set d-separates them in $\vec{G}_V$. At the bottom-up step, calling Function CombineSubgraphs, we combine the local skeletons from the leaf nodes to the root node to form the global skeleton, as shown in Figure 4. For example, local skeletons L_1 and L_2 are combined to L_3, and then L_3 and L_4 are combined to L_5, as shown in Figure 4. Similarly, we get the local skeleton L_6. At the last step, we combine L_5 and L_6 into the global skeleton. Note that the edge (c,d) in L_5 is deleted at Step 1 of Function CombineSubgraphs since the edge is not contained in L_6. After all the combinations are done, we get the global skeleton in Figure 5. We can see that the undirected independence graphs and the local skeletons are different as shown in Figure 2 and Figure 4 respectively and that the former has more edges than the latter.

At Step 2 of Procedure DecompRecovery, we save all separators to the d-separator list $\mathcal{S}$. At Step 3 of the main Algorithm, we use separators in the list $\mathcal{S}$ to recover all v-structures of the DAG. For example, there is a d-separator $\{a\}$ in $\mathcal{S}$ which d-separates c and d, and there is a structure $c - f - d$ in the global skeleton $\bar{L}_V$ where f is not contained in the separator $\{a\}$. Thus we can orient the structure $c - f - d$ as a v-structure $c \to f \leftarrow d$. Similarly, since an empty set d-separates

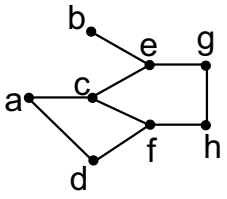


Figure 5: The global skeleton $\bar{L}_V$.

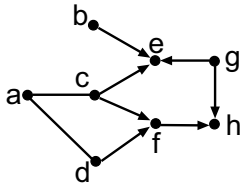


Figure 6: The recovered equivalence class.

b and g in $\vec{G}_V$, we can orient $b-e-g$ as $b \to e \leftarrow g$. After recovering all *v*-structures, we apply the orientation rule in Meek (1995) and get the desired equivalence class of $\vec{G}_V$ in Figure 6. In this equivalence class, the undirected edge (a,c) cannot be oriented uniquely because any of its orientation leads to a Markov equivalent DAG.

Below we compare the recursive algorithm with the decomposition algorithm proposed in Xie et al. (2006). We show that theoretically the recursive algorithm can decompose the entire graph into smaller subgraphs than the decomposition algorithm does because the decomposition in the decomposition algorithm is done only once, whereas the recursive algorithm tries to re-decompose undirected independence subgraphs at each recursive step. When there are a lot of *v*-structures in a DAG, many moral edges can be deleted in construction of a subgraph, and thus the recursive algorithm is more efficient than the decomposition algorithm. The following example illustrates the difference of decompositions obtained by these two algorithms.

Example 2. Consider the DAG in Figure 7. By using the decomposition algorithm proposed in Xie et al. (2006), a '*d*-separation tree' is built from an undirected independence graph (that is, the moral graph in this example), and the full variable set is decomposed into three subsets of variables at one time, see Figure 8 (a). By using the recursive algorithm proposed in this paper, we can decompose the graph into four subgraphs in Figure 8 (b), which have smaller subsets of variables. This is because the undirected independence graph over $\{a,b,c\}$ in Figure 8 (b) is re-constructed and the edge (b,c) is deleted for $b \perp\!\!\!\perp c|a$.

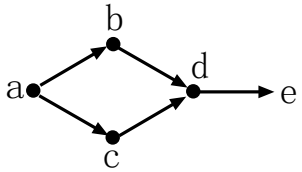
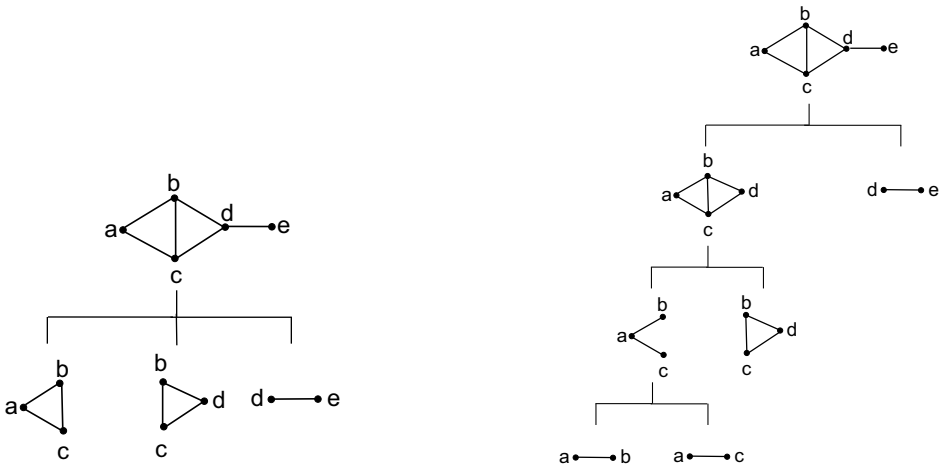


Figure 7: A DAG.



(a) The decomposition algorithm.

(b) The recursive algorithm.

Figure 8: Comparison of two different algorithms for structural learning.

4.2 Simulation Studies

Below we give numerical examples to evaluate the performance of the recursive algorithm. We first present simulation results for the ALARM network, which is a medical diagnostic network and is shown in Figure 9 (Beinlich et al., 1989; Heckerman, 1998). It is a DAG with 37 vertices and 46 edges and it is often used to evaluate performance of learning algorithms. In the following two subsections, we use the ALARM network to do simulation for the Gaussian case and the discrete case separately. Next we show simulation results for several other networks in the final subsection.

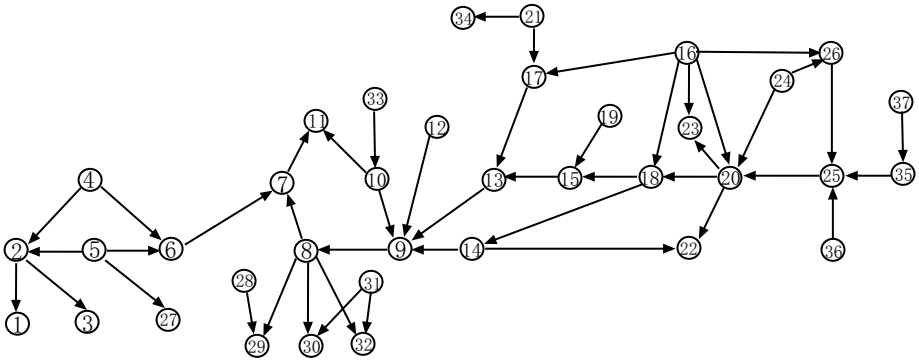


Figure 9: The ALARM network.

4.2.1 THE GAUSSIAN CASE

In this subsection, for the underlying DAG of the ALARM network, we generate a sample from a joint Gaussian distribution using a structural equation model of recursive linear regressions, whose coefficients are randomly generated from the uniform distribution in the interval $(-1.5,-0.5)\cup(0.5,1.5)$ and the residual variance is 1 for each linear regression. We apply the recursive algorithm to the generated sample to construct a DAG, and then we compare the underlying DAG with the

constructed DAG and record the number of extra edges, the number of missing edges and the structural hamming distance (SHD), where SHD is defined as the total number of operations to verify the constructed PDAG to the Markov equivalence class of the underlying DAG, and where the operations may be: add or delete an undirected edge, and add, remove or reverse an orientation of an edge (Tsamardinos et al., 2006). The likelihood ratio test introduced in Subsection 3.2 is used to test the partial correlation coefficient at the significance level $\alpha = 0.01$. We repeatedly draw $n = 1000$ sets of samples and obtain the average numbers of extra edges, missing edges and SHD from $n = 1000$ simulations. The first 3 simulation results are shown in Table 1 for different sample sizes 1000, 2000, 5000 and 10000. In Table 1, three values in a bracket denote the number of extra edges, the number of missing edges and SHD respectively. The column 'Ave' in Table 1 shows the averages of $n = 1000$ simulations. It can be seen that the algorithm performs better as the sample size increases. From the simulations, we found that most decompositions at the top-down step are correct, and we also found that when coefficients make the faithfulness assumption close to fail (that is, some of the edges only reflect weak or nearly zero associations), the learned PDAG from simulation may not be exactly the same as the underlying PDAG, and most of edge mistakes appear for these edges that represent rather weak associations.

Sample Size	1	2	3	Ave
1000	(2, 2, 13)	(0, 1, 6)	(2, 4, 10)	(1.20, 2.70, 12.8)
2000	(0, 1, 5)	(1, 0, 5)	(0, 1, 8)	(0.96, 1.77, 9.12)
5000	(1, 2, 5)	(0, 0, 2)	(1, 2, 4)	(0.85, 1.07, 6.18)
10000	(0, 1, 2)	(0, 0, 0)	(0, 2, 4)	(0.75, 0.77, 4.99)

Table 1: Extra edges, missing edges, and SHD for the first 3 simulations and averages from 1000 simulations.

Our implementation is based on the Bayesian network toolbox written by Murphy (2001) and the simulations run particularly fast. For a single simulation for all sample sizes $N = 1000$, 2000, 5000, 10000, when conditional independence tests are used to check edges, it costs only around 3 seconds in Matlab 7 on a laptop Intel 1.80GHz Pentium(R)M with 512 MByte RAM running Windows XP.

We also compare our methods with the PC algorithm (Spirtes and Glymour, 1991) and the Three Phaze Dependency Analysis (TPDA) algorithm (Cheng et al., 2002), which are readily available in the Causal Explorer System developed by Aliferis et al. (2003). The simulation is repeated 100 times for each of different network parameters and sample sizes. For each generated data set, the structure learned from each method is then compared with the true underlying structure. For each algorithm, we choose two different significance levels, that is, $\alpha = 0.01$ and 0.05. In the second row of Table 2, the underlined values in a bracket denote the number of extra edges, the number of missing edges and SHD respectively, and other rows give values relative to the second row,

Alg (Level α)	N = 1000	N = 2000	N = 5000	N = 10000	Ave Time
	(1.2, 2.4, 12)	(1.0, 1.5, 8.2)	(0.9, 0.9, 5.8)	(0.7, 0.6, 4.6)	2.55 sec
Rec(0.01)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	(1.0 ,1.0, 1.0)	1.0
Rec(0.05)	(3.1, 1.0, 1.5)	(3.5, 1.1, 1.7)	(4.2, 1.1, 2.2)	(4.7, 1.2, 2.3)	1.5
PC(0.01)	(1.8, 4.5, 3.5)	(2.6, 6.3, 4.7)	(2.8, 8.0, 6.0)	(4.0, 9.7, 7.2)	21.2
PC(0.05)	(2.9, 4.0, 3.5)	(4.3, 3.5, 4.7)	(6.0, 4.0, 6.2)	(8.6, 5.1, 7.3)	24.8
TPDA(0.01)	(3.9, 4.1, 3.5)	(4.3, 5.7, 4.8)	(3.9, 7.5, 6.1)	(3.7, 8.5, 6.8)	73.6
TPDA(0.05)	(4.4, 3.7, 3.5)	(4.7, 5.1, 4.7)	(5.7, 4.3, 6.0)	(8.5, 4.8, 7.1)	88.3

Table 2: Results relative to the recursive algorithm with $\alpha = 0.01$ and $\alpha = 0.05$: extra edges, missing edges, and SHD

which are obtained by dividing their real values by the underlined values in the second row. A relative value larger than 1 denotes that its real value is larger than the corresponding value in the second row. For example, the third row labeled *Rec*(0.01) with all values equal to 1 shows that our algorithm with $\alpha = 0.01$ has the same results as the second row; the seventh row labeled PC(0.01) shows the relative results for the PC algorithm with $\alpha = 0.01$, where $(1.8, 4.5, 3.5)$ means the real values as $(1.8 \times 1.2, 4.5 \times 2.4, 3.5 \times 12)$. The last column labeled 'Ave Time' denotes average time cost for one simulation of all 4 sample sizes. In Table 2, all values are larger than 1, which means our algorithm Rec(0.01) has the least number of extra edges, the least number of missing edges and the least SHD, and further it costs the least times.

4.2.2 THE DISCRETE CASE

Now we show simulations of the ALARM network for the discrete case where these discrete variables have two to four levels. For every simulation, the conditional probability distribution of each variable X_i given its parents pa_i is draw randomly in the following way: for each fixed configuration pa_i of the parents, we first generate a sequence $\{r_1, \ldots, r_L\}$ of random numbers from the uniform distribution $U(0,1)$, where L is the number of levels of X_i; then let $P(X_i = j | pa_i) = r_j / \sum_k r_k$ as the distribution of X_i conditional on the fixed configuration pa_i of X_i's parents. Note that the joint distribution generated in this way may be unfaithful, which together with the problem of discreteness makes the learning task harder than that for the Gaussian case. We run 100 simulations for each sample size $N = 1000$, 2000, 5000 or 10000, and then we compare our method with several other algorithms by averages from 100 simulations. In addition to the PC algorithm and the TPDA algorithm used in the Gaussian case, we also compare our method with the Sparse Candidate (SC) algorithm (Friedman et al., 1999) and the MMHC algorithm (Tsamardinos et al., 2006). For the PC

algorithm, we set the parameter max-fan-in (that is, the maximum in-degree) to its true value so that the PC algorithm can run fast. We use the TPDA and the MMHC algorithms that are implemented in the Causal Explorer System (Aliferis et al., 2003) with the default setting. For all algorithms except the SC algorithm, we use two significance levels ($\alpha = 0.01$, $\alpha = 0.05$) in the simulations. For the SC algorithm, the most important parameter to be specified is the number of candidates (the maximum size of potential parent sets), which are set to 5 and 10 separately.

Alg (Level α)	N = 1000	N = 2000	N = 5000	N = 10000	Ave Time
	(1.3, 10, 33)	(1.2, 6.6, 23)	(0.8, 4.0, 16)	(0.7, 2.6, 11)	27 sec
Rec(0.01)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	1.0
Rec(0.05)	(4.0, 0.8, 1.1)	(3.6, 0.8, 1.2)	(4.8, 0.8, 1.3)	(4.9, 0.8, 1.4)	1.3
PC(0.01)	(0.2, 1.5, 1.9)	(0.1, 1.5, 2.6)	(0.2, 1.6, 3.3)	(0.1, 1.8, 4.5)	1.7
PC(0.05)	(0.8, 1.2, 1.9)	(0.5, 1.3, 2.5)	(0.7, 1.3, 3.2)	(0.7, 1.5, 4.4)	1.9
TPDA(0.01)	(10.2, 1.2, 2.7)	(2.7, 1.6, 3.0)	(1.9, 2.7, 4.1)	(0.9, 4.1, 5.8)	0.9
TPDA(0.05)	(0.0, 2.5, 2.2)	(0.1, 3.9, 3.1)	(0.1, 6.5, 4.5)	(0.1, 9.9, 6.6)	0.2
SC(5)	(2.0, 0.8, 0.8)	(2.4, 0.8, 1.0)	(3.6, 0.9, 1.1)	(4.3, 0.9, 1.4)	4.7
SC(10)	(2.2, 0.8, 0.8)	(2.7, 0.9, 1.0)	(3.8, 0.8, 1.1)	(4.2, 0.7, 1.2)	6.6
MMHC(0.01)	(0.3, 1.3, 1.1)	(0.3, 1.4, 1.1)	(0.4, 1.5, 1.1)	(0.7, 1.5, 1.2)	1.1
MMHC(0.05)	(0.5, 1.2, 1.0)	(0.4, 1.3, 1.0)	(0.5, 1.3, 1.0)	(1.0, 1.3, 1.1)	1.4

Table 3: Results relative to the recursive algorithm with $\alpha = 0.01$ and $\alpha = 0.05$: extra edges, missing edges, and SHD

We summarize the simulation results in Table 3. In terms of SHD, our algorithm, the SC algorithm and the MMHC algorithm perform better than the PC and TPDA algorithms. It can also be seen that the performance difference between our method and the others becomes larger as the sample size increases. Although it can be seen from the last column labeled 'Ave Time' that the average CPU time cost for our algorithm is the second least, the fastest algorithm TPDA has the largest SHD among all algorithms. From the results in Tables 3, we can see that although the recursive algorithm seems to have a better performance in most cases, it is still not quite clear which one of these algorithms is superior in general. Their performance depends on preference of reducing the false positive error (including an edge that is not in the true DAG) or the false negative error

(excluding an edge that is in the true DAG). For example, the PC and MMHC algorithms have a smaller false positive error; the SC algorithm has a smaller false negative error; and the recursive algorithm has smaller SHD. We also found that choosing a good parameter is also important to achieve an optimal performance for each algorithm. The recursive algorithm seems to work better when we choose a significance level $\alpha = 0.01$, while for MMHC it is better to choose $\alpha = 0.05$.

The above comparison is based on results from randomly generated values of parameters of joint distributions. The results may change when different values of these parameters are used. The performance of an algorithm also depends on the structures of a network.

4.2.3 SIMULATIONS OF OTHER NETWORKS

In this subsection we show simulation results for other three networks: Insurance with 27 vertices and 52 edges (Binder et al., 1997), HailFinder with 56 vertices and 66 edges (Abramson et al., 1996) and Carpo with 61 vertices and 74 edges, all of which can be obtained through the online Bayesian network repository (http://www.cs.huji.ac.il/labs/compbio/ Repository). We compare the recursive algorithm with the SC and MMHC algorithms since these two have been extensively compared with many state-of-art algorithms and shown in general outperforming other algorithms by Tsamardinos et al. (2006). In our simulations, the parameter values of the joint distributions are set to the original values from the repository. For each network, 10 data sets are generated, and we give one better result in Table 4 for each algorithm with two criteria ($\alpha = 0.01$ and 0.05 for Rec and MMHC, the number of candidates = 5 and 10 for SC). From the last column 'Ave Time' of Table 4, it can be seen that the recursive algorithm is fastest in average CPU time and it also has a better performance in most cases for these networks.

4.3 Complexity Analysis

Below we discuss the complexity of the recursive algorithm proposed in this paper. We mainly focus on the number of conditional independence tests for constructing the equivalence class since decomposition of graphs is a computationally simple task compared to the conditional independence tests. In the recursive algorithm DecompRecovery, two steps (Step 1 for constructing an undirected independence graph $\bar{G}_K$ and the 'Else' part of Step 2 for constructing a local skeleton $\bar{L}_K$) involve conditional independence tests, where K is the vertex set of the subgraph. At Step 1, an undirected independence graph can be constructed by testing independence between any pair of variables conditionally on other variables, and thus the complexity is $O(|K|^2)$, where $|K|$ denotes the number of vertices in the set K. As discussed in Section 3.3, an undirected independence graph $\bar{G}_{A\cup C}$ can be constructed from the previous graph $\bar{G}_{A\cup B\cup C}$ by checking only all possible edges within the separator C. Thus the complexity for constructing an undirected independence graph can be reduced. At Step 2, we construct a local skeleton over a vertex subset K. Suppose that we use the IC algorithm. Then the complexity for constructing the local skeleton L_K is $O(|K|^2 2^{|K|-2})$. Below we consider the total expenses and suppose that the full vertex set V is recursively decomposed into H subsets $\{K_1,\ldots,K_H\}$, where $H \le n$ and $K_h \le n$ for all h. For each decomposition, we need to construct an undirected independence graph, and thus the total expenses for all decompositions is less than $O(Hn^2)$. The total expenses for constructing all skeletons is $O(\sum_h |K_h| 2^{|K_h|-2})$, which is less than $O(Hk_{max}2^{k_{max}-2})$, where $k_{\max} = \max\{|K_1|,\ldots,|K_H|\}$. The complexity for the IC algorithm is known to be $O(n^2 2^{n-2})$. Since $K_{\max}$ usually is much less than n, the recursive decomposition can greatly reduce the complexity of the IC algorithm. Of course, when no decomposition is available,

Alg (Level α)	N = 1000	N = 2000	N = 5000	N = 10000	Ave Time
			Insurance		
	(2.4, 13, 43)	(1.5, 10, 40)	(1.3, 7.4, 32)	(1.1, 6.7, 27)	16 sec
Rec(0.01)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	1.0
SC(5)	(1.6, 1.0, 1.0)	(2.3, 1.2, 1.2)	(2.5, 1.3, 1.3)	(2.9, 1.4, 1.5)	6.7
MMHC(0.05)	(0.6, 1.3, 1.1)	(1.1, 1.4, 1.2)	(1.2, 1.5, 1.2)	(1.1, 1.4, 1.2)	8.0
			Hailfinder		
	(5.9, 16, 53)	(7.1, 14, 47)	(8.0, 14, 43)	(7.3, 14, 41)	62 sec
Rec(0.01)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	1.0
SC(10)	(2.0, 1.0, 1.1)	(1.8, 1.1, 1.1)	(2.0, 1.1, 1.3)	(2.1, 0.8, 1.2)	5.6
MMHC(0.05)	(1.6, 1.2, 1.1)	(1.6, 1.1, 1.0)	(1.7, 1.1, 1.2)	(1.0, 1.9, 1.2)	17.4
			Carpo		
	(10, 12, 49)	(9.0, 5.0, 36)	(6.5, 2.6, 21)	(6.3, 1.0, 18)	74 sec
Rec(0.01)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	(1.0, 1.0, 1.0)	1.0
SC(10)	(2.3, 0.5, 1.2)	(2.6, 0.5, 1.7)	(2.8, 0.9, 2.2)	(2.3, 1.3, 2.0)	6.6
MMHC(0.05)	(2.5, 2.4, 2.1)	(2.6, 4.5, 2.6)	(3.1, 6.0, 3.4)	(3.0, 12, 3.4)	44

Table 4: Results relative to the recursive algorithm for other networks: extra edges, missing edges, and SHD

the complexity of our algorithm becomes the same as the IC algorithm, which reflects the fact that structural learning of DAGs is an NP-hard problem (Chickering et al., 2004). Similarly, the recursive decomposition can also be used to improve the performance of the PC algorithm and other algorithms.

5. Conclusion

In this paper, we proposed a recursive algorithm for structural learning of DAGs. We first present its theoretical properties, then show its experimental results and compare it with other algorithms. In the recursive algorithm, a structural learning for a large DAG is first split recursively into those for small subgraphs until each subgraph cannot be decomposed further, then we perform local learning for these subgraphs which cannot be decomposed, finally we gradually combine these locally learned subgraphs into the entire DAG. The main problem for structural learning of a DAG is the search for d-separators, which becomes exponentially complicated with the number of vertices increases. In the recursive algorithm, all searches for d-separators are localized into subsets of vertices. Thus the efficiency of structural learning and the power of statistical tests can be improved by decomposition.

There are several works related to our recursive approach. Friedman et al. (1999) discussed how the idea of recursive decomposition can be used in accelerating their Sparse Candidate algorithm, Narasimhan and Bilmes (2005) discussed the application of this idea to find a sub-optimal graphical models by noticing the corresponding decomposition of the Kullback and Leibler divergence (Kullback and Leibler, 1951) with respect to the graph separation. Geng et al. (2005) and Xie et al. (2006) proposed the decomposition algorithms for structural learning of DAGs. However, the method proposed in Geng et al. (2005) requires that each separator has a complete undirected graph. Xie et al. (2006) removed the condition, but their algorithm performs decomposition only based on the entire undirected independence graph $\overline{G}_V$ of the full vertex set V and cannot perform decomposition of undirected independence subgraphs. Theorems 1, 2 and 3 in this paper relax this requirement, and they do not require the union set $K = A \cup B \cup C$ of a decomposition (A,B,C) to be equal to the full vertex set V. Thus the recursive algorithm can delete more edges in undirected independence subgraphs and further decompose them, see Example 2. Theorems 1, 2 and 3 are also useful properties for collapsibility of DAGs.

Now we discuss several potential utilities and further works of the recursive approach. This recursive decomposition approach can also be used to localize a learning problem of interest. Suppose that V is the full set of all observed variables, but we are interested only in a local structure over a variable subset A. Using the recursive approach, we can recursively decompose the variable sets into small sets, only focus on the subtrees that contain variables in A, and ignore other subtrees that are unrelated to A. In such a way, the local structure over A can be obtained without need of learning other structures that are unrelated to A. The recursive approach can also use a prior knowledge of independencies among variables to decompose structural learning.

Acknowledgments

We would like to thank the editor and the three referees for their helpful comments and suggestions that greatly improved the previous version of this paper. This research was supported by NSFC, NBRP 2003CB715900, 863 Project of China 2007AA01Z437 and MSRA. We would also like to thank Professor Rich Maclin, the publication editor, for his help with the revision.

Appendix A.

We first give some lemmas which will be used in proofs of theorems.

Lemma 1. A subset S of vertices separates u from v in $[\vec{G}_{An(\{u\}\cup\{v\}\cup S)}]^m$ if and only if $u \perp\!\!\!\perp v|S$.
Proof. The result can be obtained directly from Proposition 3.25 of Lauritzen (1996) and Theorem 1.2.4 of Pearl (2000). □

Lemma 2. Let S be a subset of V. Then two vertices u and v in S are d-separated by a subset of S if and only if they are d-separated by $an(\{u,v\}) \cap S$.
Proof. Define $S' = an(\{u,v\}) \cap S$. The necessity is obvious since $S \supseteq S'$. For sufficiency, suppose that u and v are not d-separated by S'. Since $An(\{u,v\} \cup S') = An(\{u,v\})$, we have from Lemma 1 that there is a path l connecting u and v in $[\vec{G}_{An(\{u,v\})}]^m$ which is not separated by S' in the moral graph, that is, the path l does not contain any vertex in S'. Since l is contained in $[\vec{G}_{An(\{u,v\})}]^m$ and $S' = an(\{u,v\}) \cap S$, we then have that l does not contain any vertex in $S \setminus \{u,v\}$. Now from the condition, suppose that u and v are d-separated by $S_0 \subseteq S$. Then we also have from $an(u,v) \cap S_0 \subseteq S'$ that l does not contain any vertex in $an(u,v) \cap S_0$. Thus we obtain that l is not separated by S_0 in $[\vec{G}_{An(u,v,S)}]^m$, which by Lemma 1 implies that u and v are not d-separated by S_0. However, this contradicts the condition that u and v are d-separated by $S_0 \subseteq S$, which concludes the proof for Lemma 2. □

Lemma 3. If four disjoint sets X, Y, Z and W satisfy $X \perp\!\!\!\perp Y \cup Z|W$, then we have $X \perp\!\!\!\perp Y|Z \cup W$.
Proof. This result is obvious. □

Under the faithfulness assumption, a conditional independence is equivalent to the corresponding d-separation, and thus d-separation also has the above property.

Lemma 4. Suppose that l is a path that connects two nonadjacent vertices u and v. If l is not contained completely in $An(u) \cup An(v)$, then l is d-separated by any subset S of $an(u) \cup an(v)$.
Proof. Since l is not completely contained in $An(u) \cup An(v)$, there exists vertices m and n in $l = (u,\ldots,m,x,\ldots,y,n,\ldots,v)$ such that both m and n are contained in $An(u) \cup An(v)$ and no vertices from x to y are contained in $An(u) \cup An(v)$ where x and y, u and m, n and v may be separately the same vertex. So we have that the arrows must be oriented as $\langle m,x\rangle$ and $\langle n,y\rangle$, and then there must be a collider between m and n on l. Let $s \to w \leftarrow t$ be the collider that is closest to m. Then we have that the sub-path of l from m to w is directed. Notice that $m \in An(u) \cup An(v)$ and $w \notin An(u) \cup An(v)$. Thus we obtain that S and its subset do not contain the middle vertex w or its descendants, which implies that l is d-separated by any subset of S at the collider $s \to w \leftarrow t$. □

Proof of Theorem 1: The necessity is obvious since $(A \cup B \cup C) \supseteq (A \cup C)$. For sufficiency, let a and d be two vertices in A and $A \cup C$ respectively that are d-separated by a subset of $A \cup B \cup C$. Define $W = (an(a) \cup an(d)) \cap (A \cup B \cup C)$. By Lemma 2, a and d must be d-separated by W. Define $S' = (an(a) \cup an(d)) \cap (A \cup C)$. Then we only need to show that S' ($\subseteq A \cup C$) can d-separate every path l connecting a and d in $\vec{G}_V$. We consider the following two cases separately:

(1) a path l is not contained completely in $An(a) \cup An(d)$, and

(2) a path l is contained completely in $An(a) \cup An(d)$.

For case (1), we get from Lemma 4 that l must be d-separated by S' since S' is a subset of $an(a) \cup an(d)$.

For case (2), we have from condition $A \perp\!\!\!\perp B|C$ that $[\{a\} \cup (S' \cap A)] \perp\!\!\!\perp b|C$ for any $b \in B$, which implies, by Lemma 3, $a \perp\!\!\!\perp b|(S' \cap A) \cup C$. Since $S' \subseteq (A \cup C)$, we get

$$a \perp\!\!\!\perp b|(S' \cup C).$$

By reduction to absurdity, suppose that there is a path l contained in $An(a) \cup An(d)$ connecting a and d which cannot be d-separated by S'. Because W $(\supseteq S')$ d-separates a and d and thus d-separates l but S' does not, there must exist at least one vertex on the path l which is contained in $W \setminus S'$ $(\subseteq B)$. Let b be such a vertex that is closest to a on the path l and define l' to be the sub-path of l from a to b. It is obvious that l' is d-connected by S'; otherwise l will be d-separated by S'. Since b is closest to a on l and $b \in B$, any of other vertices on l' is not in B. From $l' \subseteq l \subseteq (An(a) \cup An(d))$ and $S' = (an(a) \cup an(d)) \cap (A \cup C)$, we have that all vertices of l' except a and b are contained in S'. Since l' is d-connected by S', l' is also d-connected by $S' \cup C$, which contradicts (A.1). Thus we showed that every path in case (2) is also d-separated by S', which concludes our proof for Theorem 1. □

The following lemma, which is non-trivial due to the fact that a sequence can contain the same vertex more than once, indicates that the d-separation for a path can be made equivalent to that for a sequence.

Lemma 5. Two non-adjacent vertices u and v are d-separated by S in $\vec{G}_V$ if and only if for any sequence $l = (u, \ldots, v)$ connecting u and v

1. l contains a "chain" $i \to m \to j$ or a "fork" $i \leftarrow m \to j$ such that the middle vertex m is in S, or

2. l contains a "collider" $i \to m \leftarrow j$ such that the collision vertex m is not in S and no descendant of m is in S.

When a sequence $l = (u, \ldots, v)$ satisfies the above conditions 1 and 2, we also say that the sequence l is d-separated by S.

Proof. The sufficiency is obvious from definition of d-separation. For necessity, suppose there are sequences connecting u and v that satisfy neither condition 1 nor 2. Let $l = (z_0 = u, z_1 \ldots, z_{k-1}, z_k = v)$ be the shortest one of such sequences, it's easy to show that such a sequence is itself a path which contradicts with the condition that u and v are d-separated by S in $\vec{G}_V$. □

Proof of Theorem 2: The necessity is obvious since $(A \cup B \cup C) \supseteq (A \cup C)$. We show the sufficiency in a similar way to proof of Theorem 1. Let c and c' be two vertices in C that are d-separated by a subset of $A \cup B \cup C$. Thus from Lemma 2 they are also d-separated by $S = (an(c) \cup an(c')) \cap (A \cup B \cup C)$. Without loss of generality, suppose that c is not an ancestor of c'. Define $S_1 = (an(c) \cup an(c')) \cap (A \cup C)$ and $S_2 = (an(c) \cup an(c')) \cap (B \cup C)$. To prove that either S_1 $(\subseteq A \cup C)$ or S_2 $(\subseteq B \cup C)$ can d-separate c and c' in $\vec{G}_V$, it is sufficient to show that there will not exist a path l_1 in $A \cup C$ and a path l_2 in $B \cup C$ such that l_1 cannot be d-separated by S_1 and l_2 cannot be d-separated by S_2. To show this, we consider the following two cases separately:

(1) a path l_i is not completely contained in $An(c) \cup An(c')$, and

(2) both paths l_1 and l_2 are contained in $An(c) \cup An(c')$.

For case (1), since both S_1 and S_2 are subsets of $an(c) \cup an(c')$, we know from Lemma 4 that l must be d-separated both by S_1 and by S_2.

For case (2), by reduction to absurdity, we suppose that there are two paths l_1 and l_2 such that l_i cannot be d-separated by S_i for $i = 1$ and 2. Since every path l_i between c and c' is d-separated by S which equals $S_1 \cup S_2$, we have that for path l_i, there is at least one vertex contained in $S \setminus S_i$. Let d_1 and d_2 be such vertices that are closest to c on l_1 and l_2 respectively. We have $d_1 \in (S \setminus S_1)$ and thus $d_1 \in B$, and similarly $d_2 \in (S \setminus S_1)$ and thus $d_2 \in A$. Let l_1' denote the sub-path from c to d_1 of

l_1 and l_2' denote the sub-path from c to d_2 of l_2. Since l_i cannot be d-separated by S_i, we have that l_i' cannot be d-separated by S_i. Connecting l_1' and l_2' at c, we get a sequence l' from d_1 to d_2 through c. Note that l' may have the same vertices and thus it may not be a path. Below we show that l' is not d-separated by C, that is, the middle vertex of each collider or its descendant is in C but any of other vertices on l' is not in C.

For any vertex u which is not the middle vertex of a collider on l_1', since u is in $an(c) \cup an(c')$ and l_1 and l_1' is not d-separated by S_1, we have that $u \notin S_1$ and thus $u \notin C$. Similarly, we can show that C does not contain any vertex u which is not the middle vertex of a collider on l_2'. Thus we have shown that C does not contain any vertex which is not a middle vertex of colliders on l' except that vertex c has not yet been considered. Now we show that vertex c is a middle vertex of a collider on l'. Let v denote the neighbor of c on l_1'. Since v is in $an(c) \cup an(c')$ and it cannot be c', v is an ancestor of c or c'. If the edge between c and v is oriented as $c \to v$, then v must be an ancestor of c'. This contradicts the supposition that c is not an ancestor of c', and thus the edge between c and v must be oriented as $c \leftarrow v$. Similarly for the neighbor w of c on l_2', we can also show that the edge between c and w must be oriented as $c \leftarrow w$, which implies that the sequence (v, c, w) must form a collider on l'. Thus we have shown that C does not contain any vertex which is not a middle vertex of colliders on l'.

For any vertex u which is a middle vertex of a collider on l_i', u or its descendant must be in S_i, otherwise l_i' and so l_i are d-separated by S_i, which contradicts the supposition. Since u is contained in $an(c) \cup an(c')$, we have that c ($\in C$) or c' ($\in C$) is a descendant of u, and thus u or its descendant must be in C. For the collider $u \to c \leftarrow v$ on the sequence l', we also have that c is in C. Thus we have shown that the middle vertex of each collider on l' or its descendant is in C.

By the above result and Lemma 5, we have $d_2 \not\perp\!\!\!\perp d_1 | C$, where $d_2 \in A$ and $d_1 \in B$. This contradicts $A \perp\!\!\!\perp B | C$. Thus either S_1 or S_2 must d-separate c and c' in $\vec{G}_V$. □

Proof of Theorem 3: This is an immediate consequence of Theorems 1 and 2. □

Proof of Theorem 4: For necessity, since $A \perp\!\!\!\perp B | C$, we have from the property of conditional independence that $u \perp\!\!\!\perp B | A \cup C \setminus \{u\}$. This and the condition $u \perp\!\!\!\perp v | A \cup C \setminus \{u, v\}$ imply $u \perp\!\!\!\perp v \cup B | A \cup C \setminus \{u, v\}$. Again, from the property of conditional independence, we have $u \perp\!\!\!\perp v | A \cup B \cup C \setminus \{u, v\}$. For sufficiency, from $A \perp\!\!\!\perp B | C$, we get $u \perp\!\!\!\perp B | A \cup C \setminus \{u\}$. This and the condition $u \perp\!\!\!\perp v | A \cup B \cup C \setminus \{u, v\}$ imply $u \perp\!\!\!\perp B \cup \{v\} | A \cup C \setminus \{u, v\}$. Then we obtain $u \perp\!\!\!\perp v | A \cup C \setminus \{u, v\}$, and this completes our proof for the theorem. □

References

B. Abramson, J. Brown, A. Murphy, and R. L. Winkler. Hailfinder: A Bayesian system for forecasting severe weather. *International Journal of Forecasting*, 12:57-71, 1996.

C.F. Aliferis, I. Tsamardinos, and A. Statnikov. Causal Explorer: A probabilistic network learning toolkit for discovery. *The 2003 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, 2003.

S. Arnborg, D.G. Corneil, and A. Proskurowski. Complexity of finding embeddings in k-trees. *SIAM Journal of Algebraic and Discrete Methods*, 8(2):277-284, 1987.

A. Becker and D. Geiger, A sufficiently fast algorithm for finding close to optimal clique trees. *Artificial Intelligence*, 125:3-17, 2001.

I. Beinlich, H. Suermondt, R. Chavez, 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 Artificial Intelligence in Medicine*, pages 247-256, Springer-Verlag, Berlin, 1989.

J. Binder, D. Koller, S.J. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213-244, 1997.

R. Castelo and A. Roverato. A robust procedure for Gaussian graphical model search from Microarray data with p larger than n. *Journal of Machine Learning Research*, 7:2621-2650, 2006.

J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137(1):43-90, 2002.

D.M. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445-498, 2002.

D.M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287-1330, 2004.

G.F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309-348, 1992.

R.G. Cowell, A. P. David, S.L. Lauritzen, and D.J. Spiegelhalter. *Probabilistic Networks and Expert Systems*, Springer Publications, New York, 1999.

A.P. Dempster. Covariance selection. *Biometrics*, 28:157-175, 1972.

B. Engelhardt, M.I. Jordan, and S. Brenner. A statistical graphical model for predicting protein molecular function. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 297-304, Pittsburgh, Pennsylvania, 2006.

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 Artificial Intelligence*, pages 206-215, Stockholm, Sweden, 1999.

N. Friedman and D. Koller. Being Bayesian about Bayesian Network structure: A Bayesian approach to structure discovery in Bayesian Networks. *Machine Learning*, 50:95-125, 2003.

Z. Geng, C. Wang, and Q. Zhao. Decomposition of search for *v*-structures in DAGs. *Journal of Multivariate Analysis*, 96(2):282-294, 2005.

D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197-243, 1995.

D. Heckerman. A tutorial on learning with Bayesian networks. Learning in graphical models, pages 301-354, M. Jordan (Ed.), Kluwer Academic Pub., Netherlands, 1998.

F.V. Jensen and F. Jensen. Optimal junction trees. In *Proccedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 360-366, San Fransisco, CA, 1994.

M. I. Jordan. Graphical models. *Statistical Science, (Special Issue on Bayesian Statistics)*, 19:140-155, 2004.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8:613-636, 2007.

D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 284-292, Bari, Italy, 1996.

S. Kullback and R.A. Leibler. On information and sufficiency, *Annals of Mathematical Statistics*, 22:79-86, 1951.

S.L. Lauritzen. *Graphical Models*, Clarendon Press, Oxford, 1996.

D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In *Proceedings of the Twelfth Advances in Neural Information Processing Systems*, Denver, Colorado, 505-511, 1999.

C. Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 403-410, Montreal, Quebec, 1995.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34: 1436-1462, 2006.

K. Murphy. The Bayes net toolbox for Matlab. *Computing Science and Statistics*, 33:331-350, 2001.

M. Narasimhan and J. Bilmes. Optimal Sub-graphical Models. In *Advances in Neural Information Processing Systems*, vol. 17, pages 961-968, L. Saul and Y. Weiss and Léon Bottou (Ed.), MIT Press, Cambridge, 2005.

J. Pearl. *Causality*, Cambridge University Press, Cambridge, 2000.

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

D. Rose, R. Tarjan, and G. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5:266-283, 1976.

M. Schmidt, A. Niculescu-Mizil, and K. Murphy. Learning graphical model structure using L1-Regularization paths. In *Proceedings of the 22nd Conference on Artificial Intelligence*, pages 1278-1283, Vancouver, British Columbia, 2007.

P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9:62-72, 1991.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*, MIT Press, Cambridge, 2000.

R. E. Tarjan and M. Yannakakis. Simple linear-time algorithm to test chordality of graphs, test acyclicity of hypergraphs, and selective reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13:566-579, 1984.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B.* 58(1):267-288, 1996.

I. Tsamardinos, C.F. Aliferis, and A. Statnikov. Algorithms for large scale Markov blanket discovery. In *Proceedings of the 16th International FLAIRS Conference*, pages 592-597, 2003.

I. Tsamardinos, L. Brown, and C. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31-78, 2006.

T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, pages 255-268, Cambridge, MA, 1990.

M. J. Wainwright, P. Ravikumar, and J.D. Lafferty. High-dimensional graphical model selection using L1-regularized logistic regression. In *Proceedings of Twentieth Advances in Neural Information Processing Systems*, pages 1465-1472, Vancouver, 2006.

J. Whittaker. *Graphical Models in Applied Multivariate Statistics*, John Wiley & Sons, New York, 1990.

S.S. Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *Annals of Mathematical Statistics*, 20:595-601, 1938.

A. Wille and P. Bühlmann. Low-order conditional independence graphs for inferring genetic networks. *Statistical Applications in Genetics and Molecular Biology*, 5(1):1-32, 2006.

X. Xie, Z. Geng, and Q. Zhao. Decomposition of structural learning about directed acyclic graphs. *Artificial Intelligence*, 170:422-439, 2006.

P. Zhao and B. Yu. On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7:2541-2563, 2006.

Journal of Machine Learning Research 9 (2008) 485-516 Submitted 5/07; Revised 12/07; Published 3/08

Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data

Onureena Banerjee ONUREENA@EECS.BERKELEY.EDU
Laurent El Ghaoui ELGHAOUI@EECS.BERKELEY.EDU
EECS Department
University of California, Berkeley
Berkeley, CA 94720 USA

Alexandre d'Aspremont ASPREMON@PRINCETON.EDU
ORFE Department
Princeton University
Princeton, NJ 08544 USA

Editor: John Lafferty

Abstract

We consider the problem of estimating the parameters of a Gaussian or binary distribution in such a way that the resulting undirected graphical model is sparse. Our approach is to solve a maximum likelihood problem with an added ℓ_1-norm penalty term. The problem as formulated is convex but the memory requirements and complexity of existing interior point methods are prohibitive for problems with more than tens of nodes. We present two new algorithms for solving problems with at least a thousand nodes in the Gaussian case. Our first algorithm uses block coordinate descent, and can be interpreted as recursive ℓ_1-norm penalized regression. Our second algorithm, based on Nesterov's first order method, yields a complexity estimate with a better dependence on problem size than existing interior point methods. Using a log determinant relaxation of the log partition function (Wainwright and Jordan, 2006), we show that these same algorithms can be used to solve an approximate sparse maximum likelihood problem for the binary case. We test our algorithms on synthetic data, as well as on gene expression and senate voting records data.

Keywords: model selection, maximum likelihood estimation, convex optimization, Gaussian graphical model, binary data

1. Introduction

Undirected graphical models offer a way to describe and explain the relationships among a set of variables, a central element of multivariate data analysis. The principle of parsimony dictates that we should select the simplest graphical model that adequately explains the data. In this paper we consider practical ways of implementing the following approach to finding such a model: given a set of data, we solve a maximum likelihood problem with an added ℓ_1-norm penalty to make the resulting graph as sparse as possible.

Many authors have studied a variety of related ideas. In the Gaussian case, model selection involves finding the pattern of zeros in the inverse covariance matrix, since these zeros correspond to conditional independencies among the variables. Traditionally, a greedy forward-backward search algorithm is used to determine the zero pattern (e.g., Lauritzen, 1996). However, this is computationally infeasible for data with even a moderate number of variables. Li and Gui (2005) introduce

a gradient descent algorithm in which they account for the sparsity of the inverse covariance matrix by defining a loss function that is the negative of the log likelihood function. Speed and Kiiveri (1986) and, more recently, Dahl et al. (Revised 2007) proposed a set of large scale methods for problems where a sparsity pattern for the inverse covariance is given and one must estimate the nonzero elements of the matrix.

Another way to estimate the graphical model is to find the set of neighbors of each node in the graph by regressing that variable against the remaining variables. In this vein, Dobra and West (2004) employ a stochastic algorithm to manage tens of thousands of variables. There has also been a great deal of interest in using ℓ_1-norm penalties in statistical applications. d'Aspremont et al. (2004) apply an ℓ_1 norm penalty to sparse principle component analysis. Directly related to our problem is the use of the Lasso of Tibshirani (1996) to obtain a very short list of neighbors for each node in the graph. Meinshausen and Bühlmann (2006) study this approach in detail, and show that the resulting estimator is consistent, even for high-dimensional graphs. A related approach is the graphical lasso, explored by Friedman et al. (2007).

Our objective here is to both estimate the sparsity pattern of the underlying graph and to obtain a regularized estimate of the covariance matrix. The difficulty of the problem formulation we consider lies in its computation. Although the problem is convex, it is non-smooth and has an unbounded constraint set. As we shall see, the resulting complexity for existing interior point methods is $O(p^6)$, where p is the number of variables in the distribution. In addition, interior point methods require that at each step we compute and store a Hessian of size $O(p^2)$. The memory requirements and complexity are thus prohibitive for $O(p)$ higher than the tens. Specialized algorithms are needed to handle larger problems.

The remainder of the paper is organized as follows. We begin by considering Gaussian data. In Section 2 we set up the problem, derive its dual, discuss properties of the solution and how heavily to weight the ℓ_1-norm penalty in our problem. In Section 3 we present a provably convergent block coordinate descent algorithm that can be interpreted a sequence of iterative ℓ_1-norm penalized regressions. In Section 4 we present a second, alternative algorithm based on Nesterov's recent work on non-smooth optimization, and give a rigorous complexity analysis with better dependence on problem size than interior point methods. In Section 5 we show that the algorithms we developed for the Gaussian case can also be used to solve an approximate sparse maximum likelihood problem for multivariate binary data, using a log determinant relaxation for the log partition function given by Wainwright and Jordan (2006). In Section 6, we test our methods on synthetic as well as gene expression and senate voting records data.

2. Problem Formulation

In this section we set up the sparse maximum likelihood problem for Gaussian data, derive its dual, and discuss some of its properties.

2.1 Problem Setup

Suppose we are given n samples independently drawn from a p-variate Gaussian distribution: $y^{(1)},\ldots,y^{(n)} \sim \mathcal{N}(\mu,\Sigma_p)$, where the covariance matrix Σ is to be estimated. Let S denote the second moment matrix about the mean:

$$S := \frac{1}{n}\sum_{k=1}^{n}(y^{(k)}-\mu)(y^{(k)}-\mu)^T.$$

Let $\hat{\Sigma}^{-1}$ denote our estimate of the inverse covariance matrix. Our estimator takes the form:

$$\hat{\Sigma}^{-1} = \arg\max_{X\succ 0} \log\det X - \mathrm{trace}(SX) - \lambda\|X\|_1. \tag{1}$$

Here, $\|X\|_1$ denotes the sum of the absolute values of the elements of the positive definite matrix X.

The scalar parameter λ controls the size of the penalty. The penalty term is a proxy for the number of nonzero elements in X, and is often used—albeit with vector, not matrix, variables—in regression techniques, such as the Lasso.

In the case where $S \succ 0$, the classical maximum likelihood estimate is recovered for $\lambda = 0$. However, when the number of samples n is small compared to the number of variables p, the second moment matrix may not be invertible. In such cases, for $\lambda > 0$, our estimator performs some regularization so that our estimate $\hat{\Sigma}$ is always invertible, no matter how small the ratio of samples to variables is.

Even in cases where we have enough samples so that $S \succ 0$, the inverse S^{-1} may not be sparse, even if there are many conditional independencies among the variables in the distribution. By trading off maximality of the log likelihood for sparsity, we hope to find a very sparse solution that still adequately explains the data. A larger value of λ corresponds to a sparser solution that fits the data less well. A smaller λ corresponds to a solution that fits the data well but is less sparse. The choice of λ is therefore an important issue that will be examined in detail in Section 2.3.

2.2 The Dual Problem and Bounds on the Solution

By expressing the ℓ_1 norm in (1) as

$$\|X\|_1 = \max_{\|U\|_\infty \le 1} \mathrm{trace}(XU),$$

where $\|U\|_\infty$ denotes the maximum absolute value element of the symmetric matrix U, we can write (1) as

$$\max_{X\succ 0}\min_{\|U\|_\infty\le\lambda} \log\det X - \mathrm{trace}(X, S+U).$$

This corresponds to seeking an estimate with the maximum worst-case log likelihood, over all additive perturbations of the second moment matrix S. A similar robustness interpretation can be made for a number of estimation problems, such as support vector machines for classification.

We can obtain the dual problem by exchanging the max and the min. The resulting inner problem in X can be solved analytically by setting the gradient of the objective to zero and solving for X. The result is

$$\min_{\|U\|_\infty\le\lambda} -\log\det(S+U) - p$$

where the primal and dual variables are related as: $X = (S+U)^{-1}$. Note that the log determinant function acts a log barrier, creating an implicit constraint that $S+U \succ 0$.

To write things neatly, let $W = S+U$. Then the dual of our sparse maximum likelihood problem is

$$\hat{\Sigma} := \max\{\log\det W : \|W-S\|_\infty \le \lambda\}. \tag{2}$$

Observe that the dual problem (2) estimates the covariance matrix while the primal problem estimates its inverse. We also observe that the diagonal elements of the solution are $\Sigma_{kk} = S_{kk} + \lambda$ for all k.

The following theorem shows that adding the ℓ_1-norm penalty regularizes the solution.

Theorem 1 *For every $\lambda > 0$, the optimal solution to (1) is unique, with bounded eigenvalues:*

$$\frac{p}{\lambda} \geq \|\hat{\Sigma}^{-1}\|_2 \geq (\|S\|_2 + \lambda p)^{-1}.$$

Here, $\|A\|_2$ denotes the maximum eigenvalue of a symmetric matrix A. The proof is contained in the appendix.

The dual problem (2) is smooth and convex, albeit with $p(p+1)/2$ variables. For small values of p, the problem can be solved by existing interior point methods (e.g., Vandenberghe et al., 1998). The complexity to compute an ε-suboptimal solution using such second-order methods, however, is $O(p^6 \log(1/\varepsilon))$, making them infeasible when p is larger than the tens.

A related problem, solved by Dahl et al. (Revised 2007), is to compute a maximum likelihood estimate of the covariance matrix when the sparsity structure of the inverse is known in advance. This is accomplished by adding constraints to (1) of the form: $X_{ij} = 0$ for all pairs (i, j) in some specified set. Our constraint set is unbounded as we hope to uncover the sparsity structure automatically, starting with a dense second moment matrix S.

2.3 Choice of Penalty Parameter

Consider the true, unknown graphical model for a given distribution. This graph has p nodes, and an edge between nodes k and j is missing if variables k and j are independent conditional on the rest of the variables. For a given node k, let C_k denote its connectivity component: the set of all nodes that are connected to node k through some chain of edges. In particular, if node $j \notin C_k$, then variables j and k are independent.

Meinshausen and Bühlmann (2006) show that the Lasso, when used to estimate the set of neighbors for each node, yields an estimate of the sparsity pattern of the graph in a way that is asymptotically consistent. Consistency, they show, hinges on the choice of the regularization parameter. In addition to asymptotic results, they prove a finite sample result: by choosing the regularization parameter in a particular way, the probability that two distinct connectivity components are falsely joined in the estimated graph can be controlled.

Following the methods used by Meinshausen and Bühlmann (2006), we can derive a corresponding choice for the regularization parameter. Let $\hat{C}_k^{\lambda}$ denote our estimate of the connectivity component of node k. In the context of our optimization problem, this corresponds to the entries of row k in $\hat{\Sigma}$ that are nonzero.

Let α be a given level in $[0, 1]$. Consider the following choice for the penalty parameter in (1):

$$\lambda(\alpha) := (\max_{i>j} \hat{\sigma}_i \hat{\sigma}_j) \frac{t_{n-2}(\alpha/2p^2)}{\sqrt{n-2+t_{n-2}^2(\alpha/2p^2)}} \tag{3}$$

where $t_{n-2}(\alpha)$ denotes the $(100-\alpha)\%$ point of the Student's t-distribution for $n-2$ degrees of freedom, and $\hat{\sigma}_i$ is the empirical variance of variable i. Then we can prove the following theorem:

Theorem 2 *Using $\lambda(\alpha)$ the penalty parameter in (1), for any fixed level α,*

$$P(\exists k \in \{1,\ldots,p\} : \hat{C}_k^{\lambda} \not\subseteq C_k) \leq \alpha.$$

Observe that, for a fixed problem size p, as the number of samples n increases to infinity, the penalty parameter $\lambda(\alpha)$ decreases to zero. Thus, asymptotically we recover the classical maximum likelihood estimate, S, which in turn converges in probability to the true covariance Σ.

Meinshausen and Bühlmann (2006) prove that Lasso recovers the underlying sparsity pattern consistently, in a scheme where the number of variables is allowed to grow with the number of samples: $p \in O(n^{\gamma})$. Although such an analysis is beyond the scope of this paper, a similar result seems to hold here provided we assume certain conditions on the true covariance matrix.

From the point of view of estimating the covariance matrix itself, when the number of variables is allowed to grow with the number of samples, it may be better to use estimates such as those described by Bickel and Levina (2008), which are shown to be consistent in the operator norm as long as $(\log p)^2/n \to 0$.

3. Block Coordinate Descent Algorithm

In this section we present an algorithm for solving (2) that uses block coordinate descent.

3.1 Algorithm Description

We begin by detailing the algorithm. For any symmetric matrix A, let $A_{\backslash k \backslash j}$ denote the matrix produced by removing column k and row j. Let A_j denote column j with the diagonal element A_{jj} removed. The plan is to optimize over one row and column of the variable matrix W at a time, and to repeatedly sweep through all columns until we achieve convergence.
Initialize: $W^{(0)} := S + \lambda I$
For $k \geq 0$

1. **For** $j = 1,\ldots,p$

   (a) Let $W^{(j-1)}$ denote the current iterate. Solve the quadratic program

   $$\hat{y} := \arg\min_y \{y^T (W^{(j-1)}_{\backslash j \backslash j})^{-1} y : \|y - S_j\|_\infty \leq \lambda\}. \tag{4}$$

   (b) Update rule: $W^{(j)}$ is $W^{(j-1)}$ with column/row W_j replaced by $\hat{y}$.

2. Let $\hat{W}^{(0)} := W^{(p)}$.

3. After each sweep through all columns, check the convergence condition. Convergence occurs when

   $$\text{trace}((\hat{W}^{(0)})^{-1} S) - p + \lambda \|(\hat{W}^{(0)})^{-1}\|_1 \leq \varepsilon.$$

3.2 Convergence and Property of Solution

Using Schur complements, we can prove convergence:

Theorem 3 *The block coordinate descent algorithm described above converges, achieving an* ε*-suboptimal solution to (2). In particular, the iterates produced by the algorithm are strictly positive definite: each time we sweep through the columns,* $W^{(j)} \succ 0$ *for all* j*.*

The proof of Theorem 3 sheds some interesting light on the solution to problem (1). In particular, we can use this method to show that the solution has the following property:

Theorem 4 *Fix any* $k \in \{1,\ldots,p\}$*. If* $\lambda \geq |S_{kj}|$ *for all* $j \neq k$*, then column and row* k *of the solution* $\hat{\Sigma}$ *to (2) are zero, excluding the diagonal element.*

This means that, for a given second moment matrix S, if λ is chosen such that the condition in Theorem 4 is met for some column k, then the sparse maximum likelihood method estimates variable k to be independent of all other variables in the distribution. In particular, Theorem 4 implies that if $\lambda \geq |S_{kj}|$ for all $k > j$, then (1) estimates all variables in the distribution to be pairwise independent.

Using the work of Luo and Tseng (1992), it may be possible to show that the local convergence rate of this method is at least linear. In practice we have found that a small number of sweeps through all columns, independent of problem size p, is sufficient to achieve convergence. In each iteration, we must solve one quadratic program (QP). In a standard primal-dual interior point method for solving QPs, the computational cost is dominated by the cost of finding a search direction, which involves inverting matrices of size p. The cost of each iteration is therefore $O(p^3)$. For a fixed number of K sweeps, the cost of the method is $O(Kp^4)$.

3.3 Interpretation as Iterative Penalized Regression

The dual of (4) is

$$\min_x x^T W_{\setminus j \setminus j}^{(j-1)} x - S_j^T x + \lambda \|x\|_1. \tag{5}$$

Strong duality obtains so that problems (5) and (4) are equivalent. If we let Q denote the square root of $W_{\setminus j \setminus j}^{(j-1)}$, and $b := \frac{1}{2} Q^{-1} S_j$, then we can write (5) as

$$\min_x \|Qx - b\|_2^2 + \lambda \|x\|_1. \tag{6}$$

The problem (6) is a penalized least-squares problems, known as the Lasso. If $W_{\setminus j \setminus j}^{(j-1)}$ were the j-th principal minor of the sample covariance S, then (6) would be equivalent to a penalized regression of variable j against all others. Thus, the approach is reminiscent of the approach explored by Meinshausen and Bühlmann (2006), but there are two differences. First, we begin by adding a multiple of the identity to the sample covariance matrix, so we can guarantee that each penalized regression problem has a unique solution, without requiring the data or solutions to satisfy any special conditions (see Osborne et al., 2000, for conditions that guarantee uniqueness of the Lasso). Second, and more importantly, we update the problem data after each regression: except at the very first update, $W_{\setminus j \setminus j}^{(j-1)}$ is never a minor of S. In this sense, the coordinate descent method can be interpreted as a sequence of iterative Lasso problems. The approach is pursued to great advantage by Friedman et al. (2007), who also provide a conceptual link to the approach of Meinshausen and Bühlmann (2006).

4. Nesterov's First Order Method

In this section we apply the recent results due to Nesterov (2005) to obtain a first order algorithm for solving (1) with lower memory requirements and a rigorous complexity estimate with a better dependence on problem size than those offered by interior point methods. Our purpose is not to obtain another algorithm, as we have found that the block coordinate descent is fairly efficient; rather, we seek to use Nesterov's formalism to derive a rigorous complexity estimate for the problem, improved over that offered by interior-point methods. In particular, although we have found in practice that a small, fixed number K of sweeps through all columns is sufficient to achieve convergence using the block coordinate descent method, we have not been able to compute a bound on K. In what follows, we compute a guaranteed theoretical upper bound on the complexity of solving (1).

As we will see, Nesterov's framework allows us to obtain an algorithm that has a complexity of $O(p^{4.5}/\varepsilon)$, where $\varepsilon > 0$ is the desired accuracy on the objective of problem (1). This is in contrast to the complexity of interior-point methods, $O(p^6 \log(1/\varepsilon))$. Thus, Nesterov's method provides a much better dependence on problem size and lower memory requirements at the expense of a degraded dependence on accuracy.

4.1 Idea of Nesterov's Method

Nesterov's method applies to a class of non-smooth, convex optimization problems of the form

$$\min_x \{ f(x) : x \in Q_1 \} \tag{7}$$

where the objective function can be written as

$$f(x) = \hat{f}(x) + \max_u \{ \langle Ax, u \rangle_2 : u \in Q_2 \}.$$

Here, Q_1 and Q_2 are bounded, closed, convex sets, $\hat{f}(x)$ is differentiable (with a Lipschitz-continuous gradient) and convex on Q_1, and A is a linear operator. The challenge is to write our problem in the appropriate form and choose associated functions and parameters in such a way as to obtain the best possible complexity estimate, by applying general results obtained by Nesterov (2005).

Observe that we can write (1) in the form (7) if we impose bounds on the eigenvalues of the solution, X. To this end, we let

$$Q_1 := \{x : aI \preceq X \preceq bI\},$$
$$Q_2 := \{u : \|u\|_\infty \leq \lambda\}$$

where the constants a, b are given such that $b > a > 0$. By Theorem 1, we know that such bounds always exist. We also define $\hat{f}(x) := -\log\det x + \langle S, x \rangle$, and $A := I$.

To Q_1 and Q_2, we associate norms and continuous, strongly convex functions, called prox-functions, $d_1(x)$ and $d_2(u)$. For Q_1 we choose the Frobenius norm, and a prox-function $d_1(x) = -\log\det x + p\log b$. For Q_2, we choose the Frobenius norm again, and a prox-function $d_2(x) = \|u\|_F^2/2$.

The method applies a smoothing technique to the non-smooth problem (7), which replaces the objective of the original problem, $f(x)$, by a penalized function involving the prox-function $d_2(u)$:

$$\tilde{f}(x) = \hat{f}(x) + \max_{u \in Q_2} \{ \langle Ax, u \rangle - \mu d_2(u) \}. \tag{8}$$

The above function turns out to be a smooth uniform approximation to f everywhere. It is differentiable, convex on Q_1, and a has a Lipschitz-continuous gradient, with a constant L that can be computed as detailed below. A specific gradient scheme is then applied to this smooth approximation, with convergence rate $O(L/\varepsilon)$.

4.2 Algorithm and Complexity Estimate

To detail the algorithm and compute the complexity, we must first calculate some parameters corresponding to our definitions and choices above. First, the strong convexity parameter for $d_1(x)$ on Q_1 is $\sigma_1 = 1/b^2$, in the sense that

$$\nabla^2 d_1(X)[H,H] = \mathrm{trace}(X^{-1}HX^{-1}H) \geq b^{-2}\|H\|_F^2$$

for every symmetric H. Furthermore, the center of the set Q_1 is $x_0 := \arg\min_{x\in Q_1} d_1(x) = bI$, and satisfies $d_1(x_0) = 0$. With our choice, we have $D_1 := \max_{x\in Q_1} d_1(x) = p\log(b/a)$.

Similarly, the strong convexity parameter for $d_2(u)$ on Q_2 is $\sigma_2 := 1$, and we have

$$D_2 := \max_{u\in Q_2} d_2(U) = p^2/2.$$

With this choice, the center of the set Q_2 is $u_0 := \arg\min_{u\in Q_2} d_2(u) = 0$.

For a desired accuracy ε, we set the smoothness parameter $\mu := \varepsilon/2D_2$, and set $x_0 = bI$. The algorithm proceeds as follows:

For $k \geq 0$ **do**

1. Compute $\nabla \tilde{f}(x_k) = -x^{-1} + S + u^*(x_k)$, where $u^*(x)$ solves (8).
2. Find $y_k = \arg\min_y \{\langle \nabla \tilde{f}(x_k), y - x_k\rangle + \frac{1}{2}L(\varepsilon)\|y - x_k\|_F^2 \ : \ y \in Q_1\}$.
3. Find $z_k = \arg\min_x \{\frac{L(\varepsilon)}{\sigma_1} d_1(X) + \sum_{i=0}^k \frac{i+1}{2}\langle \nabla \tilde{f}(x_i), x - x_i\rangle \ : \ x \in Q_1\}$.
4. Update $x_k = \frac{2}{k+3}z_k + \frac{k+1}{k+3}y_k$.

In our case, the Lipschitz constant for the gradient of our smooth approximation to the objective function is

$$L(\varepsilon) := M + D_2\|A\|^2/(2\sigma_2\varepsilon)$$

where $M := 1/a^2$ is the Lipschitz constant for the gradient of $\tilde{f}$, and the norm $\|A\|$ is induced by the Frobenius norm, and is equal to λ.

The algorithm is guaranteed to produce an ε-suboptimal solution after a number of steps not exceeding

$$\begin{aligned} &N(\varepsilon) := \\ &4\|A\|\sqrt{\frac{D_1 D_2}{\sigma_1\sigma_2}} \cdot \tfrac{1}{\varepsilon} + \sqrt{\tfrac{MD_1}{\sigma_1\varepsilon}} \\ &= (\kappa\sqrt{(\log\kappa)})(4p^{1.5}a\lambda/\sqrt{2} + \sqrt{\varepsilon p})/\varepsilon. \end{aligned} \tag{9}$$

where $\kappa = b/a$ is a bound on the condition number of the solution.

Now we are ready to estimate the complexity of the algorithm. For Step 1, the gradient of the smooth approximation is computed in closed form by taking the inverse of x. Step 2 essentially

amounts to projecting on Q_1, and requires that we solve an eigenvalue problem. The same is true for Step 3. In fact, each iteration costs $O(p^3)$. The number of iterations necessary to achieve an objective with absolute accuracy less than ε is given in (9) by $N(\varepsilon) = O(p^{1.5}/\varepsilon)$. Thus, if the condition number κ is fixed in advance, the complexity of the algorithm is $O(p^{4.5}/\varepsilon)$.

5. Binary Variables: Approximate Sparse Maximum Likelihood Estimation

In this section, we consider the problem of estimating an undirected graphical model for multivariate binary data. Recently, Wainwright et al. (2006) applied an ℓ_1-norm penalty to the logistic regression problem to obtain a binary version of the high-dimensional consistency results of Meinshausen and Bühlmann (2006). We apply the log determinant relaxation of Wainwright and Jordan (2006) to formulate an approximate sparse maximum likelihood (ASML) problem for estimating the parameters in a multivariate binary distribution. We show that the resulting problem is the same as the Gaussian sparse maximum likelihood (SML) problem, and that we can therefore apply our previously-developed algorithms to sparse model selection in a binary setting.

Consider a distribution made up of p binary random variables. Using n data samples, we wish to estimate the structure of the distribution. An Ising model for this distribution is

$$p(x;\theta) = \exp\{\sum_{i=1}^{p} \theta_i x_i + \sum_{i=1}^{p-1} \sum_{j=i+1}^{p} \theta_{ij} x_i x_j - A(\theta)\} \tag{10}$$

where

$$A(\theta) = \log \sum_{x \in \mathcal{X}^p} \exp\{\sum_{i=1}^{p} \theta_i x_i + \sum_{i=1}^{p-1} \sum_{j=i+1}^{p} \theta_{ij} x_i x_j\} \tag{11}$$

is the log partition function.

The sparse maximum likelihood problem in this case is to maximize (10) with an added ℓ_1-norm penalty on terms θ_{kj}. Specifically, in the undirected graphical model, an edge between nodes k and j is missing if $\theta_{kj} = 0$.

A well-known difficulty is that the log partition function has too many terms in its outer sum to compute. However, if we use the log determinant relaxation for the log partition function developed by Wainwright and Jordan (2006), we can obtain an approximate sparse maximum likelihood (ASML) estimate.

In their paper, Wainwright and Jordan (2006) consider the problem of computing marginal probabilities over subsets of nodes in a graphical model for a discrete-valued Markov random field. This problem is closely related to the one we examine here, and exactly solving the problem for general graphs is intractable. Wainwright and Jordan (2006) propose a relaxation by using a Gaussian bound on the log partition function (11) and a semidefinite outer bound on the polytope of marginal probabilities. In the next section, we use their results to obtain an approximate estimate of undirected graphical model for binary variables, and we show that, when using the simplest semidefinite outer bound on the constraint set, we obtain a problem that is almost exactly the same as that considered in previous sections, for the Gaussian case. In particular, this means that we can reuse the block coordinate descent or Nesterov algorithms to approximately estimate graphical models in the case of binary data.

5.1 Problem Formulation

Let's begin with some notation. Letting $d := p(p+1)/2$, define the map $R : \mathbf{R}^d \to S^{p+1}$ as follows:

$$R(\theta) = \begin{pmatrix} 0 & \theta_1 & \theta_2 & \dots & \theta_p \\ \theta_1 & 0 & \theta_{12} & \dots & \theta_{1p} \\ \vdots & & & & \\ \theta_p & \theta_{1p} & \theta_{2p} & \dots & 0 \end{pmatrix}$$

Suppose that our n samples are $z^{(1)}, \dots, z^{(n)} \in \{-1, +1\}^p$. Let $\overline{z_i}$ and $\overline{z_{ij}}$ denote sample mean and second moments. The sparse maximum likelihood problem is

$$\hat{\theta}_{\text{exact}} := \arg\max_{\theta} \frac{1}{2}\langle R(\theta), R(\overline{z})\rangle - A(\theta) - \lambda\|\theta\|_1. \tag{12}$$

Finally define the constant vector $m = (1, \frac{4}{3}, \dots, \frac{4}{3}) \in \mathbf{R}^{p+1}$. Wainwright and Jordan (2006) give an upper bound on the log partition function as the solution to the following variational problem:

$$\begin{array}{l} A(\theta) \leq \max_{\mu} \frac{1}{2}\log\det(R(\mu) + \operatorname{diag}(m)) + \langle\theta, \mu\rangle \\ = \frac{1}{2} \cdot \max_{\mu} \log\det(R(\mu) + \operatorname{diag}(m)) + \langle R(\theta), R(\mu)\rangle. \end{array} \tag{13}$$

If we use the bound (13) in our sparse maximum likelihood problem (12), we won't be able to extract an optimizing argument $\hat{\theta}$. Our first step, therefore, will be to rewrite the bound in a form that will allow this.

Lemma 5 *We can rewrite the bound (13) as*

$$A(\theta) \leq \frac{p}{2}\log(\frac{e\pi}{2}) - \frac{1}{2}(p+1) - \frac{1}{2} \cdot \{\max_{\nu} \nu^T m + \log\det(-(R(\theta) + diag(\nu))). \tag{14}$$

Using this version of the bound (13), we have the following theorem.

Theorem 6 *Using the upper bound on the log partition function given in (14), the approximate sparse maximum likelihood problem has the following solution:*

$$\begin{array}{l} \hat{\theta}_k = \overline{\mu_k} \\ \hat{\theta}_{kj} = -(\hat{\Gamma})^{-1}_{kj} \end{array} \tag{15}$$

where the matrix $\hat{\Gamma}$ is the solution to the following problem, related to (2):

$$\hat{\Gamma} := \arg\max\{\log\det W : W_{kk} = S_{kk} + \frac{1}{3},\ |W_{kj} - S_{kj}| \leq \lambda\}. \tag{16}$$

Here, S is defined as before:

$$S = \frac{1}{n}\sum_{k=1}^{n} (z^{(k)} - \overline{\mu})(z^{(k)} - \overline{\mu})^T$$

where $\overline{\mu}$ is the vector of sample means $\overline{z_i}$.

In particular, this means that we can reuse the algorithms developed in Sections 3 and 4 for problems with binary variables. The relaxation (13) is the simplest one offered by Wainwright and Jordan (2006). The relaxation can be tightened by adding linear constraints on the variable μ.

5.2 Penalty Parameter Choice for Binary Variables

For the choice of the penalty parameter λ, we can derive a formula analogous to (3). Consider the choice

$$\lambda(\alpha)_{\text{bin}} := \frac{(\chi^2(\alpha/2p^2,1))^{\frac{1}{2}}}{(\min_{i>j}\hat{\sigma}_i\hat{\sigma}_j)\sqrt{n}} \tag{17}$$

where $\chi^2(\alpha,1)$ is the $(100-\alpha)\%$ point of the chi-square distribution for one degree of freedom. Since our variables take on values in $\{-1,1\}$, the empirical variances are of the form:

$$\hat{\sigma}_i^2 = 1-\bar{\mu}_i^2.$$

Using (17), we have the following binary version of Theorem 2:

Theorem 7 *With (17) chosen as the penalty parameter in the approximate sparse maximum likelihood problem, for a fixed level α,*

$$P(\exists k \in \{1,\ldots,p\} : \hat{C}_k^{\lambda} \not\subseteq C_k) \leq \alpha.$$

6. Numerical Results

In this section we present the results of some numerical experiments, both on synthetic and real data. The synthetic experiments were performed on continuous data, and tests only the original problem formulation. Both continuous and discrete real data sets are used, however, testing the Gaussian and binary formulations respectively.

6.1 Synthetic Experiments

Synthetic experiments require that we generate underlying sparse inverse covariance matrices. To this end, we first randomly choose a diagonal matrix with positive diagonal entries. A given number of nonzeros are inserted in the matrix at random locations symmetrically. Positive definiteness is ensured by adding a multiple of the identity to the matrix if needed. The multiple is chosen to be only as large as necessary for inversion with no errors.

6.2 Sparsity and Thresholding

A very simple approach to obtaining a sparse estimate of the inverse covariance matrix would be to apply a threshold to the inverse empirical covariance matrix, S^{-1}. However, even when S is easily invertible, it can be difficult to select a threshold level. We solved a synthetic problem of size $p = 100$ where the true concentration matrix density was set to $\delta = 0.1$. Drawing $n = 200$ samples, we plot in Figure 1 the sorted absolute value elements of S^{-1} on the left and $\hat{\Sigma}^{-1}$, the solution to (1), on the right.

It is clearly easier to choose a threshold level for the sparse maximum likelihood estimate. Applying a threshold to either S^{-1} or $\hat{\Sigma}^{-1}$ would decrease the log likelihood of the estimate by an unknown amount. One way to compute the largest possible threshold t that preserves positive definiteness in S can be computed by solving the minimization problem:

$$t \leq \min_{\|v\|_1=1} v^T S^{-1} v.$$

The condition (6.2) is equivalent to the condition that $S^{-1}+L \succ 0$ for all matrices L such that $\|L\|_\infty \leq t$.

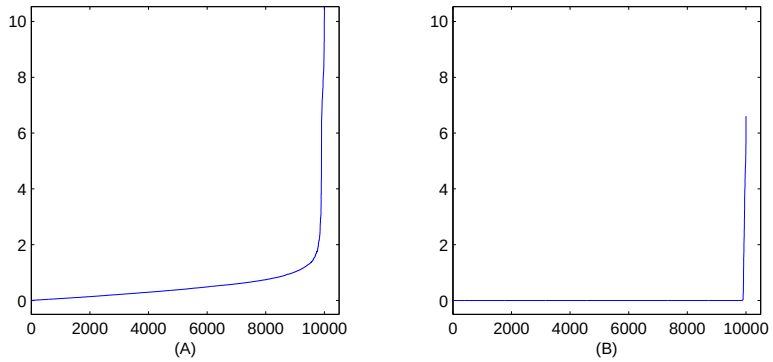


Figure 1: Sorted absolute value of elements of (A) S^{-1} and (B) $\hat{\Sigma}^{-1}$. The solution $\hat{\Sigma}^{-1}$ to (1) is un-thresholded.

6.3 Recovering Structure

We begin with a small experiment to test the ability of the method to recover the sparse structure of an underlying covariance matrix. Figure 2 (A) shows a sparse inverse covariance matrix of size $p = 30$. Figure 2 (B) displays a corresponding S^{-1}, using $n = 60$ samples. Figure 2 (C) displays the solution to (1) for $\lambda = 0.1$. The value of the penalty parameter here is chosen arbitrarily, and the solution is not thresholded. Nevertheless, we can still pick out features that were present in the true underlying inverse covariance matrix.

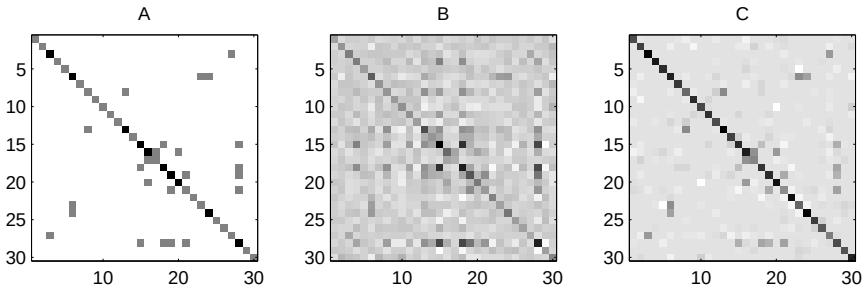


Figure 2: Recovering the sparsity pattern. We plot (A) the original inverse covariance matrix Σ^{-1}, (B) the noisy sample inverse S^{-1}, and (C) the solution to problem (1) for $\lambda = 0.1$.

Using the same underlying inverse covariance matrix, we repeat the experiment using smaller sample sizes. We solve (1) for $n = 30$ and $n = 20$ using the same arbitrarily chosen penalty parameter value $\lambda = 0.1$, and display the solutions in Figure 3. As expected, our ability to pick out features of the true inverse covariance matrix diminishes with the number of samples. This is an added reason to choose a larger value of λ when we have fewer samples, as in (3).

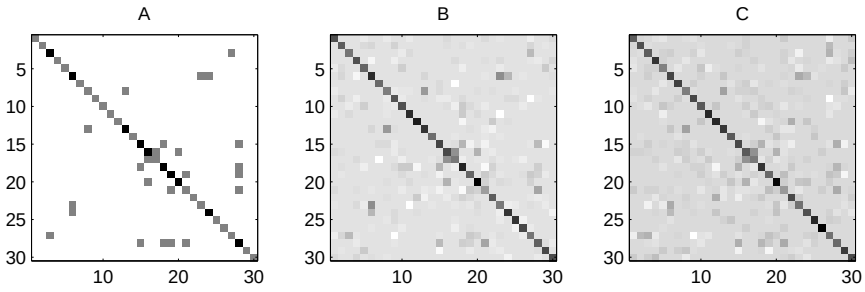


Figure 3: Recovering the sparsity pattern for small sample size. We plot (A) the original inverse covariance matrix Σ^{-1}, (B) the solution to problem (1) for $n = 30$ and (C) the solution for $n = 20$. A penalty parameter of $\lambda = 0.1$ is used for (B) and (C).

6.4 CPU Times Versus Problem Size

For a sense of the practical performance of the block coordinate descent method, we implemented it in Matlab, using Mosek to solve the quadratic program at each step. For problem sizes ranging from $p = 400$ to $p = 1000$, we solved the problem using $n = 2p$ samples. In all cases, one sweep through all the columns was sufficient to achieve an absolute accuracy of $\varepsilon = 1e-7$. The Nesterov algorithm, implemented in C, was found to have a comparable computation time provided one chooses a large value for the desired accuracy. Since the Nesterov algorithm aims for a lower accuracy and requires the setting of bounds a and b on the eigenvalues of the solution, it was not included in this experiment.

In Figure 4 we plot the average CPU time to achieve convergence, along with CPU times for the Lasso for comparision. CPU times were computed using an AMD Athlon 64 2.20Ghz processor with 1.96GB of RAM. Using this very simple implementation, the block coordinate descent method solves a problem of size $p = 1000$ in about an hour and a half. The computation time could possibly be cut down by using a fast Lasso implementation to solve the optimization problem, as described in Section 3.3, and by using a suitable method to compute the square root matrix Q from (6).

6.5 Path Following Experiments

Figure 6.5 shows two path following examples. We solve two randomly generated problems of size $p = 5$ and $n = 100$ samples. The red lines correspond to elements of the solution that are zero in the true underlying inverse covariance matrix. The blue lines correspond to true nonzeros. The vertical lines mark ranges of λ for which we recover the correct sparsity pattern exactly. Note that, by Theorem 4, for λ values greater than those shown, the solution will be diagonal.

On a related note, we observe that (1) also works well in recovering the sparsity pattern of a matrix masked by noise. The following experiment illustrates this observation. We generate a sparse inverse covariance matrix of size $p = 50$ as described above. Then, instead of using an empirical covariance S as input to (1), we use $S = (\Sigma^{-1} + V)^{-1}$, where V is a randomly generated uniform noise of size $\sigma = 0.1$. We then solve (1) for various values of the penalty parameter λ.

In Figure 6, for each value of λ shown, we randomly selected 10 sample covariance matrices S of size $p = 50$ and computed the number of misclassified zeros and nonzero elements in the solution

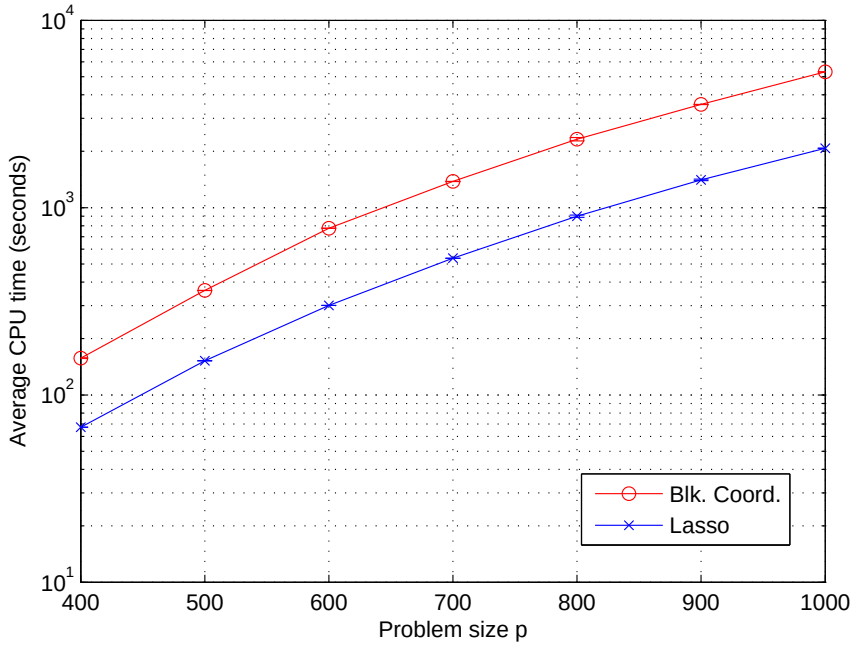


Figure 4: Average CPU times vs. problem size using block coordinate descent. We plot the average CPU time (in seconds) versus problem size. The standard deviations for the block coordinate descent method are: 0.6077, 0.2652, 0.1989, 1.0165, 48.0833, 2.7732, and 6.2314 seconds respectively. For the Lasso, the standard deviations are: 0.0221, 0.3646, 0.1878, 0.5966, 12.7279, 13.1257, and 0.6187 seconds respectively.

to (1). We plot the average percentage of errors (number of misclassified zeros plus misclassified nonzeros divided by p^2), as well as error bars corresponding to one standard deviation. As shown, the error rate is nearly zero on average when the penalty is set to equal the noise level σ.

6.6 Estimating the Stability of the Solution Using Cross-validation

Viewing our estimator as a machine that classifies elements of the concentration matrix as either zero or nonzero, we next turn to the question of the stability of the classification. Classifier instability is defined here as the probability that the classification of an arbitrary element of the matrix is changed by some small disturbance in the data. In other words, once we obtain a concentration graph by treating our estimated matrix $\hat{\Sigma}$ as an adjacency matrix, we would like to measure the stability of our results.

To empirically estimate the instability of our classifier, we can use the following simple procedure. First, we use all the available data to obtain an estimate of the concentration matrix $\hat{\Sigma}_0^{-1}$. Then we perform K-fold cross validation: randomly dividing the available data into K subsamples, we compute K different estimates $\hat{\Sigma}_i^{-1}$ by successively leaving out one subsample each time. After each estimate $\hat{\Sigma}_i^{-1}$ is computed, we count the number of matrix entries with classifications that are

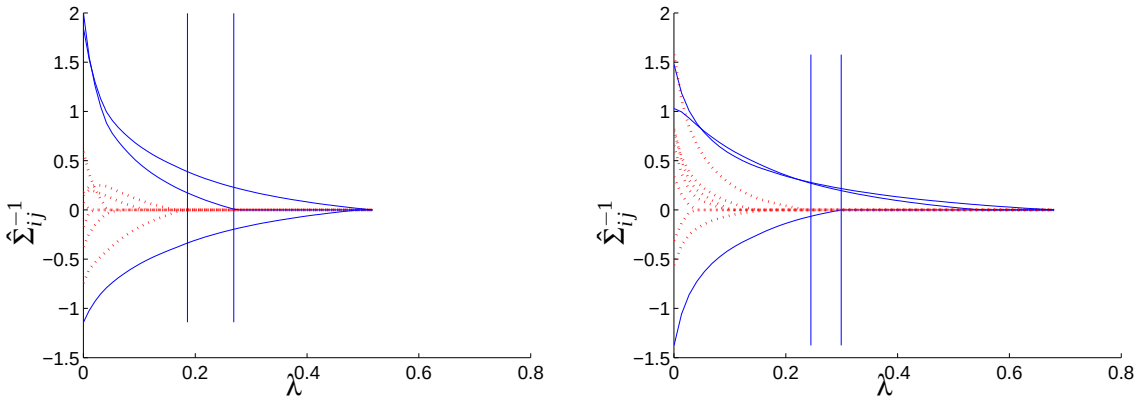


Figure 5: Path following: elements of solution to (1) as λ increases. Dashed lines correspond to elements that are zero in the true inverse covariance matrix; solid lines correspond to true nonzeros. Vertical lines mark a range of λ values using which we recover the sparsity pattern exactly.

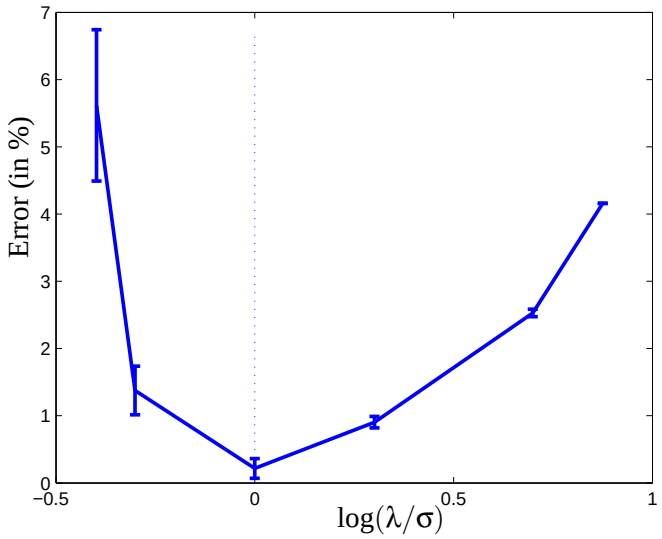


Figure 6: Recovering sparsity pattern in a matrix with added uniform noise of size $\sigma = 0.1$. We plot the average percentage or misclassified entries as a function of $\log(\lambda/\sigma)$.

different from those of $\hat{\Sigma}_0^{-1}$. Our estimate of the instability is then the average number of entries, taken over the K estimates, that were classified differently from those of $\hat{\Sigma}_0^{-1}$.

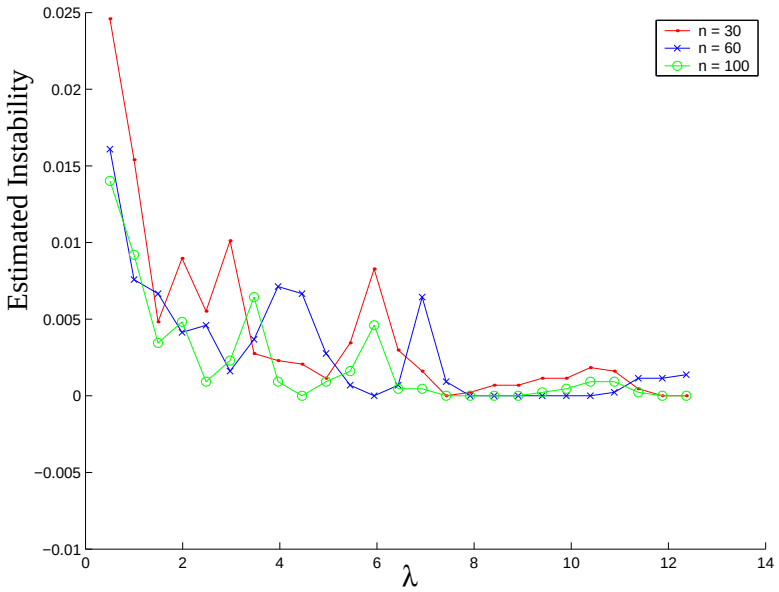


Figure 7: Estimating stability of the solution using cross validation. We plot estimated instability, the probability that the classification of an arbitrary element is changed by a small disturbance in the data, for a range of values of λ using data sets of size $n = 30, 60$, and 100 samples for a problem with $p = 30$ variables.

Below we show the results of some experiments using synthetic data. First, we randomly generated an underlying concentration matrix of size $p = 30$. Let $\delta = 0.25$ denote the fraction of nonzero elements in Σ^{-1}. Then, from the corresponding covariance matrix Σ, we generated three data sets of sizes $n = 30, 60$, and 100.

In Figure 7 we plot estimated instability versus a range of values for the penalty parameter λ, using 10-fold cross validation. We applied Theorem 4 to compute a maximum value of λ; beyond this value, the estimated graph is empty for all three data sets. As shown, even for a small number of samples, the estimated graph is fairly stable for all values of λ.

We can also see how our estimate of the instability changes as we divide the available data into a greater number of subsamples. In Figure 8 we repeat the experiment described above, this time fixing $\lambda = 0.2$ and instead varying the number of folds K in the cross validation. In these examples, for each value of n, dividing the available data into more subsamples for cross validation decreases our estimate of the instability.

6.7 Performance as a Binary Classifier

In this section we numerically examine the ability of the sparse maximum likelihood (SML) method to correctly classify elements of the inverse covariance matrix as zero or nonzero. For comparision, we will use the Lasso estimate of Meinshausen and Bühlmann (2006), which has been shown to perform extremely well. The Lasso regresses each variable against all others one at a time. Upon obtaining a solution $\theta^{(k)}$ for each variable k, one can estimate sparsity in one of two ways: either by

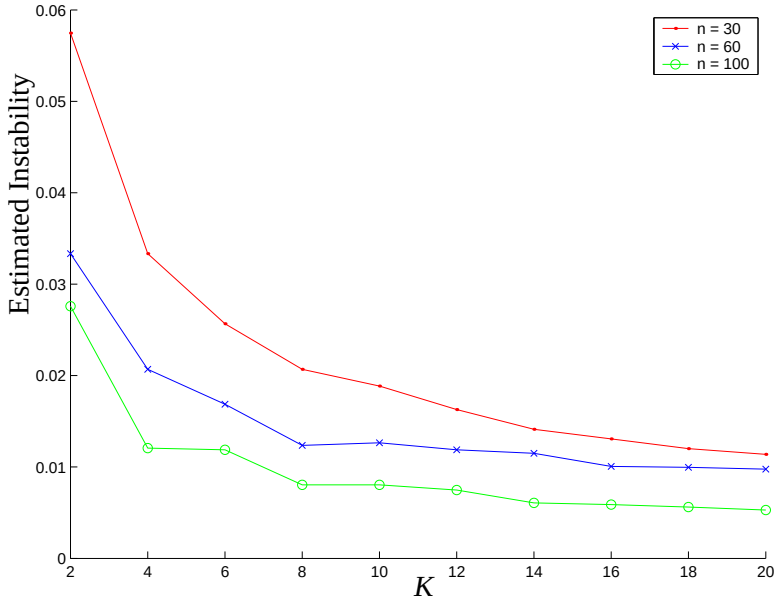


Figure 8: Estimating instability of the solution using cross validation. We plot the estimate of the instability obtained using K samples, for various values of K. Once again, we use data sets of size $n = 30, 60$, and 100 samples for a problem with $p = 30$ variables.

declaring an element $\hat{\Sigma}_{ij}$ nonzero if both $\theta_i^{(k)} \neq 0$ and $\theta_j^{(k)} \neq 0$ (Lasso-AND) or, less conservatively, if either of those quantities is nonzero (Lasso-OR).

As noted previously, Meinshausen and Bühlmann (2006) have also derived a formula for choosing their penalty parameter. Both the SML and Lasso penalty parameter formulas depend on a chosen level α, which is a bound on the same error probability for each method. For these experiments, we set $\alpha = 0.05$.

In the following experiments, we fixed the problem size p at 30 and generated sparse underlying inverse covariance matrices as described above. We varied the number of samples n from 10 to 310. For each value of n shown, we ran 30 trials in which we estimated the sparsity pattern of the inverse covariance matrix using the SML, Lasso-OR, and Lasso-AND methods. We then recorded the average number of nonzeros estimated by each method, and the average number of entries correctly identified as nonzero (true positives).

We show two sets of plots. Figure 6.7 corresponds to experiments where the true density was set to be low, $\delta = 0.05$. We plot the power (proportion of correctly identified nonzeros), positive predictive value (proportion of estimated nonzeros that are correct), and the density estimated by each method. Figure 6.7 corresponds to experiments where the true density was set to be high, $\delta = 0.40$, and we plot the same three quantities.

Meinshausen and Bühlmann (2006) report that, asymptotically, Lasso-AND and Lasso-OR yield the same estimate of the sparsity pattern of the inverse covariance matrix. At a finite number of samples, the SML method seems to fall in in between the two methods in terms of power, positive predictive value, and the density of the estimate. It typically offers, on average, the lowest total

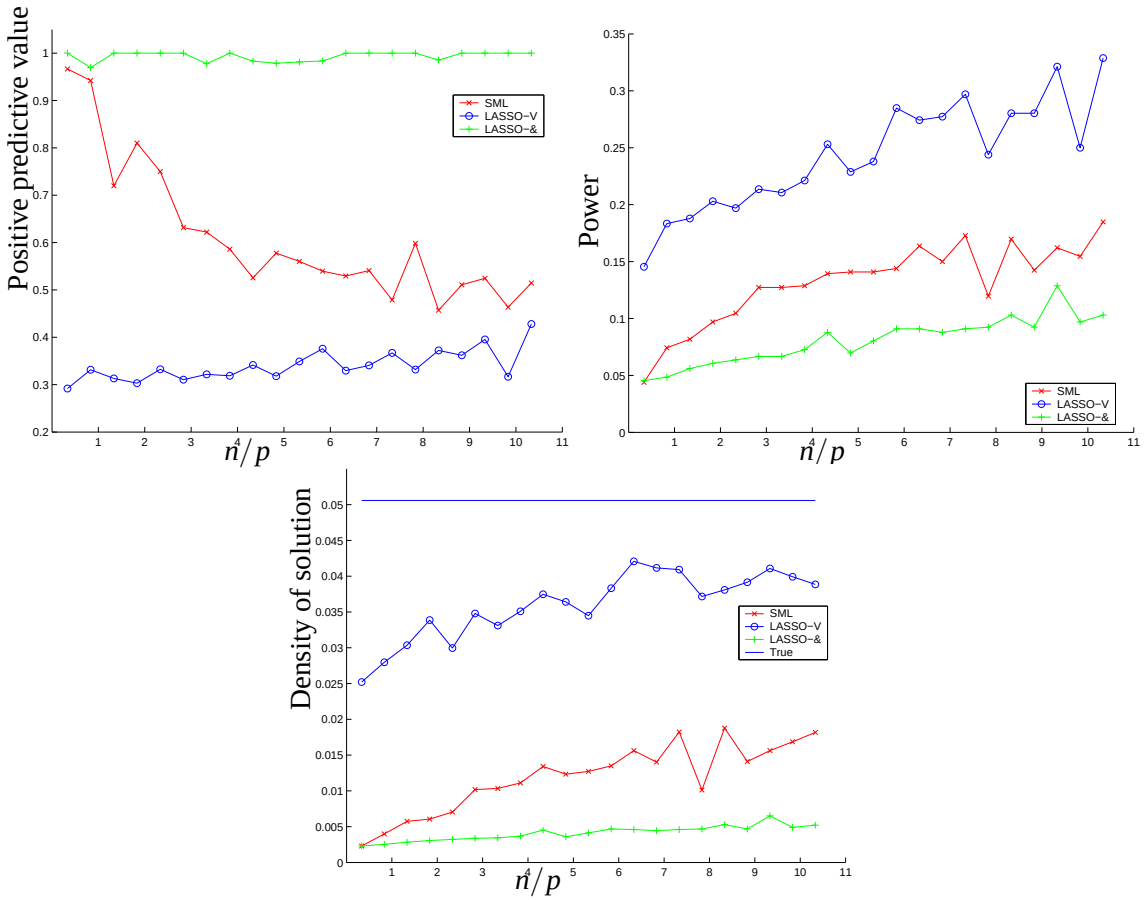


Figure 9: Classifying zeros and nonzeros for a true density of $\delta = 0.05$. We plot the positive predictive value, the power, and the estimated density using SML, Lasso-OR and Lasso-AND.

number of errors, tied with either Lasso-AND or Lasso-OR. Among the two Lasso methods, it would seem that if the true density is very low, it is slightly better to use the more conservative Lasso-AND. If the density is higher, it may be better to use Lasso-OR. When the true density is unknown, we can achieve an accuracy comparable to the better choice among the Lasso methods by computing the SML estimate. Figure 11 shows one example of sparsity pattern recovery when the true density is low.

The Lasso and SML methods have a comparable computational complexity. However, unlike the Lasso, the SML method is not parallelizable. Parallelization would render the Lasso a more computationally attractive choice, since each variable can regressed against all other separately, at an individual cost of $O(p^3)$. In exchange, SML can offer a more accurate estimate of the sparsity pattern, as well as a well-conditioned estimate of the covariance matrix itself.

6.8 Recovering Structure from Binary Data

Next we turn our attention to the binary version of the algorithm, as described in Section 5. For $p = 100$ variables, we randomly generated Ising models (10) such that the maximum degree of the

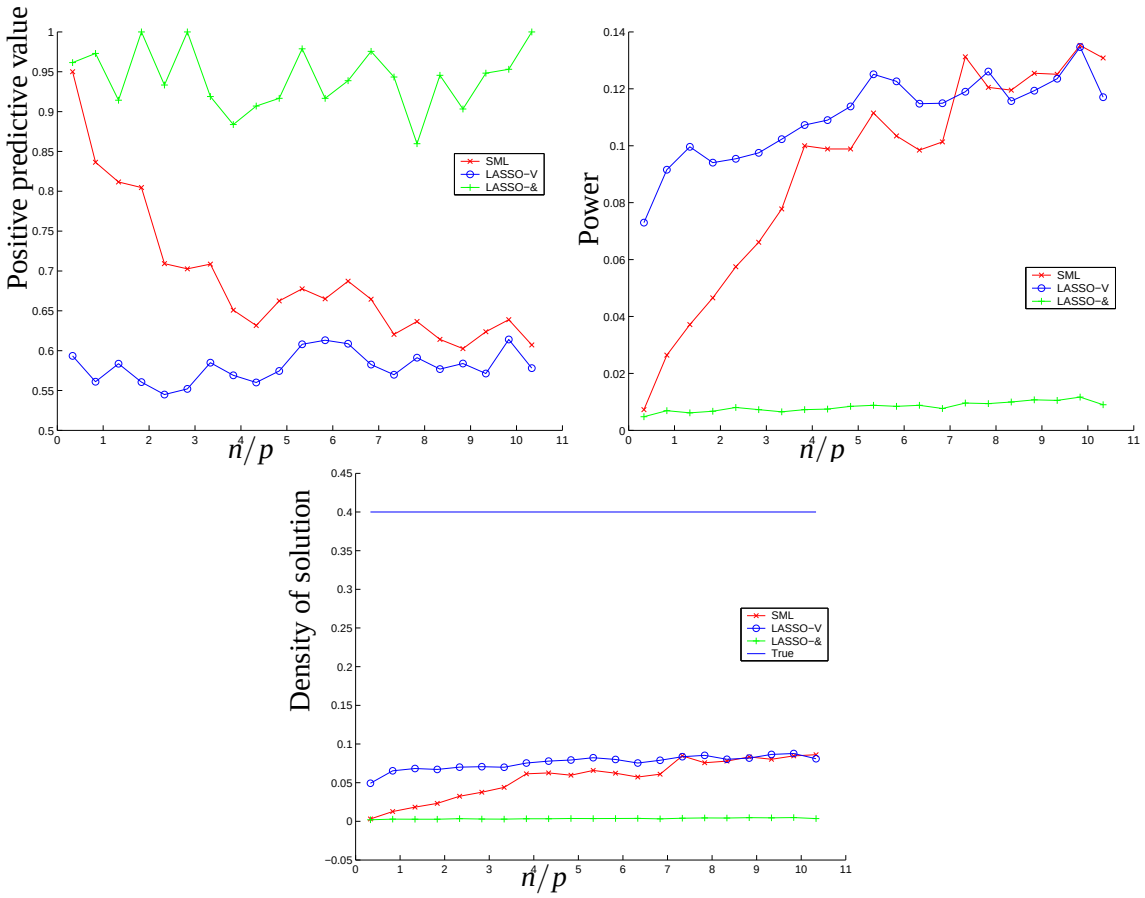


Figure 10: Classifying zeros and nonzeros for a true density of $\delta = 0.40$. We plot the positive predictive value, the power, and the estimated density using SML, Lasso-OR and Lasso-AND.

resulting graph was $d = 4$. We drew n i.i.d. samples and then attempted to recover the underlying sparsity pattern using both the approximate sparse maximum likelihood method and ℓ_1-norm penalized logistic regression as described by Wainwright et al. (2006).

In Figure 12 we plot the average sensitivity (the fraction of actual nonzeros that are correctly identified as nonzeros) as well as the average specificity (the fraction of actual zeros that are correctly identified as zeros) for a range of sample sizes. We used a significance level of $\alpha = 0.05$ for the approximate sparse maximum likelihood method, and a regularization parameter choice of $\lambda_n = 0.04 * (\log p)^3/\sqrt{n}$ for the penalized logistic regression AND method, as described by Wainwright et al. (2006).

7. Gene Expression and U.S. Senate Voting Records Data

We tested our algorithms on three sets of data: two gene expression data sets, as well as US Senate voting records. Gene expression data is often assumed to be approximately normally distributed,

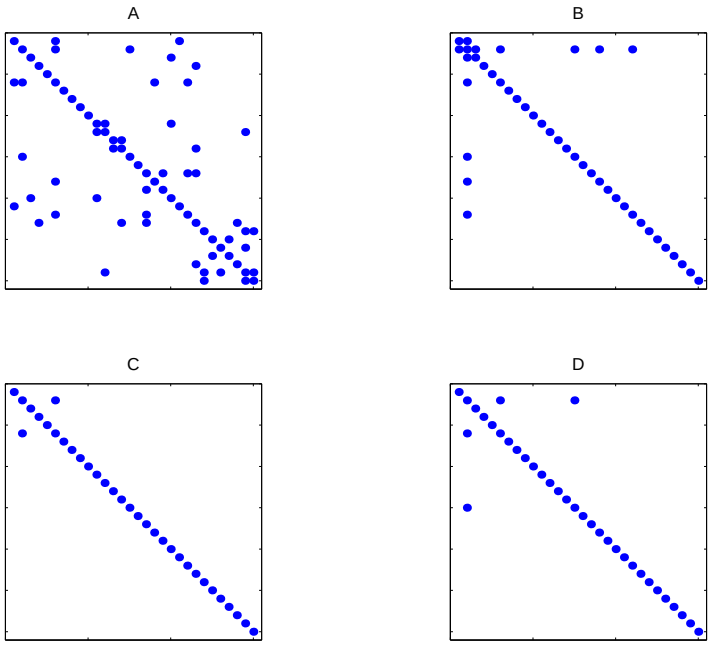


Figure 11: Comparing sparsity pattern recovery to the Lasso. (A) true covariance (B) Lasso-OR (C) Lasso-AND (D) SML.

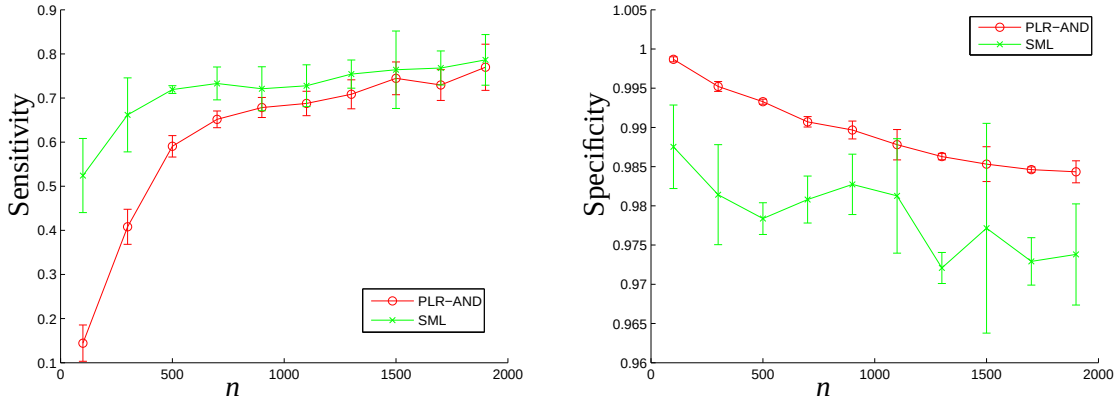


Figure 12: Comparing sparsity pattern recovery to the ℓ_1-norm penalized logistic regression (PLR). We show the sensitivity (left) as well as the specificity (right) as the number of samples is increased.

while the voting records data is here considered binary. In this section we briefly explore their respective estimated graphical models.

7.1 Rosetta Inpharmatics Compendium

We applied our algorithms to the Rosetta Inpharmatics Compendium of gene expression profiles described by Hughes et al. (2000). The 300 experiment compendium contains $n = 253$ samples with $p = 6136$ variables. With a view towards obtaining a very sparse graph, we replaced $\alpha/2p^2$ in (3) by α, and set $\alpha = 0.05$. The resulting penalty parameter is $\lambda = 0.0313$.

This is a large penalty for this data set, and by applying Theorem 4 we find that all but 270 of the variables are estimated to be independent from all the rest, clearly a very conservative estimate. Figure 13 displays the resulting graph.

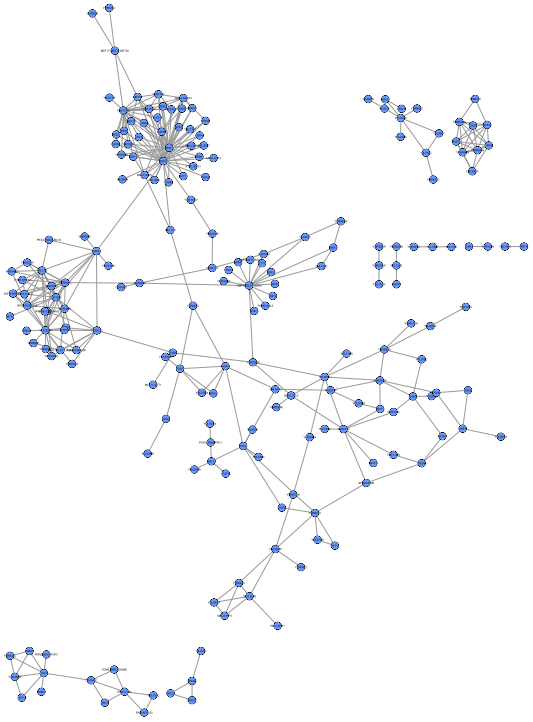


Figure 13: Application to Hughes compendium. The above graph results from solving (1) for this data set with a penalty parameter of $\lambda = 0.0313$.

Figure 14 focuses on a region of Figure 13, a cluster of genes that is unconnected to the remaining genes in this estimate. According to Gene Ontology (see Ashburner et al., 2000), these genes are associated with iron homeostasis. The probability that a gene has been false included in this cluster is at most 0.05.

As a second example, in Figure 15, we show a subgraph of genes associated with cellular membrane fusion. All three graphs were rendered using Cytoscape.

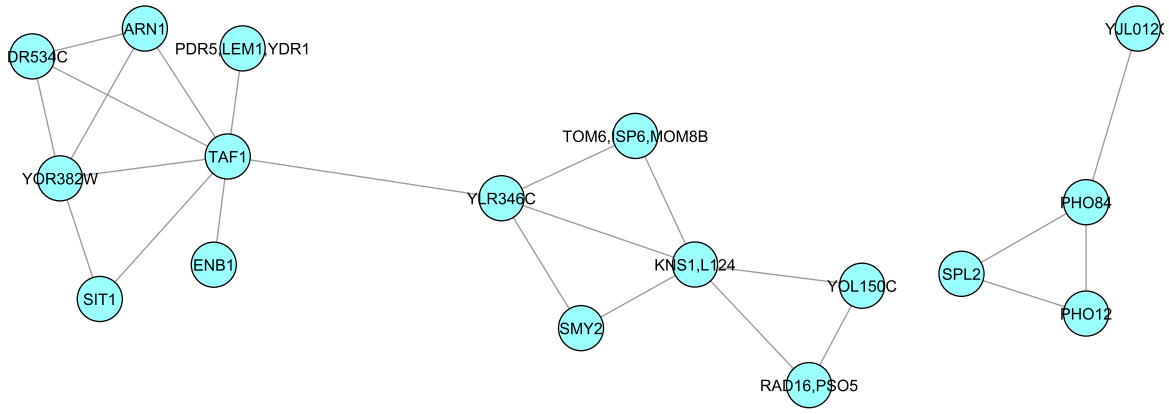


Figure 14: Application to Hughes data set (closeup of Figure 13). These genes are associated with iron homeostasis.

7.2 Iconix Microarray Data

Next we analyzed a subset of a $10,000$ gene microarray data set from 160 drug treated rat livers (Natsoulis et al., 2005). In this study, rats were treated with a variety of fibrate, statin, or estrogen receptor agonist compounds. Taking the 500 genes with the highest variance, we once again replaced $\alpha/2p^2$ in (3) by α, and set $\alpha = 0.05$. The resulting penalty parameter is $\lambda = 0.0853$.

By applying Theorem 4 we find that all but 339 of the variables are estimated to be independent from the rest. This estimate is less conservative than that obtained in the Hughes case since the ratio of samples to variables is 160 to 500 instead of 253 to 6136.

The first order neighbors of any node in a Gaussian graphical model form the set of predictors for that variable. In the estimated obtained by solving (1), we found that LDL receptor had one of the largest number of first-order neighbors in the Gaussian graphical model. The LDL receptor is believed to be one of the key mediators of the effect of both statins and estrogenic compounds on LDL cholesterol. Table 1 lists some of the first order neighbors of LDL receptor.

It is perhaps not surprising that several of these genes are directly involved in either lipid or steroid metabolism (K03249, AI411979, AI410548, NM_013200, Y00102). Other genes such as Cbp/p300 are known to be global transcriptional regulators. Finally, some are un-annotated ESTs. Their connection to the LDL receptor in this analysis may provide clues to their function.

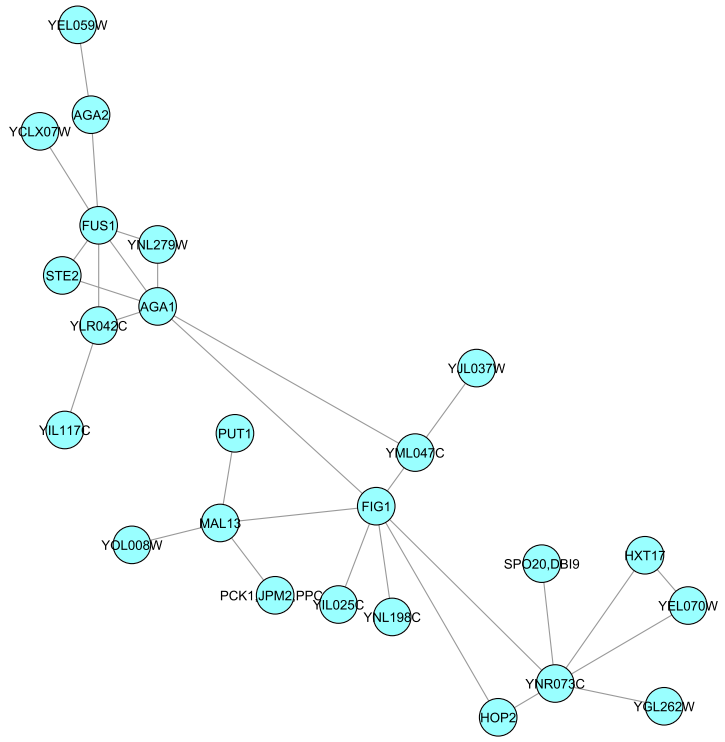


Figure 15: Application to Hughes data set (subgraph of Figure 13). These genes are associated with cellular membrane fusion.

ACCESSION	GENE
BF553500	CBP/P300-INTERACTING TRANSACTIVATOR
BF387347	EST
BF405996	CALCIUM CHANNEL, VOLTAGE DEPENDENT
NM_017158	CYTOCHROME P450, 2C39
K03249	ENOYL-COA, HYDRATASE/3-HYDROXYACYL CO A DEHYDROG.
BE100965	EST
AI411979	CARNITINE O-ACETYLTRANSFERASE
AI410548	3-HYDROXYISOBUTYRYL-CO A HYDROLASE
NM_017288	SODIUM CHANNEL, VOLTAGE-GATED
Y00102	ESTROGEN RECEPTOR 1
NM_013200	CARNITINE PALMITOYLTRANSFERASE 1B

Table 1: Predictor genes for LDL receptor.

7.3 Senate Voting Records Data

We conclude our numerical experiments by testing our approximate sparse maximum likelihood estimation method on binary data. The data set consists of US senate voting records data from the 109th congress (2004 - 2006). There are one hundred variables, corresponding to 100 senators.

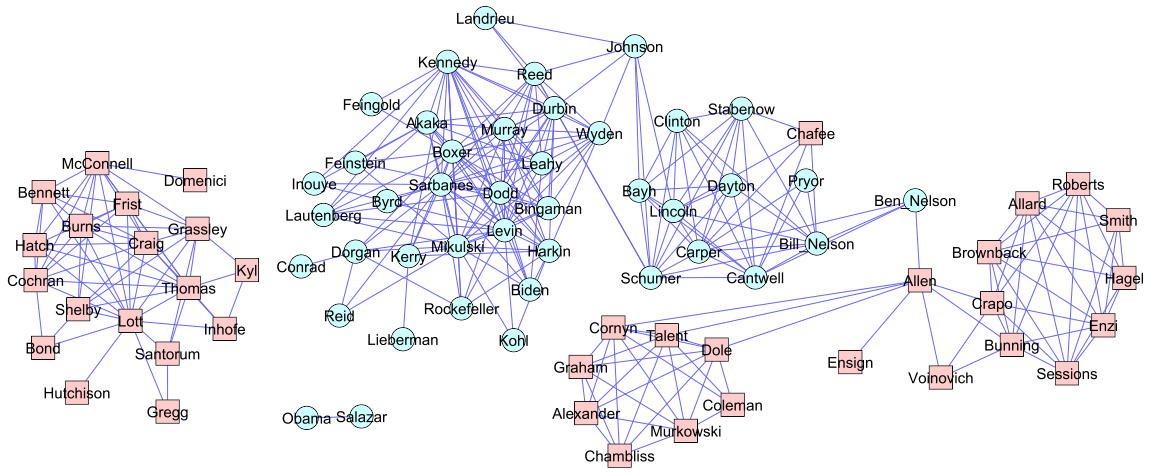


Figure 16: US Senate, 109th Congress (2004-2006). The graph displays the solution to (12) obtained using the log determinant relaxation to the log partition function of Wainwright and Jordan (2006). Democratic senators are represented by round nodes and Republican senators are represented by square nodes.

Each of the 542 samples is bill that was put to a vote. The votes are recorded as -1 for no and 1 for yes.

There are many missing values in this data set, corresponding to missed votes. Since our analysis depends on data values taken solely from $\{-1,1\}$, it was necessary to impute values to these. For this experiment, we replaced all missing votes with noes (-1). We chose the penalty parameter $\lambda(\alpha)$ according to (17), using a significance level of $\alpha = 0.05$. Figure 16 shows the resulting graphical model, rendered using Cytoscape. Red nodes correspond to Republican senators, and blue nodes correspond to Democratic senators.

We can make some tentative observations by browsing the network of senators. As neighbors most Democrats have only other Democrats and Republicans have only other Republicans. Senator Chafee (R, RI) has only Democrats as his neighbors, an observation that supports media statements made by and about Chafee during those years. Senator Allen (R, VA) unites two otherwise separate groups of Republicans and also provides a connection to the large cluster of Democrats through Ben Nelson (D, NE), which also supports media statements made about him prior to his 2006 re-election campaign. Thus, although we obtained this graphical model via a relaxation of the log partition function, the resulting picture is supported by conventional wisdom. Figure 17 shows a subgraph consisting of neighbors of degree three or lower of Senator Allen.

Finally, we estimated the instability of these results using 10-fold cross validation, as described in Section 6.6. The resulting estimate of the instability is 0.00376, suggesting that our estimate of the graphical model is fairly stable.

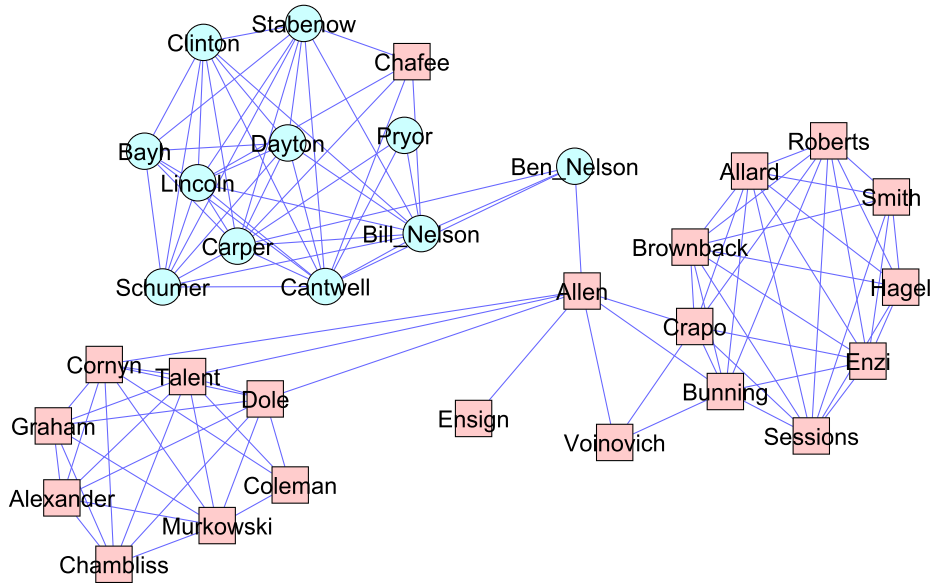


Figure 17: US Senate, 109th Congress. Neighbors of Senator Allen (degree three or lower).

Acknowledgments

We are indebted to Georges Natsoulis for his interpretation of the Iconix data set analysis and for gathering the senate voting records. We also thank Martin Wainwright, Bin Yu, Peter Bartlett, and Michael Jordan for many enlightening conversations.

Appendix A. Proof of Solution Properties and Block Coordinate Descent Convergence

In this section, we give short proofs of the two theorems on properties of the solution to (1), as well as the convergence of the block coordinate descent method.

Proof of Theorem 1:

Since $\hat{\Sigma}$ satisfies $\hat{\Sigma} = S + \hat{U}$, where $\|U\|_\infty \leq \lambda$, we have:

$$\begin{aligned}\|\hat{\Sigma}\|_2 &= \|S+\hat{U}\|_2 \\ &\leq \|S\|_2 + \|U\|_2 \leq \|S\|_2 + \|U\|_\infty \leq \|S\|_2 + \lambda p\end{aligned}$$

which yields the lower bound on $\|\hat{\Sigma}^{-1}\|_2$. Likewise, we can show that $\|\hat{\Sigma}^{-1}\|_2$ is bounded above. At the optimum, the primal dual gap is zero:

$$\begin{aligned}&-\log\det\hat{\Sigma}^{-1} + \mathrm{trace}(S\hat{\Sigma}^{-1}) + \lambda\|\hat{\Sigma}^{-1}\|_1 - \log\det\hat{\Sigma} - p \\ &= \mathrm{trace}(S\hat{\Sigma}^{-1}) + \lambda\|\hat{\Sigma}^{-1}\|_1 - p = 0\end{aligned}$$

We therefore have

$$\begin{aligned}&\|\hat{\Sigma}^{-1}\|_2 \leq \|\hat{\Sigma}^{-1}\|_F \leq \|\hat{\Sigma}^{-1}\|_1 \\ &= p/\lambda - \mathrm{trace}(S\hat{\Sigma}^{-1})/\lambda \leq p/\lambda\end{aligned}$$

where the last inequality follows from $\mathrm{trace}(S\hat{\Sigma}^{-1}) \geq 0$, since $S \succeq 0$ and $\hat{\Sigma}^{-1} \succ 0$. ■

Next we prove the convergence of block coordinate descent:

Proof of Theorem 3:

To see that optimizing over one row and column of W in (2) yields the quadratic program (4), let all but the last row and column of W be fixed. Since we know the diagonal entries of the solution, we can fix the remaining diagonal entry as well:

$$W = \begin{pmatrix} W_{\backslash p \backslash p} & w_p \\ w_p^T & W_{pp} \end{pmatrix}.$$

Then, using Schur complements, we have that

$$\det W = \det W_{\backslash p \backslash p} \cdot (W_{pp} - w_p^T (W_{\backslash p \backslash p})^{-1} w_p)$$

which gives rise to (4).

By general results on block coordinate descent algorithms (e.g., Bertsekas, 1998), the algorithms converges if (4) has a unique solution at each iteration. Thus it suffices to show that, at every sweep, $W^{(j)} \succ 0$ for all columns j. Prior to the first sweep, the initial value of the variable is positive definite: $W^{(0)} \succ 0$ since $W^{(0)} := S + \lambda I$, and we have $S \succeq 0$ and $\lambda > 0$ by assumption.

Now suppose that $W^{(j)} \succ 0$. This implies that the following Schur complement is positive:

$$w_{jj} - W_j^T (W_{\backslash j \backslash j}^{(j)})^{-1} W_j > 0$$

By the update rule we have that the corresponding Schur complement for $W^{(j+1)}$ is even greater:

$$w_{jj} - W_j^T (W_{\backslash j \backslash j}^{(j+1)})^{-1} W_j > w_{jj} - W_j^T (W_{\backslash j \backslash j}^{(j)})^{-1} W_j > 0$$

so that $W^{(j+1)} \succ 0$. ■

Finally, we apply Theorem 3 to prove the second property of the solution.

Proof of Theorem 4:

Suppose that column j of the second moment matrix satisfies $|S_{ij}| \leq \lambda$ for all $i \neq j$. This means that the zero vector is in the constraint set of (4) for that column. Each time we return to column j, the objective function will be different, but always of the form $y^T A y$ for $A \succ 0$. Since the constraint set will not change, the solution for column j will always be zero. By Theorem 3, the block coordinate descent algorithm converges to a solution, and so therefore the solution must have $\hat{\Sigma}_j = 0$. ■

Appendix B. Proof of Error Bounds

Next we shall show that the penalty parameter choice given in (3) yields the error probability bound of Theorem 2. The proof is nearly identical to that of (Meinshausen and Bühlmann, 2006, Theorem 3). The differences stem from a different objective function, and the fact that our variable is a matrix of size p rather than a vector of size p. Our proof is only an adaptation of their proof to our problem.

B.1 Preliminaries

Before we begin, consider problem (1), for a matrix S of any size:

$$\hat{X} = \arg\min -\log\det X + \mathrm{trace}(SX) + \lambda\|X\|_1$$

where we have dropped the constraint $X \succ 0$ since it is implicit, due to the log determinant function. Since the problem is unconstrained, the solution $\hat{X}$ must correspond to setting the subgradient of the objective to zero:

$$\begin{array}{ll} S_{ij} - X_{ij}^{-1} = -\lambda & \text{for } X_{ij} > 0, \\ S_{ij} - X_{ij}^{-1} = \lambda & \text{for } X_{ij} < 0, \\ |S_{ij} - X_{ij}^{-1}| \leq \lambda & \text{for } X_{ij} = 0. \end{array} \tag{18}$$

Recall that by Theorem 1, the solution is unique for λ positive.

B.2 Proof of Error Bound for Gaussian Data

Now we are ready to prove Theorem 2.

Proof of Theorem 2:

Sort columns of the covariance matrix so that variables in the same connectivity component are grouped together. The correct zero pattern for the covariance matrix is then block diagonal. Define

$$\Sigma^{\text{correct}} := \text{blk diag}(C_1, \ldots, C_\ell) \tag{19}$$

The inverse $(\Sigma^{\text{correct}})^{-1}$ must also be block diagonal, with possible additional zeros inside the blocks. If we constrain the solution to (1) to have this structure, then by the form of the objective, we can optimize over each block separately. For each block, the solution is characterized by (18).

Now, suppose that

$$\lambda > \max_{i \in N, j \in N \setminus C_i} |S_{ij} - \Sigma_{ij}^{\text{correct}}|. \tag{20}$$

Then, by the subgradient characterization of the solution noted above, and the fact that the solution is unique for $\lambda > 0$, it must be the case that $\hat{\Sigma} = \Sigma^{\text{correct}}$. By the definition of Σ^{correct}, this implies that, for $\hat{\Sigma}$, we have $\hat{C}_k = C_k$ for all $k \in N$.

Taking the contrapositive of this statement, we can write:

$$\begin{array}{l} P(\exists k \in N : \hat{C}_k \not\subseteq C_k) \\ \leq P(\max_{i \in N, j \in N \setminus C_i} |S_{ij} - \Sigma_{ij}^{\text{correct}}| \geq \lambda) \\ \leq p^2(n) \cdot \max_{i \in N, j \in N \setminus C_i} P(|S_{ij} - \Sigma_{ij}^{\text{correct}}| \geq \lambda) \\ = p^2(n) \cdot \max_{i \in N, j \in N \setminus C_i} P(|S_{ij}| \geq \lambda). \end{array} \tag{21}$$

The equality at the end follows since, by definition, $\Sigma_{ij}^{\text{correct}} = 0$ for $j \in N \setminus C_i$. It remains to bound $P(|S_{ij}| \geq \lambda)$.

The statement $|S_{kj}| \geq \lambda$ can be written as:

$$|R_{kj}|(1 - R_{kj}^2)^{-\frac{1}{2}} \geq \lambda(s_{kk}s_{jj} - \lambda^2)^{-\frac{1}{2}}$$

where R_{kj} is the correlation between variables k and j, since

$$|R_{kj}|(1 - R_{kj}^2)^{-\frac{1}{2}} = |S_{kj}|(S_{kk}S_{jj} - S_{kj}^2)^{-\frac{1}{2}}.$$

Furthermore, the condition $j \in N \backslash C_k$ is equivalent to saying that variables k and j are independent: $\Sigma_{kj} = 0$. Conditional on this, the statistic

$$R_{kj}(1-R_{kj}^2)^{-\frac{1}{2}}(n-2)^{\frac{1}{2}}$$

has a Student's t-distribution for $n-2$ degrees of freedom. Therefore, for all $j \in N \backslash C_k$,

$$\begin{aligned} &P(|S_{kj}| \geq \lambda | S_{kk} = s_{kk}, S_{jj} = s_{jj}) \\ &= 2P(T_{n-2} \geq \lambda (s_{kk}s_{jj} - \lambda^2)^{-\frac{1}{2}}(n-2)^{\frac{1}{2}} | S_{kk} = s_{kk}, S_{jj} = s_{jj}) \\ &\leq 2\tilde{F}_{n-2}(\lambda(\hat{\sigma}_k^2\hat{\sigma}_j^2 - \lambda^2)^{-\frac{1}{2}}(n-2)^{\frac{1}{2}}) \end{aligned} \tag{22}$$

where $\hat{\sigma}_k^2$ is the sample variance of variable k, and $\tilde{F}_{n-2} = 1 - F_{n-2}$ is the CDF of the Student's t-distribution with $n-2$ degree of freedom. This implies that, for all $j \in N \backslash C_k$,

$$P(|S_{kj}| \geq \lambda) \leq 2\tilde{F}_{n-2}(\lambda(\hat{\sigma}_k^2\hat{\sigma}_j^2 - \lambda^2)^{-\frac{1}{2}}(n-2)^{\frac{1}{2}})$$

since $P(A) = \int P(A|B)P(B)dB \leq K \int P(B)dB = K$. Putting the inequalities together, we have that:

$$\begin{aligned} &P(\exists k : \hat{C}_k^\lambda \not\subseteq C_k) \\ &\leq p^2 \cdot \max_{k, j \in N \backslash C_k} 2\tilde{F}_{n-2}(\lambda(\hat{\sigma}_k^2\hat{\sigma}_j^2 - \lambda^2)^{-\frac{1}{2}}(n-2)^{\frac{1}{2}}) \\ &= 2p^2\tilde{F}_{n-2}(\lambda((n-2)/((\max_{i>j}\hat{\sigma}_k\hat{\sigma}_j)^2 - \lambda^2))^{\frac{1}{2}}). \end{aligned}$$

For any fixed α, our required condition on λ is therefore

$$\tilde{F}_{n-2}(\lambda((n-2)/((\max_{i>j}\hat{\sigma}_k\hat{\sigma}_j)^2 - \lambda^2))^{\frac{1}{2}}) = \alpha/2p^2$$

which is satisfied by choosing λ according to (3). ■

B.3 Proof of Bound for Binary Data

We can reuse much of the previous proof to derive a corresponding formula for the binary case.

Proof of Theorem 7:

The proof of Theorem 7 is identical to the proof of Theorem 2, except that we have a different null distribution for $|S_{kj}|$. The null distribution of

$$nR_{kj}^2$$

is chi-squared with one degree of freedom. Analogous to (22), we have:

$$\begin{aligned} &P(|S_{kj}| \geq \lambda | S_{kk} = s_{kk}, S_{jj} = s_{jj}) \\ &= 2P(nR_{kj}^2 \geq n\lambda^2 s_{kk}s_{jj} | S_{kk} = s_{kk}, S_{jj} = s_{jj}) \\ &\leq 2\tilde{G}(n\lambda^2\hat{\sigma}_k^2\hat{\sigma}_j^2) \end{aligned}$$

where $\hat{\sigma}_k^2$ is the sample variance of variable k, and $\tilde{G} = 1 - G$ is the CDF of the chi-squared distribution with one degree of freedom. This implies that, for all $j \in N \backslash C_k$,

$$P(|S_{kj}| \geq \lambda) \leq 2\tilde{G}((\lambda\hat{\sigma}_k\hat{\sigma}_j\sqrt{n})^2).$$

Putting the inequalities together, we have that:

$$\begin{aligned} &P(\exists k : \hat{C}_k^{\lambda} \not\subseteq C_k) \\ &\leq p^2 \cdot \max_{k,j \in N \setminus C_k} 2\tilde{G}((\lambda \hat{\sigma}_k \hat{\sigma}_j \sqrt{n})^2) \\ &= 2p^2 \tilde{G}((\min_{i>j} \hat{\sigma}_k \hat{\sigma}_j)^2 n\lambda^2) \end{aligned}$$

so that, for any fixed α, we can achieve our desired bound by choosing $\lambda(\alpha)$ according to (17). ■

Appendix C. Proof of Connection Between Gaussian SML and Binary ASML

We end with a proof of Theorem 6, which connects the exact Gaussian sparse maximum likelihood problem with the approximate sparse maximum likelihood problem obtained by using the log determinant relaxation of Wainwright and Jordan (2006). First we must prove Lemma 5.

Proof of Lemma 5:

The conjugate function for the convex normalization $A(\theta)$ is defined as

$$A^*(\mu) := \sup_{\theta} \{\langle \mu, \theta \rangle - A(\theta)\}. \tag{23}$$

Wainwright and Jordan derive a lower bound on this conjugate function using an entropy bound:

$$A^*(\mu) \geq B^*(\mu). \tag{24}$$

Since our original variables are spin variables $x \{-1, +1\}$, the bound given in the paper is

$$B^*(\mu) := -\frac{1}{2} \log \det(R(\mu) + \mathrm{diag}(m)) - \frac{p}{2} \log(\frac{e\pi}{2}) \tag{25}$$

where $m := (1, \frac{4}{3}, \ldots, \frac{4}{3})$.

The dual of this lower bound is $B(\theta)$:

$$B^*(\mu) := \max_{\theta} \langle \theta, \mu \rangle - B(\theta) \leq \max_{\theta} \langle \theta, \mu \rangle - A(\theta) =: A^*(\mu). \tag{26}$$

This means that, for all μ, θ,

$$\langle \theta, \mu \rangle - B(\theta) \leq A^*(\mu) \tag{27}$$

or

$$B(\theta) \geq \langle \theta, \mu \rangle - A^*(\mu) \tag{28}$$

so that in particular

$$B(\theta) \geq \max_{\mu} \langle \theta, \mu \rangle - A^*(\mu) =: A(\theta) \tag{29}$$

Using the definition of $B(\theta)$ and its dual $B^*(\mu)$, we can write

$$\begin{aligned}
&B(\theta) := \max_\mu \langle \theta, \mu\rangle - B^*(\mu)\\
&= \tfrac{p}{2}\log(\tfrac{e\pi}{2}) + \max_\mu \tfrac{1}{2}\langle R(\theta), R(\mu)\rangle + \tfrac{1}{2}\log\det(R(\mu)+\mathrm{diag}(m))\\
&= \tfrac{p}{2}\log(\tfrac{e\pi}{2}) + \tfrac{1}{2}\cdot\max\{\langle R(\theta), X-\mathrm{diag}(m)\rangle + \log\det(X) : X \succ 0, \mathrm{diag}(X) = m\}\\
&= \tfrac{p}{2}\log(\tfrac{e\pi}{2}) + \tfrac{1}{2}\cdot\{\max_{X\succ 0}\min_\nu \langle R(\theta), X-\mathrm{diag}(m)\rangle + \log\det(X) + \nu^T(\mathrm{diag}(X)-m)\}\\
&= \tfrac{p}{2}\log(\tfrac{e\pi}{2}) + \tfrac{1}{2}\cdot\{\max_{X\succ 0}\min_\nu \langle R(\theta)+\mathrm{diag}(\nu), X\rangle + \log\det(X) - \nu^T m\}\\
&= \tfrac{p}{2}\log(\tfrac{e\pi}{2}) + \tfrac{1}{2}\cdot\{\min_\nu -\nu^T m + \max_{X\succ 0}\langle R(\theta)+\mathrm{diag}(\nu), X\rangle + \log\det(X)\}\\
&= \tfrac{p}{2}\log(\tfrac{e\pi}{2}) + \tfrac{1}{2}\cdot\{\min_\nu -\nu^T m - \log\det(-(R(\theta)+\mathrm{diag}(\nu))) - (p+1)\}\\
&= \tfrac{p}{2}\log(\tfrac{e\pi}{2}) - \tfrac{1}{2}(p+1) + \tfrac{1}{2}\cdot\{\min_\nu -\nu^T m - \log\det(-(R(\theta)+\mathrm{diag}(\nu)))\}\\
&= \tfrac{p}{2}\log(\tfrac{e\pi}{2}) - \tfrac{1}{2}(p+1) - \tfrac{1}{2}\cdot\{\max_\nu \nu^T m + \log\det(-(R(\theta)+\mathrm{diag}(\nu\lambda)\}.
\end{aligned} \tag{30}$$

■

Now we use Lemma 5 to prove the main result of Section 5.1. Having expressed the upper bound on the log partition function as a constant minus a maximization problem will help when we formulate the sparse approximate maximum likelihood problem.

Proof of Theorem 6:

The approximate sparse maximum likelihood problem is obtained by replacing the log partition function $A(\theta)$ with its upper bound $B(\theta)$, as derived in Lemma 5:

$$\begin{aligned}
&n\cdot\{\max_\theta \tfrac{1}{2}\langle R(\theta), R(\overline{z})\rangle - B(\theta) - \lambda\|\theta\|_1\}\\
&= n\cdot\{\max_\theta \tfrac{1}{2}\langle R(\theta), R(\overline{z})\rangle - \lambda\|\theta\|_1 + \tfrac{1}{2}(p+1) - \tfrac{p}{2}\log(\tfrac{e\pi}{2})\\
&+\tfrac{1}{2}\cdot\{\max_\nu \nu^T m + \log\det(-(R(\theta)+\mathrm{diag}(\nu)))\}\}\\
&= \tfrac{n}{2}(p+1) - \tfrac{np}{2}\log(\tfrac{e\pi}{2}) + \tfrac{n}{2}\cdot\max_{\theta,\nu}\{\nu^T m + \langle R(\theta), R(\overline{z})\rangle\\
&+\log\det(-(R(\theta)+\mathrm{diag}(\nu))) - 2\lambda\|\theta\|_1\}.
\end{aligned} \tag{31}$$

We can collect the variables θ and ν into an unconstrained symmetric matrix variable $Y := -(R(\theta)+\mathrm{diag}(\nu))$.

Observe that

$$\begin{aligned}
\langle R(\theta), R(\overline{z})\rangle &= \langle -Y - \mathrm{diag}(\nu), R(\overline{z})\rangle\\
&= -\langle Y, R(\overline{z})\rangle - \langle \mathrm{diag}(\nu), R(\overline{z})\rangle = -\langle Y, R(\overline{z})\rangle
\end{aligned} \tag{32}$$

and that

$$\begin{aligned}
\nu^T m &= \langle \mathrm{diag}(\nu), \mathrm{diag}(\nu)\rangle = \langle -Y - R(\theta), \mathrm{diag}(m)\rangle\\
&= -\langle Y, \mathrm{diag}(m)\rangle - \langle R(\theta), \mathrm{diag}(m)\rangle = -\langle Y, \mathrm{diag}(m)\rangle.
\end{aligned} \tag{33}$$

The approximate sparse maximum likelihood problem can then be written in terms of Y:

$$\begin{aligned}
&\tfrac{n}{2}(p+1) - \tfrac{np}{2}\log(\tfrac{e\pi}{2}) + \tfrac{n}{2}\cdot\max_{\theta,\nu}\{\nu^T m + \langle R(\theta), R(\overline{z})\rangle\\
&+\log\det(-(R(\theta)+\mathrm{diag}(m))) - 2\lambda\|\theta\|_1\}\\
&= \tfrac{n}{2}(p+1) - \tfrac{np}{2}\log(\tfrac{e\pi}{2}) + \tfrac{n}{2}\cdot\max\{\log\det Y - \langle Y, R(\overline{z})+\mathrm{diag}(m)\rangle\\
&-2\lambda\textstyle\sum_{i=2}^{p}\sum_{j=i+1}^{p+1}|Y_{ij}|\}.
\end{aligned} \tag{34}$$

If we let $M := R(\overline{z}) + \mathrm{diag}(m)$, then:

$$M = \begin{pmatrix} 1 & \overline{\mu}^T \\ \overline{\mu} & Z + \frac{1}{3}I \end{pmatrix}$$

where $\bar{\mu}$ is the sample mean and

$$Z = \frac{1}{n}\sum_{k=1}^{n} z^{(k)}(z^{(k)})^T.$$

Due to the added $\frac{1}{3}I$ term, we have that $M \succ 0$ for any data set.

The problem can now be written as:

$$\hat{Y} := \arg\max\{\log\det Y - \langle Y, M\rangle - 2\lambda \sum_{i=2}^{p}\sum_{j=i+1}^{p+1} |Y_{ij}| : Y \succ 0\}. \tag{35}$$

Since we are only penalizing certain elements of the variable Y, the solution $\hat{X}$ of the dual problem to (35) will be of the form:

$$\hat{X} = \begin{pmatrix} 1 & \bar{\mu}^T \\ \bar{\mu} & \tilde{X} \end{pmatrix}$$

where

$$\tilde{X} := \arg\max\{\log\det V : V_{kk} = Z_{kk} + \frac{1}{3},\ |V_{kj} - Z_{kj}| \leq \lambda\}.$$

We can write an equivalent problem for estimating the covariance matrix. Define a new variable:

$$\Gamma = V - \bar{\mu}\bar{\mu}^T.$$

Using this variable, and the fact that the second moment matrix about the mean, defined as before, can be written

$$S = \frac{1}{n}\sum_{k=1}^{n} z^{(k)}(z^{(k)})^T - \bar{\mu}\bar{\mu}^T = Z - \bar{\mu}\bar{\mu}^T$$

we obtain the formulation (16). Using Schur complements, we see that our primal variable is of the form:

$$Y = \begin{pmatrix} * & * \\ * & \hat{\Gamma}^{-1} \end{pmatrix}.$$

From our definition of the variable Y, we see that the parameters we are estimating, $\hat{\theta}_{kj}$, are the negatives of the off-diagonal elements of $\hat{\Gamma}^{-1}$, which gives us (15). ■

References

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. Rubin, and G. Sherlock. Gene ontology: Tool for the unification of biology. *Nature Genet.*, 25:25–29, 2000.

D. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1998.

P. J. Bickel and E. Levina. Regularized estimation of large covariance matrices. *Annals of Statistics*, 36(1):199–227, 2008.

J. Dahl, V. Roychowdhury, and L. Vandenberghe. Covariance selection for non-chordal graphs via chordal embedding. *To appear in Optimization Methods and Software*, Revised 2007.

A. d'Aspremont, L. El Ghaoui, M.I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *Advances in Neural Information Processing Systems*, 17, 2004.

A. Dobra and M. West. Bayesian covariance selection. Technical Report, 2004.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 2007.

T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, M. J. Kidd, A. M. King, M. R. Meyer, D. Slade, P. Y. Lum, S. B. Stepaniants, D. D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126, 2000.

S. Lauritzen. *Graphical Models*. Springer Verlag, 1996.

H. Li and J. Gui. Gradient directed regularization for sparse gaussian concentration graphs, with applications to inference of genetic networks. *University of Pennsylvania Technical Report*, 2005.

Z. Q. Luo and P. Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of statistics*, 34:1436–1462, 2006.

G. Natsoulis, L. El Ghaoui, G. Lanckriet, A. Tolley, F. Leroy, S. Dunlea, B. Eynon, C. Pearson, S. Tugendreich, and K. Jarnagin. Classification of a large microarray data set: algorithm comparison and analysis of drug signatures. *Genome Research*, 15:724 –736, 2005.

Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Prog.*, 103(1):127–152, 2005.

M. R. Osborne, B. Presnell, and B. A. Turlach. On the lasso and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000.

T. P. Speed and H. T. Kiiveri. Gaussian markov distributions over finite graphs. *Annals of Statistics*, 14(1):138–150, 1986.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal statistical society, series B*, 58(267-288), 1996.

L. Vandenberghe, S. Boyd, and S.-P. Wu. Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications*, 19(4):499 – 533, 1998.

M. Wainwright and M. Jordan. Log-determinant relaxation for approximate inference in discrete markov random fields. *IEEE Transactions on Signal Processing*, 2006.

M. Wainwright, P. Ravikumar, and J. D. Lafferty. High-dimensional graphical model selection using ℓ_1-regularized logistic regression. *Proceedings of Advances in Neural Information Processing Systems*, 2006.

Journal of Machine Learning Research 9 (2008) 517-519 Submitted 10/07; Revised 12/07; Published 3/08

Comments on the Complete Characterization of a Family of Solutions to a Generalized *Fisher* Criterion

Jieping Ye JIEPING.YE@ASU.EDU
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287, USA

Editor: Xiaotong Shen

Abstract

Loog (2007) provided a complete characterization of the family of solutions to a generalized *Fisher* criterion. We show that this characterization is essentially equivalent to the original characterization proposed in Ye (2005). The computational advantage of the original characterization over the new one is discussed, which justifies its practical use.

Keywords: linear discriminant analysis, dimension reduction, linear transformation

1. Generalized Fisher Criterion

For a given data set consisting of n data points $\{a_i\}_{i=1}^n$ in $\mathbb{R}^d$, a linear transformation $G \in \mathbb{R}^{d\times\ell}$ ($\ell < d$) maps each a_i for $1 \le i \le n$ in the d-dimensional space to a vector $\tilde{a}_i$ in the ℓ-dimensional space as follows:

$$G : a_i \in \mathbb{R}^d \to \tilde{a}_i = G^T a_i \in \mathbb{R}^\ell.$$

Assume that there are k classes in the data set. The within-class scatter matrix S_w, the between-class scatter matrix S_b, and the total scatter matrix S_t involved in linear discriminant analysis are defined as follows (Fukunaga, 1990):

$$\begin{aligned} S_w &= \sum_{i=1}^k (A_i - c_i e^T)(A_i - c_i e^T)^T, \\ S_b &= \sum_{i=1}^k n_i (c_i - c)(c_i - c)^T, \\ S_t &= \sum_{i=1}^k (A_i - ce^T)(A_i - ce^T)^T, \end{aligned}$$

where A_i denotes the data matrix of the i-th class, $c_i = A_i e/n_i$ is the centroid of the i-th class, n_i is the sample size of the i-th class, $c = Ae/n$ is the global centroid, and e is the vector of all ones with an appropriate length. It is easy to verify that $S_t = S_b + S_w$.

In Ye (2005), the optimal transformation G is computed by maximizing a generalized *Fisher* criterion as follows:

$$G = \arg\max_{G\in\mathbb{R}^{m\times\ell}} \operatorname{trace}\left(\left(G^T S_t G\right)^+ G^T S_b G\right), \tag{1}$$

where M^+ denotes the pseudo-inverse (Golub and Van Loan, 1996) of M and it is introduced to overcome the singularity problem when dealing with high-dimensional low-sample-size data.

1.1 Equivalent Transformation

Two linear transformations G_1 and G_2 can be considered equivalent if there is a vector v such that $G_1^T(a_i - v) = G_2^T(a_i - v)$, for $i = 1, \cdots, n$. Indeed, in this case, the difference between the projections by G_1 and G_2 is a mere shift.

Definition 1.1 *For a given data set $\{a_1, \cdots, a_n\}$, two transformations G_1 and G_2 are equivalent, if there is a vector v such that*

$$G_1^T(a_i - v) = G_2^T(a_i - v), \text{ for } i = 1, \cdots, n.$$

2. Characterization of Solutions to the Generalized *Fisher* Criterion

Let $S_t = U\Sigma U^T$ be the orthogonal eigendecomposition of S_t (note that S_t is symmetric and positive semi-definite), where $U \in \mathbb{R}^{d\times d}$ is orthogonal and $\Sigma \in \mathbb{R}^{d\times d}$ is diagonal with nonnegative diagonal entries sorted in nonincreasing order. Denote Σ_r as the r-th principal submatrix of Σ, where $r = \text{rank}(S_t)$. Partition U into two components as $U = [U_1, U_2]$, where $U_1 \in \mathbb{R}^{d\times r}$ and $U_2 \in \mathbb{R}^{d\times(d-r)}$. Note that $r \leq n$, and for high-dimensional low-sample-size data, U_1 is much smaller than U_2.

In Loog (2007), a complete family of solutions $\mathcal{S}$ to the maximization problem in Eq. (1) is given as (We correct the error in Loog (2007) by using U instead of U^T.)

$$\mathcal{S} = \left\{ U \begin{pmatrix} \Lambda Z \\ Y \end{pmatrix} \in \mathbb{R}^{d\times\ell} \;\middle|\; Z \in \mathbb{R}^{\ell\times\ell} \text{ is nonsingular }, \; Y \in \mathbb{R}^{(n-r)\times\ell} \right\},$$

where $\Lambda \in \mathbb{R}^{r\times\ell}$ maximizes the following objective function:

$$F_0(X) = \text{trace}\left(\left(X^T\Sigma_r X\right)^{-1} X^T (U_1^T S_b U_1) X \right).$$

In Ye (2005), a family of solutions $\tilde{\mathcal{S}}$ is given as

$$\tilde{\mathcal{S}} = \left\{ U \begin{pmatrix} \Lambda Z \\ 0 \end{pmatrix} \in \mathbb{R}^{d\times\ell} \;\middle|\; Z \in \mathbb{R}^{\ell\times\ell} \text{ is nonsingular } \right\}.$$

The only difference between these two characterizations of solutions is the matrix Y in $\mathcal{S}$, which is replaced by the zero matrix in $\tilde{\mathcal{S}}$. We show in the next section the equivalence relationship between these two characterizations.

3. Equivalent Solution Characterizations

Consider the following two transformations G_1 and G_2 from $\mathcal{S}$ and $\tilde{\mathcal{S}}$ respectively:

$$G_1 = U \begin{pmatrix} \Lambda Z \\ Y \end{pmatrix} \in \mathcal{S}, \quad G_2 = U \begin{pmatrix} \Lambda Z \\ 0 \end{pmatrix} \in \tilde{\mathcal{S}}.$$

Recall that $U = [U_1, U_2]$, where the columns of U_2 span the null space of S_t. Hence,

$$0 = U_2^T S_t U_2 = \sum_{i=1}^{n} U_2^T (a_i - c) \cdot (U_2^T (a_i - c))^T,$$

and $U_2^T (a_i - c) = 0$, for $i = 1, \cdots, n$, where c is the global centroid. It follows that

$$G_1^T (a_i - c) = Z^T \Lambda^T U_1^T (a_i - c) + Y^T U_2^T (a_i - c) = Z^T \Lambda^T U_1^T (a_i - c) = G_2^T (a_i - c),$$

for $i = 1, \cdots, n$. That is, G_1 and G_2 are equivalent transformations. Hence, the two solution characterizations $\mathcal{S}$ and $\tilde{\mathcal{S}}$ are essentially equivalent.

Remark 3.1 *The analysis above shows that the additional information contained in $\mathcal{S}$ is the null space, U_2, of S_t, which leads to an equivalent transformation. In $\tilde{\mathcal{S}}$, the null space U_2 is removed, which can be further justified as follows. Since $S_t = S_b + S_w$, we have*

$$0 = U_2^T S_t U_2 = U_2^T S_b U_2 + U_2^T S_w U_2.$$

It follows that $U_2^T S_b U_2 = 0$, as both S_b and S_w are positive semi-definite. Thus, the null space U_2 does not contain any discriminant information. This explains why the null space of S_t is removed in most discriminant analysis based algorithms proposed in the past.

4. Efficiency Comparison

In $\mathcal{S}$, the full matrix U is involved, whose computation may be expensive, especially for high-dimensional data. In contrast, only the first component $U_1 \in \mathbb{R}^{d \times r}$ of U is involved in $\tilde{\mathcal{S}}$, which can be computed efficiently for high-dimensional low-sample-size problem by directly working on the Gram matrix instead of the covariance matrix.

In summary, we show that $\mathcal{S}$ and $\tilde{\mathcal{S}}$ are equivalent characterizations of the solutions to the generalized *Fisher* criterion in Eq. (1). However, the latter one is preferred in practice due to its relative efficiency for high-dimensional low-sample-size data.

References

K. Fukunaga. *Introduction to Statistical Pattern Classification*. Academic Press, San Diego, California, USA, 1990.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

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.

Journal of Machine Learning Research 9 (2008) 521-540 Submitted 3/07; Revised 9/07; Published 3/08

Estimating the Confidence Interval for Prediction Errors of Support Vector Machine Classifiers

Bo Jiang BO-JIANG02@MAILS.TSINGHUA.EDU.CN
Xuegong Zhang ZHANGXG@TSINGHUA.EDU.CN
MOE Key Laboratory of Bioinformatics & Bioinformatics Div., TNLIST and Department of Automation
Tsinghua University
Beijing 100084, China

Tianxi Cai TCAI@HSPH.HARVARD.EDU
Department of Biostatistics
Harvard University
Boston, MA 02115, U.S.A.

Editor: Nicolas Vayatis

Abstract

Support vector machine (SVM) is one of the most popular and promising classification algorithms. After a classification rule is constructed via the SVM, it is essential to evaluate its prediction accuracy. In this paper, we develop procedures for obtaining both point and interval estimators for the prediction error. Under mild regularity conditions, we derive the consistency and asymptotic normality of the prediction error estimators for SVM with finite-dimensional kernels. A perturbation-resampling procedure is proposed to obtain interval estimates for the prediction error in practice. With numerical studies on simulated data and a benchmark repository, we recommend the use of interval estimates centered at the cross-validated point estimates for the prediction error. Further applications of the proposed procedure in model evaluation and feature selection are illustrated with two examples.

Keywords: *k*-fold cross-validation, model evaluation, perturbation-resampling, prediction errors, support vector machine

1. Introduction

As a state-of-the-art machine learning algorithm in classifying high-dimensional data, support vector machines (SVMs) developed by Vapnik and his colleagues (1995, 1998) have gained popularity due to many attractive features. The SVM has been used frequently in practice for developing prediction rules. After a prediction rule is constructed, the common practice is to provide a point estimate of the corresponding accuracy without accounting for the sampling variability in the estimated accuracy of the prediction rule. However, to ensure the reproducibility of the reported results, it is crucial to account for such sampling variability and provide interval estimates for the accuracy measures, especially when the sample size is not large relative to the number of unknown model parameters.

Various methods have been available to estimate the prediction error of classifiers based on the cross-validation and bootstrap methods (Efron, 1986; Efron and Tibshirani, 1997; Fu et al., 2005; Molinaro et al., 2005; Shao, 1996; Varma and Simon, 2006). When the sample size is not

sufficiently large, point estimates may be inadequate for choosing the classifier with optimized parameters or features (Reunanen, 2003; Varma and Simon, 2006). For example, in Table 1, we summarize the accuracies of SVM classifiers with different kernels based on on two artificial data sets that are generated as in Section 4.3. It appears that the polynomial kernel outperforms the linear kernel for both data sets with higher accuracy. However, it is unclear whether the difference in the higher accuracy is due to randomness. Due to its high generalization ability, the linear kernel may be preferred unless it results in significantly lower accuracy. As such, the point estimates of the accuracy measures may not provide sufficient evidence for determining which type of kernel should be used.

To adequately assess the accuracy and draw valid conclusions, it is important to account for the sampling variability in the estimated prediction error. Some studies have suggested performing hypothesis testing by considering the variability in the cross-validated estimator (Dietterich, 1998). Bengio and Grandvalet (2004) and Nadeau and Bengio (2003) pointed out that there exists no universally unbiased estimator of the variance of K-fold cross-validated estimator that is based only on the results of the cross-validation experiments. Therefore, the estimation of uncertainty around the prediction error estimators remains a theoretical, as well as practical problem.

Data Type	Sample Size	Linear Kernel Accuracy	Polynomial Kernel Accuracy
1	100	94%	95%
2	100	92%	96%

Table 1: Kernel selection in SVM classifiers based on the cross-validation point estimates for the prediction error.

To assess the predictive performance of SVM derived from data with finite sample size, probabilistic bounds such as VC-based bounds (Vapnik, 1998) and stability-based bounds (Kearns and Ron, 1999; Bousquet and Elisseeff, 2002) have been proposed. However, those theoretical bounds are too conservative to give an accurate estimation. In particular, they do not account for the sampling variability inherent in different types of data. In statistical literature, the bootstrap resampling procedure (Efron, 1979) and its variants (Efron, 1987; Wu, 1986; Liu, 1988; Hall and Mammen, 1994) provide a general framework for ascertaining variances and constructing confidence intervals, but limited effort has been made to study the distributional properties of the estimated prediction error (Efron and Tibshirani, 1995, Section 5).

In this article, we develop procedures to approximate the distribution of the estimated accuracy measures for SVM classifiers and construct confidence intervals for the accuracy measures. The proposed method, which may be linked to the weighted bootstrap resampling (Hall and Mammen, 1994; Hall and Maesono, 2000) and the Bayesian bootstrap method (Rubin, 1981), directly builds on the perturbation-resampling procedure considered in Park and Wei (2003) and Cai et al. (2005). The accuracy measure we consider is the expected absolute difference between the true and predicted responses for future subjects. For SVMs with finite-dimensional kernels, we show that the accuracy measure can be consistently estimated via cross-validation procedures, and the resulting estimators are asymptotically normal. A practical perturbation-resampling procedure is proposed to approximate the sampling distribution of the prediction error. This inference procedure is valid

without having to specify the true association between the response and the predictors. This is particularly appealing when it is difficult, if not impossible, to identify the *true* model under which the data are generated. Numerical studies based on simulated data and a benchmark repository suggest that both the variance estimator and the interval estimator centered at the cross-validated point estimator perform well. The proposed procedure is further illustrated with applications in kernel selection and in the genotypic testing for drug resistance.

2. Estimating the Prediction Error of SVM Classifiers

In this section, we provide a brief review on the construction of SVM classifiers and introduce point estimators of the accuracy measure used for evaluating the performance of SVM classifiers.

2.1 Basic Notations and Construction of SVM Classifiers

The SVM classifier is derived based on the hinge loss function:

$$L(Y, f(\mathbf{X})) = [1 - Y f(\mathbf{X})]_+ = \begin{cases} 0 & , \quad Y f(\mathbf{X}) > 1 \\ 1 - Y f(\mathbf{X}) & , \quad Y f(\mathbf{X}) \le 1 \end{cases},$$

where $\mathbf{X}$ is the input vector and $Y \in \{-1, 1\}$ is the output label, and $f(\mathbf{X})$ is the prediction function. Here, we first consider the case when $f(\mathbf{X})$ is a linear function, $f(\mathbf{X};\theta) = \mathbf{w}'\mathbf{X} + b$ (we use $\mathbf{V}'$ to denote the transpose of the vector $\mathbf{V}$ hereafter), where $\theta = (\mathbf{w}', b)'$ is the adjustable parameter. Based on $f(\cdot)$, we predict Y by the decision function $\hat{Y}(\mathbf{X},\theta) = \text{sign}\{f(\mathbf{X};\theta)\}$, where $\text{sign}(\cdot)$ denotes the sign of the function value.

To construct an optimal prediction rule, one may consider the prediction function $f(\mathbf{X};\theta)$ that minimizes the SVM risk function

$$Q(\theta) = E\{[1 - Y f(\mathbf{X};\theta)]_+\}.$$

To approximate the expected risk function $Q(\theta)$, one may consider its penalized empirical counterpart,

$$\hat{Q}_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}[1 - Y_i f(\mathbf{X}_i;\theta)]_+ + \lambda_n \mathbf{w}'\mathbf{w}, \tag{1}$$

and obtain $\hat{\theta} = \text{argmin}_\theta \hat{Q}_n(\theta)$, where $\{(\mathbf{X}_i, Y_i); i = 1, \ldots, n\}$ are n independent realizations of $(\mathbf{X}, Y)$, and λ_n is the regularization parameter that controls the amount of penalty. Subsequently, the prediction of Y may be made based on $f(\mathbf{X};\hat{\theta})$.

In practice, the minimizer $\hat{\theta}$ may be ascertained through quadratic programming techniques since the minimization of $\hat{Q}_n(\theta)$ is equivalent to the minimization of

$$\min_\alpha \{\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i Y_i (\mathbf{X}_i'\mathbf{X}_j) Y_j \alpha_j\}, \tag{2}$$

with linear constraints $0 \le \alpha_i \le C, i = 1, \ldots, n$ and $\sum_{i=1}^{n}\alpha_i Y_i = 0$, where $C = 1/(2\lambda_n n)$. Here, the constraint parameter $C = C(n)$ depends on the sample size n and typically satisfies $nC(n) \to \infty$, or equivalently $\lambda_n \to 0$, under which requirement SVM classifiers are universally consistent (Steinwart, 2002).

Note that the only way in which the input vectors appear in the minimizing problem (2) is in the form of inner products, $\mathbf{X}_i'\mathbf{X}_j$. If the input vectors are mapped to a so called "feature space" $\mathcal{H}$ via a mapping denoted by Φ, then the minimizing algorithm would only depend on the data through inner products in $\mathcal{H}$, that is, functions of the form $\Phi(\mathbf{X}_i)'\Phi(\mathbf{X}_j)$. Hence, if there is a *kernel function* $K(\cdot,\cdot)$ such that $K(\mathbf{X}_i,\mathbf{X}_j)=\Phi(\mathbf{X}_i)'\Phi(\mathbf{X}_j)$, one may carry out the minimization based on kernel function $K(\cdot,\cdot)$ only. For the simplest case when $K(\mathbf{X}_i,\mathbf{X}_j)=\mathbf{X}_i'\mathbf{X}_j$, we will refer to function $K(\cdot,\cdot)$ as the *linear kernel*. Other examples include the *polynomial kernel* $K(\mathbf{X}_i,\mathbf{X}_j)=(\gamma\mathbf{X}_i'\mathbf{X}_j+b)^d$, and the *RBF kernel* $K(\mathbf{X}_i,\mathbf{X}_j)=\exp\{-\|\mathbf{X}_i-\mathbf{X}_j\|^2/2\sigma^2\}$, with specified hyper-parameters γ, b, d and σ.

2.2 Point Estimators for the Prediction Error

To evaluate how well the trained SVM performs on a future, independent subject $(\mathbf{X}_0,Y_0)$ from the same population of $(\mathbf{X},Y)$, we consider the absolute prediction error D_0:

$$D_0 = E\left|Y_0-\hat{Y}(\mathbf{X}_0,\hat{\theta})\right|, \tag{3}$$

where $\hat{\theta}$ is the solution to minimizing function (1), and $\hat{Y}(\mathbf{X},\theta)$ is the decision function introduced in Section 2.1. Note that $\hat{\theta}$ is a function of random variables $\{(\mathbf{X}_i,Y_i);i=1,\ldots,n\}$, and the expectation E in (3) is with respect to $\{(\mathbf{X}_i,Y_i);i=1,\ldots,n\}$ and $(\mathbf{X}_0,Y_0)$. Thus, D_0 depends on sample size n and is sometimes referred to as the generalization error (see Nadeau and Bengio, 2003). To estimate D_0, we first consider the training error, which is also called apparent or re-substitution error in statistical literature, $\hat{D}=\hat{D}(\hat{\theta})$, where

$$\hat{D}(\theta) = n^{-1}\sum_{i=1}^{n}\left|Y_i-\hat{Y}(\mathbf{X}_i,\theta)\right|. \tag{4}$$

When the sample size n is small or moderate relative to the dimension of parameter θ, training error $\hat{D}(\hat{\theta})$ tends to be biased downward as an estimate of D_0. One remedy to reduce such a bias is to use the cross-validation procedure. Here we consider the commonly used K-fold cross-validation. Specifically, we randomly split the data into K disjoint subsets of about equal size and label them as $I_k, k=1,\ldots,K$. For each k, we use all observations which are not in I_k to obtain an estimate $\hat{\theta}_{(-k)}$ for θ via (1), and then compute the prediction error estimate $\hat{D}_{(k)}(\theta)$ via (4) based on observations in I_k. Then, the cross-validated prediction error estimator for D_0 is

$$\hat{\mathcal{D}} = K^{-1}\sum_{k=1}^{K}\hat{D}_{(k)}(\hat{\theta}_{(-k)}). \tag{5}$$

We show in the next section that the cross-validation estimator $\hat{\mathcal{D}}$ is consistent for estimating the prediction error of SVM classifiers under certain conditions. However, as we have mentioned above, point estimates are not adequate in drawing valid conclusions, and we need to further study the distributional properties of the estimated prediction error.

3. Interval Estimators for the Prediction Error

In this section, we provide large sample properties of the estimated prediction error. In particular, we discuss the consistency and asymptotic normality of the estimators. Based on these theoretical results, we present a simple perturbation-resampling procedure to obtain interval estimates for the prediction error. In addition, we provide inference procedures for comparing two competing models.

3.1 Large Sample Properties of Point Estimators

Suppose that the parameter θ belongs to a compact set Θ, and both the expectation $E(\mathbf{X})$ and the covariance matrix var$(\mathbf{X})$ of the input vector $\mathbf{X}$ are finite. To derive the asymptotic properties for $\hat{D}$, we first need to establish that $\hat{\theta}$ "stabilizes" as n increases, that is, $\hat{\theta}$ converges to a constant vector in probability, as $n \to \infty$. In **Theorem 1** of Appendix A, we show that under some regularity conditions, the limiting objective function $Q(\theta)$ is strictly convex with a unique minimizer θ_0, and thus for large n, there exists a unique minimizer, $\hat{\theta}$, of $\hat{Q}_n(\theta)$. Furthermore, as $n \to \infty$, $\hat{\theta} \to \theta_0$ and $\hat{D}(\hat{\theta}) \to D_0$ in probability.

To further study the large sample property of $\hat{D}$, we explore the distribution of

$$W = n^{1/2}\{\hat{D}(\hat{\theta}) - D_0\} \, . \tag{6}$$

Note that although $\hat{D}(\theta)$ is not differentiable with respect to θ, $E[\hat{D}(\theta)]$ is continuously differentiable at θ_0. In **Theorem 2** of Appendix B, we show that W is asymptotically equivalent to $n^{-1/2}\sum_{i=1}^{n}\eta_i$, and converges in distribution to a zero mean normal with variance $E(\eta_i^2)$, where η_i is defined in (14) of Appendix B. The variance of W can be approximated by

$$n^{-1}\sum_{i=1}^{n}\hat{\eta}_i^2 \, , \tag{7}$$

where $\hat{\eta}_i$ is obtained by replacing all the theoretical quantities in η_i by their empirical counterparts.

It is commonly known that the training error $\hat{D}$ is biased downward as an estimate of D_0 and hence should not be used without correction. To reduce such a bias, we consider the K-fold cross-validated estimator given in (5), where K is fixed and relatively small with respect to n. Using similar arguments as for the convergence of $\hat{D}(\hat{\theta})$, one may show that $\hat{\mathcal{D}}$ converges to D_0 in probability. Furthermore, we show in **Theorem 3** of Appendix C that

$$\mathcal{W} = n^{1/2}\{\hat{\mathcal{D}} - D_0\} \tag{8}$$

is asymptotically equivalent to W in (6) and thus $\mathcal{W}$ also converges in distribution to a zero mean normal with variance $E(\eta_i^2)$. This implies that the cross-validated estimator $\hat{\mathcal{D}}$, while potentially has less bias compared to the training error $\hat{D}$, is expected to have the same magnitude of variability as that of $\hat{D}$. Thus, we recommend to construct confidence intervals for D_0 by centering at $\hat{\mathcal{D}}$ with width determined by the variability in W. Although the proposed procedure is derived through large sample approximations, the results of numerical studies given below indicate that the distributions of W and $\mathcal{W}$ are reasonably close in finite samples.

3.2 Perturbation-Resampling Procedure for Estimating the Confidence Interval

Estimating the variance of $\mathcal{W}$ based on (7) may be difficult in practice with high-dimensional θ since it requires the estimation of the gradient of an unknown non-parametric function. To overcome such difficulties, we propose a computationally efficient perturbation-resampling procedure to approximate the distribution of $\mathcal{W}$. To be specific, let $\{G_i; i = 1, \ldots, n\}$ be a vector of independent and identically distributed *positive* random variables with unit mean and unit variance that are generated independent of the data. In practice, one may generate $\{G_i; i = 1, \ldots, n\}$ from an exponential

distribution. For any given set of $\{G_i; i = 1, \ldots, n\}$, we define

$$\hat{Q}_n^*(\theta) = \frac{1}{n}\sum_{i=1}^{n} G_i\{[1 - Y_i f(\mathbf{X}_i;\theta)]_+ + \lambda_n \mathbf{w}'\mathbf{w}\}\ , \tag{9}$$

and let θ^* be the minimizer of $\hat{Q}_n^*(\theta)$. Note that conditionally on the observed data, the only random quantities in $\hat{Q}_n^*(\theta)$ are the G's. Next, let

$$W^* = n^{-1/2}\sum_{i=1}^{n}\{|Y_i - \hat{Y}(\mathbf{X}_i, \theta^*)| - \hat{D}(\hat{\theta})\}G_i\ . \tag{10}$$

It follows from the arguments given in Appendix D that the distribution of $\mathcal{W}$ in (8) can be approximated well by the conditional distribution of W^* in (10) given the data $\{(\mathbf{X}_i, Y_i); i = 1, \ldots, n\}$. The random variables G_i used in (10) may be linked to the Bayesian bootstrap method (Rubin, 1981) with $G_i/(n^{-1}\sum_{i=1}^n G_i)$ being the weights instead.

To obtain θ^* numerically, one may solve the dual problem of (9),

$$\min_{\alpha}\{\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i Y_i(\mathbf{X}_i'\mathbf{X}_j)Y_j\alpha_j\}\ , \tag{11}$$

under the constraints $\sum_{i=1}^n \alpha_i Y_i = 0$ and $0 \le \alpha_i \le CG_i$ for $i = 1, \ldots, n$. The solution of $\mathbf{w}$ is given by $\mathbf{w}^* = (\sum_{i=1}^n Y_i\alpha_i\mathbf{X}_i)/(n^{-1}\sum_{i=1}^n G_i)$. Note that the only difference between (2) and (11) is that there is a random multiplier on the upper bound of α_i in (11). For each generated set of $\{G_i; i = 1, \ldots, n\}$, we compute the corresponding W^* via (10). By repeatedly generating $\{G_i; i = 1, \ldots, n\}$, we may obtain a large number of realizations of W^* which may be used to approximate the distribution of $\mathcal{W}$ and construct confidence intervals for D_0. For example, a $100(1-\alpha)\%$ confidence interval for D_0 may be obtained as

$$[\hat{\mathcal{D}} - n^{-1/2}\hat{\xi}_{1-\alpha/2}, \hat{\mathcal{D}} - n^{-1/2}\hat{\xi}_{\alpha/2}],$$

where $\hat{\xi}_\alpha$ is the α^{th} percentile of W^*. The integrated procedure of perturbation-resampling is given in **Algorithm 1**, where N is the number of perturbations.

Algorithm 1 Perturbation-Resampling Procedure

1: Given data $\{(\mathbf{X}_i, Y_i); i = 1, \ldots, n\}$, a classifier is trained based on the SVM algorithm
2: Estimate the cross-validation error of the classifier by using (5)
3: **for** $r = 1 \to N$ **do**
4: Generate independent positive random variables $\{G_i; i = 1, \ldots, n\}$ from an exponential distribution with unit mean and unit variance
5: Solve the quadratic programming problem (11), and calculate W_r^* by using (10)
6: **end for**
7: Estimate the resampling distribution of W^* based on $\{W_r^*; r = 1, \ldots, N\}$, which approximates the distribution of W in (6), or asymptotically the distribution of $\mathcal{W}$ in (8)
8: Use the resampling distribution to estimate the confidence interval of the prediction error centered at the cross-validation error estimate
9: Statistical evaluation of different models can be further made based on the resampling distribution (see Section 3.3)

3.3 Comparing Models Based on Interval Estimates

Suppose there are two competing models, say, $f_j(\mathbf{X};\hat{\theta}_j), j = 1,2$, where the functions f_1 and f_2 could be different in the kernels or features used, and $\hat{\theta}_j$ is the solution via (1) with the function f_j and the data $\{(\mathbf{X}_i, Y_i); i = 1,\ldots,n\}$. The theoretical and empirical prediction errors D_{0j} and $\hat{D}_j(\theta_j)$ are defined by (3) and (4) accordingly, $j = 1,2$. We are interested in making inference about, for example, $\Delta = D_{02} - D_{01}$ to assess how much improvement Model 2 is over Model 1.

A consistent estimator for Δ is $\hat{\Delta} = \hat{D}_2(\hat{\theta}_2) - \hat{D}_1(\hat{\theta}_1)$. It follows from the argument presented in Section 3.1 that

$$W_\Delta = n^{1/2}\{\hat{\Delta} - \Delta\}$$

is asymptotically normal with mean zero. To approximate this normal distribution, one may use the perturbation-resampling technique discussed in Section 3.2. Specifically, let θ_j^* be the minimizer of $n^{-1}\sum_{i=1}^n G_i\{[1 - Y_i f_j(\mathbf{X}_i;\theta)]_+ + \lambda_n \mathbf{w}'\mathbf{w}\}$, $j = 1,2$. Also, let

$$W_j^* = n^{-1/2}\sum_{i=1}^n \{|Y_i - \hat{Y}_j(\mathbf{X}_i,\theta_j^*)| - \hat{D}_j(\hat{\theta}_j)\}G_i\ ,$$

where $\hat{Y}_j(\mathbf{X},\theta) = \text{sign}\{f_j(\mathbf{X};\theta)\}$. Then, the distribution of W_Δ can be approximated by the conditional distribution of $W_\Delta^* = W_2^* - W_1^*$. Confidence intervals for Δ can then be constructed.

Note that even if $\hat{\Delta}$ is a consistent estimator for the prediction gain Δ, it represents the fitting gain of using Model 2 and may lead to a wrong comparison between models with a large probability. By applying the cross-validation procedure, the overfitted model is likely to have a larger prediction error and one would choose the more parsimonious model. Thus, the K-fold cross-validated estimator $\hat{\mathcal{D}}_2 - \hat{\mathcal{D}}_1$, where $\hat{\mathcal{D}}_j$ is defined by (5) for Model $j, j = 1,2$, may be less biased than $\hat{\Delta}$ particularly in non-asymptotic situations. Let $\mathcal{W}_j$ be defined by (8) based on Model j. Again, the resampling distribution of $\mathcal{W}_2 - \mathcal{W}_1$ can be asymptotically approximated by W_Δ^*. Based on the results of our simulated experiments, this approximation performs quite well even with limited number of samples.

4. Numerical Studies and Examples

In this section, we examine the finite-sample performance of the proposed inference procedure via extensive numerical studies based on both simulated data and a benchmark repository. Furthermore, we illustrate the new procedure with examples in kernel and biomarker selections.

4.1 Simulation Studies

We first conduct simulation studies to examine how well the proposed inference procedure performs in finite samples. The data are generated as follows: (1) the response Y is generated from $\{-1,1\}$ with equal probabilities; (2) given Y, the input vector $\mathbf{X}$ are generated from d-dimensional multivariate normal with mean $\mathbf{1}_{d\times 1}I(Y=1) + (-\mathbf{1})_{d\times 1}I(Y=-1)$, where $\mathbf{1}_{d\times 1}$ is a d-dimensional vector of ones. We consider sample sizes $n = 50$ and 100, and dimensions $d = 10, 20$, and 30. For each configuration, we generate $1,000$ independent data sets. For each simulated data set, SVM classifiers are trained by using the LIBSVM program (Chang and Lin, 2001) with a linear kernel. For simplicity, we set the penalty parameter C equal to 1 here. We estimate the empirical absolute prediction

error via 5-fold cross-validation. The distribution of the empirical absolute prediction is obtained by using perturbation-resampling procedure with $1,000$ times of perturbations ($N = 1,000$ in **Algorithm 1**). Confidence interval with nominal level of 95% is then constructed based on empirical percentiles of the resampling distribution. To evaluate normal approximation and cross-validation procedures, we also construct confidence intervals based on normal assumption with both the estimated variance and the true variance calculated from the simulation parameters of the samples. For comparison, VC-based bounds (Vapnik, 1998) and stability-based bounds (Bousquet and Elisseeff, 2002) on the prediction error are also obtained with the same nominal level of 95%.

To evaluate these interval estimates, the true prediction errors of the trained SVM classifiers are calculated according to $10,000$ replications of simulated data sets for each setting. Confidence intervals are compared with the true prediction error, and their coverage accuracies are obtained by averaging on $1,000$ data sets. Coverage accuracy is defined as the frequency for true value to fall inside the estimated confidence interval, which measures the accuracy of interval estimates. In the ideal case, the coverage accuracy of an estimated interval should be equal or close to its level of confidence, and with its length as small as possible. In Table 2, we report the coverage accuracies and average lengths of 95% confidence intervals centered at 5-fold cross-validation errors for different procedures.

Sample Size	Dimension	Empirical Percentiles[1]		Normal Estimated[2]		Normal True[3]		VC Bound	Stability Bound
		CA	AL	CA	AL	CA	AL	CA	CA
50	10	94.7	0.20	93.9	0.19	94.8	0.20	100.0	100.0
	20	94.4	0.16	92.5	0.15	94.5	0.20	100.0	100.0
	30	93.8	0.12	90.4	0.14	94.2	0.17	100.0	100.0
100	10	95.1	0.15	94.8	0.14	95.2	0.16	100.0	100.0
	20	95.2	0.15	94.5	0.13	95.1	0.16	100.0	100.0
	30	94.6	0.12	93.2	0.12	95.1	0.15	100.0	100.0

Table 2: Coverage accuracies (CA) and average lengths (AL) of 95% confidence intervals obtained by using different procedures on simulated data.

As shown in Table 2, at sample size of $n = 100$, the empirical coverage levels for the 95% confidence intervals under normal approximation with the true variance range from 95.1% to 95.2%, which validates the accuracy of cross-validation and normal approximation. In practice, the true variance of the prediction error estimator is unknown and thus the perturbation-resampling procedure would be used to ascertain the variability of the estimator. From the results in Table 2, we can see that confidence intervals obtained by the empirical percentiles of the perturbed samples perform slightly better than those constructed via normal approximation with estimated variances, in a sense that intervals based on the empirical percentiles have larger coverage accuracies with comparable

1. Interval estimates are constructed by using empirical percentiles of the resampling distribution obtained by perturbation-resampling.
2. Interval estimates are constructed as $\hat{\mathcal{D}} \pm 1.96 n^{-1/2} \hat{\sigma}$ with $\hat{\sigma}^2$ being the conditional variance of W^* estimated by perturbation-resampling.
3. Interval estimates are constructed as $\hat{\mathcal{D}} \pm 1.96 n^{-1/2} \sigma$ with σ^2 calculated as the true variance of $\mathcal{W}$.

lengths. Although the proposed algorithm may fail when the dimension of the unknown parameters is equal to or larger than the sample size, the simulation results indicate that the procedure derived through large sample approximations performs well even when sample size is moderate relative to the dimension of the parameters. On the contrary, we note that confidence bounds based on VC dimension or stability are too conservative with relative small number of samples in this example. Since these bounds are proposed to provide general guides on the construction of classifiers, they may not be suitable to account for the sampling variability from a specific population.

4.2 Variance Estimation on Benchmark Repository

We further validate the ability of the proposed procedure in estimating the variance of the cross-validation estimator on the benchmark repository used in Mika et al. (1999) and Chang and Lin (2001). The benchmark repository consists of 10 artificial and real-world data sets from the UCI, DELVE and STATLOG benchmark repositories. These data sets are collected from a variety of research areas ranging from oncology and disease diagnosis to molecular biology, astronomy, banking and signal processing. Each data set is randomly divided into 100 partitions with equal size (50 partitions for the *flare-solar*, *image* and *titanic* data sets).

To evaluate the variance estimator obtained by the perturbation-resampling procedure, we estimate the standard deviation of 5-fold cross-validation error based *only* on the *first* partition of each data set. We also obtain the 5-fold cross-validation estimates of the SVM classifier on the rest 99 partitions, and the results are used to calculate the sample standard deviation of the cross-validation estimator, which is regarded as the true value. For comparison, we estimate the standard deviation based on two other methods proposed by Nadeau and Bengio (2003) using the first partition of each data set. The first approach is performed by randomly splitting data into two distinct sets (we name it "splitting" method here), and the second approach is based on the approximation of a so-called statistic ρ (we name it "ρ-based" method here). The description of the data sets, the standard deviations estimated by different methods, and their computational efficiencies are shown in Table 3. Computational time is tested on a PC with a Pentium 4 running at 2.8GHz and 512MB of RAM.

The results in Table 3 suggest that the perturbation-resampling based estimate of the standard deviation using *only* the first partition of each data set is rather close to the sample standard deviation estimated using the entire data set. To the contrary, the standard deviation estimated by splitting the data set tends to be biased upward, while the ρ-based method tends to underestimate the standard deviation of the cross-validation error. In the results shown above, $1,000$ times of randomly splitting or resampling are used in all the three methods, and as a result, the actual computational efficiencies of different methods are comparable. This study demonstrates that the proposed perturbation-resampling procedure can be an accurate and efficient way to estimate the variance of the cross-validation error.

4.3 Example in Kernel Selection

To illustrate the application of the proposed procedure in model comparison, we perform kernel selection for SVM classifiers on simulated data. Samples $\{(X_{1i}, X_{2i}); i = 1, \ldots, n\}$ are generated from a uniform distribution on two-dimensional area $[0,1] \times [0,1]$. For data type 1, two classes of samples are separated by the curve corresponding to a linear function, $X_1 + X_2 = 1$, with a few exceptions introduced as "noise". For data type 2, the separating curve corresponds to a cubic function $X_1^3 + X_2^3 = 1$. Intuitively, samples of data type 1 should be classified well by using the simple linear

Data Set	Sample Size	Dimension	True[4] Sd	Resampling[5] Sd	Resampling[5] Time	Splitting[6] Sd	Splitting[6] Time	ρ-based[7] Sd	ρ-based[7] Time
banana	53	2	0.089	0.090	1.8	0.095	2.2	0.029	2.1
covtype	200	54	0.058	0.060	29	0.044	36	0.016	29
flare-solar	21	9	0.120	0.124	1.3	0.113	1.2	0.036	1.2
ijcnn1	283	22	0.022	0.023	21	0.024	25	0.009	23
image	46	18	0.081	0.089	4.0	0.123	4.2	0.037	3.9
ringnorm	74	20	0.065	0.067	17	0.107	16	0.030	16
svmguide1	70	4	0.029	0.030	88	0.055	71	0.017	66
titanic	44	3	0.078	0.082	1.3	0.123	1.3	0.034	1.2
twonorm	74	20	0.027	0.026	3.9	0.084	3.9	0.021	3.8
waveform	50	21	0.046	0.049	3.1	0.130	3.4	0.033	3.4

Table 3: Estimating the standard deviation of the 5-fold cross-validation error using different methods (computational time is shown in seconds).

kernel, while the cubic polynomial kernel might perform better when classifying samples from data type 2. We generate each type of data with sample size n equal to 100 and 200, respectively.

Polynomial kernels can be generalized as $K(\mathbf{X}_i, \mathbf{X}_j) = (\gamma \mathbf{X}_i' \mathbf{X}_j + b)^d$, where $\mathbf{X}_i$ and $\mathbf{X}_j$, $i, j = 1, \ldots, n$, are input vectors. In our study, we choose the hyper-parameters as $\gamma = 1/n$, $b = 0$, and $d = 3$. Then we apply the SVM algorithm by using the linear kernel and the polynomial kernel with optimal hyper-parameter C chosen by a cross-validation procedure, respectively. To make inference about the performances of different kernels, we use the model comparison procedure introduced in Section 3.3 to obtain 95% confidence intervals for the difference in cross-validation errors when using different kernels. We also compute the true prediction errors, together with the exact confidence intervals on the difference between their cross-validated estimates, based on the prediction results of $1,000$ replications of simulated data sets for each setting. In Table 4, we report the 10-fold cross-validation errors by using linear and polynomial kernels, the 95% confidence intervals on the difference between errors, and their respective true values.

For the first type of data, although the polynomial kernel could potentially lead to slightly lower error rates compared to the linear kernel, 95% confidence intervals for error difference are quite tight around zero. This suggests that the classifiers obtained based on these two types of kernels have similar accuracies as we expect. On the other hand, for the second type of data, 95% confidence intervals for error differences tend to deviate downward from zero, which indicates that the polynomial kernel indeed performs better than the linear kernel. (At the significant level of 0.05, $n = 100$ is not sufficient to conclude this, whereas $n = 200$ allows to make the above statement.) These

4. The true standard deviation (Sd) is calculated based on 5-fold cross-validation errors estimated on the rest 99 partitions of the data.
5. Standard deviation (Sd) and computation time (Time) are obtained by applying perturbation-resampling method on the first partition of the data.
6. Standard deviation (Sd) and computation time (Time) are obtained by applying splitting method (Nadeau and Bengio, 2003) on the first partition of the data.
7. Standard deviation (Sd) and computation time (Time) are obtained by applying ρ-based method (Nadeau and Bengio, 2003) on the first partition of the data.

Data Type	Sample Size	Linear Kernel Errors		Polynomial Kernel Errors		.95 Interval on Difference Between CV Errors	
		CV[8]	True[9]	CV	True	Estimated	Exact[10]
1	100	0.060	0.055	0.050	0.065	[-0.111, 0.066]	[-0.050, 0.080]
2	100	0.080	0.078	0.040	0.037	[-0.120, 0.004]	[-0.130, 0.001]
1	200	0.080	0.074	0.075	0.077	[-0.041, 0.024]	[-0.035, 0.025]
2	200	0.150	0.153	0.085	0.087	[-0.106, -0.004]	[-0.105, -0.005]

Table 4: Kernel selection based on the interval estimates of the difference in cross-validation errors.

conclusions are consistent with the intuitions behind the data generating procedure. In particular, the predicted results are consistent with the true values of both point and interval estimates obtained by simulating a large number of data sets. This study serves as an example to demonstrate how to use the proposed model comparison procedure to choose an appropriate kernel in constructing SVM classifiers.

4.4 Example in the Genotypic Testing for Drug Resistance

In this section, we give an example to show how the proposed procedure can be used in selecting important markers in the genotypic testing for HIV protease inhibitor (PI) resistance on the HIV RT and Protease Sequence Database (Rhee et al., 2003). First, we divide the sample set into two classes by labeling each protease sequence sample with 99 amino acids as either "resistant" or "susceptible", depending on whether the resistance factor of the sample exceeds a certain drug-specific cutoff value or not (Beerenwinkel et al., 2002). Then, we predict the resistance to seven FDA-approved PIs using 10 sites on the substrate binding cleft or its flap that are reported to cause resistance by reducing the binding affinity between the inhibitor and the mutant protease enzyme. Aside from these mutations, mutation information at site 90, denoted by $X_{(90)}$, on the protease sequence has been reported to either contribute to or directly confer *in vitro* and *in vivo* resistance to each of the seven approved PIs, but the mechanism by which these mutations cause PI resistance is still not known. It is interesting to assess the incremental value of $X_{(90)}$ in predicting HIV drug resistance. To this end, we compare the prediction errors for the models with and without $X_{(90)}$ and evaluate the incremental value of $X_{(90)}$ based on the reduction in the prediction error, denoted by $\Delta_{X_{(90)}}$. We obtain the point and interval estimates of $\Delta_{X_{(90)}}$ based on the model comparison method discussed in Section 3.3 with 10-fold cross-validation. In both cases, the hyper-parameter C is chosen by using a cross-validation procedure, respectively.

The results in Table 5 show that the 95% confidence intervals for $\Delta_{X_{(90)}}$ are tight around zero for drugs APV, ATV, and LPV, which indicates that $X_{(90)}$ adds rather modest value, if any, on top of other variables, for predicting resistance to these drugs. On the other hand, by including information on $X_{(90)}$, the prediction of drug resistance to IDV and RTV can be significantly improved in a sense that

8. 10-fold cross-validation errors are computed.
9. The true errors are estimated based on the prediction results of $1,000$ replications of simulated data sets for each setting.
10. The exact confidence intervals are estimated based on the prediction results of $1,000$ replications of simulated data sets for each setting.

Drug Name	Sample Size	Resistant Fraction	Without Site 90 Error	Without Site 90 .95 Interval	With Site 90 Error	With Site 90 .95 Interval	.95 Interval for Difference
APV	577	38.1%	0.149	[0.130,0.202]	0.147	[0.129,0.198]	[-0.025,0.025]
ATV	142	51.4%	0.261	[0.195,0.403]	0.197	[0.135,0.270]	[-0.110,0.022]
IDV	579	50.6%	0.123	[0.108,0.163]	0.081	[0.067,0.133]	[-0.063,-0.006]
LPV	253	74.7%	0.119	[0.090,0.236]	0.115	[0.078,0.167]	[-0.032,0.027]
NFV	617	64.0%	0.113	[0.093,0.147]	0.092	[0.076,0.130]	[-0.050,0.001]
RTV	510	50.2%	0.098	[0.069,0.123]	0.057	[0.039,0.090]	[-0.060,0.000]
SQV	598	43.6%	0.172	[0.146,0.211]	0.132	[0.113,0.166]	[-0.080,0.002]

Table 5: Interval estimates for the prediction errors and their difference in the genotypic testing for HIV drug resistance with or without mutation information at site 90 on the protease sequence.

the 95% confidence intervals for $\Delta_{X_{(90)}}$ tend to locate on the negative side of the zero point. These results are consistent with studies in literature (see Para et al., 2000; Shulman et al., 2002; Campo et al., 2003; Saah et al., 2003). Therefore, $X_{(90)}$ is an important marker for choosing antiretroviral drugs and therapies, and the roles played by $X_{(90)}$ in reducing the susceptibility of these two drugs need to be further studied.

5. Discussion

In this paper, we propose procedures for making inference about the prediction error of SVM classifiers based on cross-validated point estimators and their corresponding interval estimators. We establish large sample theory for the cross-validated estimators, and present a perturbation-resampling procedure to construct the confidence interval for prediction errors. The proposed interval estimates are obtained by approximating the spread of $\mathcal{W}$ with that of W. Alternatively, one may consider directly perturbing $\mathcal{W}$ to yield potentially better approximations. However, such a perturbation procedure may be computationally intensive since a K-fold cross-validation scheme has to be conducted for each realization of the resampling weights. Results from extensive simulation studies suggest that the proposed point and interval estimators perform well in finite samples. Furthermore, through numerical studies, we demonstrate that the interval estimates provide much more information about the true underlying prediction accuracy than the point estimates. Although it is unclear whether similar theoretical results hold for SVM classifier with the RBF kernel (see the discussion in Appendix B), the framework in this article is likely to be applicable to other inductive learning algorithms with different types of loss functions.

The proposed procedures also allow us to tackle the issue of model evaluation and selection by taking the uncertainty of estimators for the prediction error into account. We give several examples to illustrate some direct applications of the method, such as to provide confidence intervals around the estimated prediction error in kernel and biomarker selections. In addition to the examples outlined above, the proposed procedures may have other practical applications in model evaluation or variable selection.

Acknowledgments

We thank the Editor and the reviewers for many helpful comments and invaluable suggestions. This work is supported in part by NSFC grants 60575014, 30625012, 60721003, the National Basic Research Program (2004CB518605) and Hi-tech Research and Development Program (2006AA02Z325) of China, and NIH grant R01 EB006195 of USA.

Appendix A. Consistency of $\hat{\theta}$ and $\hat{D}$

In the following theorem, we will show that as $n \to \infty$, $\hat{\theta} \to \theta_0$ and the training error $\hat{D}(\hat{\theta})$ will converge to the absolute prediction error D_0 in probability. Without loss of generality, we assume that $g_0(\mathbf{X}) = P(Y = 1 \mid \mathbf{X})$ and the distribution function of $\mathbf{X}$ are continuously differentiable hereafter.

Theorem 1 Let $\theta_0 = (\mathbf{w}_0', b_0)' = \text{argmin}_{\theta \in \Theta} Q(\theta)$, Ω be the input vector space, and

$$\Lambda(Y, \theta_1) = \{\mathbf{X} \in \Omega \mid [1 - Y(\mathbf{w}_0'\mathbf{X} + b_0)][1 - Y(\mathbf{w}_1'\mathbf{X} + b_1)] < 0\}$$

for $\theta_1 = (\mathbf{w}_1', b_1)'$. Furthermore, we assume the following regularity condition:

$$P(Y = 1, \mathbf{X} \in \Lambda(1, \theta_1)) + P(Y = -1, \mathbf{X} \in \Lambda(-1, \theta_1)) > 0 \tag{12}$$

for any $\theta_1 \neq \theta_0$. Then, as $n \to \infty$, $\hat{\theta} \to \theta_0$ and $\hat{D}(\hat{\theta}) \to D_0$ in probability.

Proof. In view of Theorem 2.1 of Newey and McFadden (1994, Section 2), we can establish the convergence of $\hat{\theta} \to \theta_0$ by showing that (*a*) $Q(\theta)$ has a unique minimizer θ_0; and (*b*) $\hat{Q}_n(\theta)$ converges to $Q(\theta)$ in probability, uniformly in θ.

For (*a*), we note that since $Q(\theta)$ is continuous with respect to θ and Θ is compact, it must have a minimum within Θ. Furthermore, it is easy to verify that for any $a, b \in R$,

$$(a + b)_+ \leq a_+ + b_+ , \tag{13}$$

and a strict inequality holds if and only if $ab < 0$. As a result, under condition (12), $Q(\theta)$ is a strictly convex function at θ_0, and thus has a unique minimizer θ_0.

For (*b*), since $\hat{Q}_n(\theta)$ is also a convex function of θ because of (13), and $\hat{Q}_n(\theta)$ converges in probability to $Q(\theta)$ for each $\theta \in \Theta$, we have $\sup_{\theta \in \Theta} |\hat{Q}_n(\theta) - Q(\theta)|$ goes to zero in probability, a uniform convergence property for convex functions proved by Pollard (1991, Section 6). This concludes the proof for the convergence of $\hat{\theta}$ to θ_0 in probability.

It remains to show the consistency of $\hat{D}(\hat{\theta})$ for D_0. Since $g_0(\mathbf{X})$ is continuously differentiable, $E|Y_0 - \hat{Y}(\mathbf{X}_0, \theta)|$ is continuously differentiable in θ with bounded derivatives. Moreover, since $0 \leq E|Y_0 - \hat{Y}(\mathbf{X}_0, \theta)| \leq 2$, it follows from a uniform law of large numbers (Pollard, 1990, Chapter 8) that $\sup_{\theta \in \Theta} |\hat{D}(\theta) - E|Y_0 - \hat{Y}(\mathbf{X}_0, \theta)|| \to 0$ in probability. This, coupled with the convergence of $\hat{\theta}$ to θ_0, implies that $\hat{D}(\hat{\theta}) - D_0 \to 0$ in probability. ■

The regularity condition in (12) guarantees the existence and uniqueness of the minimizer to the objective function. This condition states that any deviation of the parameter θ from the minimizer θ_0 will always result in the change of output labels of certain samples. Given the continuous differentiability of both $g_0(\mathbf{X}) = P(Y = 1 \mid \mathbf{X})$ and the distribution function of $\mathbf{X}$, the condition can be satisfied if the probability density function of the input vector $\mathbf{X}$ is not equal to zero in some neighboring area of the optimal separating hyperplane.

Appendix B. Large Sample Distribution for $\hat{D}$

With the assumption that $g_0(\mathbf{X}) = P(Y = 1 \mid \mathbf{X})$ and the distribution function of $\mathbf{X}$ are continuously differentiable, we have $\nabla_{\theta=\theta_0} Q(\theta) = -E\{YI(Yf(\mathbf{X};\theta_0) < 1)(\mathbf{X}', 1)'\}$, which is also differentiable, almost everywhere $\theta \in \Theta$. Thus, $Q(\theta)$ is twice differentiable almost everywhere $\theta \in \Theta$. Let $\mathbf{H}$ be the Hessian matrix of $Q(\theta)$ at θ_0, $d(\theta) = E\{\hat{D}(\theta)\}$ and $\dot{d}(\theta) = \nabla d(\theta)$, we prove the following theorem:

Theorem 2 Under the regularity condition (12) in Theorem 1, the distribution of W is asymptotically equivalent to $n^{-1/2}\sum_{i=1}^n \eta_i$ and converges to a zero mean normal with variance $E(\eta_i^2)$, where

$$\eta_i = |Y_i - \hat{Y}(\mathbf{X}_i, \theta_0)| - D_0 - \dot{d}(\theta_0)\mathbf{H}^{-1}\mathbf{M}_i(\theta_0) \,, \tag{14}$$

and $\mathbf{M}_i(\theta_0) = -Y_i I(Y_i f(\mathbf{X}_i;\theta_0) < 1)(\mathbf{X}_i', 1)' + 2\lambda_n(\mathbf{w}_0', 0)'$.

Proof. Under the regularity condition, the limiting objective function $Q(\theta)$ is strictly convex at θ_0, and thus its Hessian matrix $\mathbf{H}$ at θ_0 is positive definite. To derive the asymptotic distribution theory for W, we first show that

$$\sqrt{n}(\hat{\theta} - \theta_0) = -n^{-1/2}\mathbf{H}^{-1}\sum_{i=1}^n \mathbf{M}_i(\theta_0) + o_p(1) \,, \tag{15}$$

where $\mathbf{M}_i(\theta_0) = -Y_i I(Y_i f(\mathbf{X}_i;\theta_0) < 1)(\mathbf{X}_i', 1)' + 2\lambda_n(\mathbf{w}_0', 0)'$.

To this end, let $\mathbf{Z} = (\mathbf{X}', Y)'$, $\mathbf{t} = (\mathbf{w}_t', b_t)'$, and write

$$[1 - Yf(\mathbf{X};\theta_0 + \mathbf{t})]_+ - [1 - Yf(\mathbf{X};\theta_0)]_+ = \mathbf{B}(\mathbf{Z})'\mathbf{t} + R(\mathbf{Z}, \mathbf{t}) \,, \tag{16}$$

where $\mathbf{B}(\mathbf{Z}) = -YI(Yf(\mathbf{X};\theta_0) < 1)(\mathbf{X}', 1)'$, and

$$R(\mathbf{Z}, \mathbf{t}) \;=\; \{1 - Yf(\mathbf{X};\theta_0 + \mathbf{t})\}[I\{Yf(\mathbf{X};\theta_0 + \mathbf{t}) < 1\} - I\{Yf(\mathbf{X};\theta_0) < 1\}] \,.$$

Noting that $R(\mathbf{Z}, \mathbf{0}) = 0$, and that the distribution function of $\mathbf{X}$ and the conditional probability mass function of Y given $\mathbf{X}$ are continuous differentiable, it is easy to verify that

$$ER(\mathbf{Z}, \mathbf{t}) = \frac{1}{2}\mathbf{t}'\mathbf{H}\mathbf{t} + o(\|\mathbf{t}\|^2) \text{ and } ER(\mathbf{Z}, \mathbf{t})^2 = O(\|\mathbf{t}\|^3) \,.$$

Furthermore, $E\mathbf{B}(\mathbf{Z})$ is just the first order derivative of $Q(\theta)$ at θ_0, thus $E\mathbf{B}(\mathbf{Z}) = 0$. Let $\mathbf{Z}_i = (\mathbf{X}_i', Y_i)'$, $\mathbf{s} = (\mathbf{w}_s', b_s)'$, and $A_n(\mathbf{s}) = \sum_{i=1}^n \{[1 - Y_i f(\mathbf{X}_i;\theta_0 + \mathbf{s}/\sqrt{n})]_+ + \lambda_n(\mathbf{w}_0 + \mathbf{w}_s/\sqrt{n})'(\mathbf{w}_0 + \mathbf{w}_s/\sqrt{n}) - [1 - Y_i f(\mathbf{X}_i;\theta_0)]_+ - \lambda_n \mathbf{w}_0'\mathbf{w}_0\}$. $A_n(\mathbf{s})$ is convex with respect to $\mathbf{s}$ because of (13), and it is minimized by $\sqrt{n}(\hat{\theta}_n - \theta_0)$. Note first that $nER(\mathbf{Z}, \mathbf{s}/\sqrt{n}) = \frac{1}{2}\mathbf{s}'\mathbf{H}\mathbf{s} + r_{n,0}(\mathbf{s})$, where $r_{n,0}(\mathbf{s}) = o(\|\mathbf{s}\|^2) \to 0$ for fixed $\mathbf{s}$. Accordingly, using (16),

$$\begin{aligned} A_n(\mathbf{s}) \;&=\; \sum_{i=1}^n \{[\mathbf{B}(\mathbf{Z}_i) + 2\lambda_n(\mathbf{w}_0', 0)']'\mathbf{s}/\sqrt{n} + R(\mathbf{Z}_i, \mathbf{s}/\sqrt{n}) - ER(\mathbf{Z}_i, \mathbf{s}/\sqrt{n})\} \\ &\quad + nER(\mathbf{Z}, \mathbf{s}/\sqrt{n}) + \lambda_n \mathbf{w}_s'\mathbf{w}_s \\ &=\; U_n'\mathbf{s} + \frac{1}{2}\mathbf{s}'\mathbf{H}\mathbf{s} + r_{n,0}(\mathbf{s}) + r_{n,1}(\mathbf{s}) + r_{n,2}(\mathbf{s}) \,, \end{aligned}$$

where $U_n = n^{-1/2}\sum_{i=1}^n \{\mathbf{B}(\mathbf{Z}_i) + 2\lambda_n(\mathbf{w}_0', 0)'\} = n^{-1/2}\sum_{i=1}^n \mathbf{M}_i(\theta_0)$, $r_{n,2}(\mathbf{s}) = \lambda_n \mathbf{w}_s'\mathbf{w}_s$, and $r_{n,1}(\mathbf{s}) = \sum_{i=1}^n \{R(\mathbf{Z}_i, \mathbf{s}/\sqrt{n}) - ER(\mathbf{Z}_i, \mathbf{s}/\sqrt{n})\}$. Now $r_{n,1}(\mathbf{s})$ tends to be zero in probability for each $\mathbf{s}$, since its

mean is zero and its variance is $\sum_{i=1}^{n} \mathrm{var}\{R(\mathbf{Z}_i, \mathbf{s}/\sqrt{n})\} = o(\|\mathbf{s}\|^2)$. Moreover, $r_{n,2}(\mathbf{s})$ also tends to be zero in probability when $\lambda_n \to 0$. Since $A_n(\mathbf{s})$ is a convex function, $\mathbf{H}$ is positive definite, and the covariance matrix $\mathrm{var}(\mathbf{X})$ is finite, it follows from the Basic Corollary of Hjort and Pollard (1993) that (15) holds.

Secondly, we show that the class of functions indexed by θ, $\Im = \{|Y - \hat{Y}(\mathbf{X}, \theta)| : \|\theta - \theta_0\| \le \delta\}$ is a Donsker class, where δ is a given positive number and $\hat{Y}(\mathbf{X}, \theta) = \mathrm{sign}(\mathbf{w}'\mathbf{X} + b)$. Both of the classes of function $\{1 + \hat{Y}(\mathbf{X}, \theta) : \|\theta - \theta_0\| \le \delta\}$ and $\{1 - \hat{Y}(\mathbf{X}, \theta) : \|\theta - \theta_0\| \le \delta\}$ are VC classes (van der Vaart and Wellner, 1996, Lemma 2.6.15), and hence are Donsker. Note that in this case $\Im = \{I(Y = -1)[1 + \hat{Y}(\mathbf{X}, \theta)] + I(Y = 1)[1 - \hat{Y}(\mathbf{X}, \theta)] : \|\theta - \theta_0\| \le \delta\}$, and therefore is a Donsker class. It follows that $n^{1/2}[\hat{D}(\theta) - d(\theta)]$, a process in θ, converges weakly to a zero mean Gaussian process and thus is stochastic equicontinuous at θ_0. This, coupled with (15), implies that $n^{1/2}\{\hat{D}(\hat{\theta}) - D_0\} = n^{1/2}[\hat{D}(\hat{\theta}) - \hat{D}(\theta_0)] + n^{1/2}[\hat{D}(\theta_0) - D_0]$ is asymptotically equivalent to

$$n^{1/2}\{\hat{D}(\theta_0) - D_0\} + \dot{d}(\theta_0) n^{1/2}(\hat{\theta} - \theta_0) \simeq n^{-1/2} \sum_{i=1}^{n} \eta_i \,,$$

where

$$\eta_i = |Y_i - \hat{Y}(\mathbf{X}_i, \theta_0)| - D_0 - \dot{d}(\theta_0)\mathbf{H}^{-1}\mathbf{M}_i(\theta_0) \,.$$

Here and in the sequel, we use the notation $a \simeq b$ to denote that $a = b + o_p(1)$. Thus, $W = n^{1/2}\{\hat{D}(\hat{\theta}) - D_0\}$ converges in distribution to a zero mean normal random variable. ■

Since the limiting variance of W is $\sigma^2 = E(\eta_i^2)$ and $n^{-1}\sum_{i=1}^{n} \eta_i^2$ converges to σ^2 in probability (based on the law of large numbers), one may estimate σ^2 by $n^{-1}\sum_{i=1}^{n} \eta_i^2$. Furthermore, it is not difficult to show that $n^{-1}\sum_{i=1}^{n}(\eta_i^2 - \hat{\eta}_i^2) \to 0$ since we expect that all the empirical estimates of the theoretical quantities in η_i are consistent. We note that although $n^{-1}\sum_{i=1}^{n} \hat{\eta}_i^2$ is a consistent estimator of σ^2, we approximate σ^2 based on the resampling method, not $n^{-1}\sum_{i=1}^{n} \hat{\eta}_i^2$.

For a more general case, when the prediction function $f(\mathbf{X})$ in (1) is not a linear function of the input vector $\mathbf{X}$, one can rewrite the prediction function in the form of $f(\mathbf{X}) = \mathbf{w}'\Phi(\mathbf{X}) + b$, where Φ is a mapping from the input vector space to a "feature space" $\mathcal{H}$. Given that the expectation $E\Phi(\mathbf{X})$, the covariance matrix $\mathrm{var}(\Phi(\mathbf{X}))$, and the VC dimension of $f(\mathbf{X})$ are all finite (for example, when Φ is a mapping corresponding to a polynomial kernel function), and coupled with the fact that a $\{0,1\}$-valued class of functions is a uniform Donsker class if and only if its VC dimension is finite (Dudley, 1999), one can prove all the results given above using similar arguments. Note that when it comes to construct SVM classifier using the RBF kernel, however, these conditions cannot be satisfied because of the infinite-dimensional feature space of the RBF kernel.

Appendix C. Large Sample Property of $\hat{\mathcal{D}}$

In this appendix, we will show that the distribution of $\mathcal{W}$ is asymptotically equivalent to that of W based on training error.

Theorem 3 $\mathcal{W}$ in (8) is asymptotically equivalent to W in (6).

Proof. For each partition I_k, $n^{-1/2}\{\hat{D}_{(k)}(\hat{\theta}_{(-k)}) - D_0\}$ is asymptotically equivalent to $n^{-1/2}K\sum_{i=1}^{n} I(\xi_i = k)\{|Y_i - \hat{Y}(\mathbf{X}_i, \hat{\theta}_{(-k)})| - D_0\}$, where $\{\xi_i; i = 1, \ldots, n\}$ are n exchangeable discrete random

variables uniformly distributed over $\{1,\ldots,K\}$, independent of the data, which satisfy $\sum_{i=1}^{n} I(\xi_i = k) \simeq n/K, k = 1,\ldots,K$. Conditionally on $\{\xi_i; i = 1,\ldots,n\}$, it follows from similar arguments in Appendix B that

$$\hat{\theta}_{(-k)} - \theta_0 = -\frac{K}{n(K-1)}\mathbf{H}^{-1}\sum_{i=1}^{n} I(\xi_i \neq k)\mathbf{M}_i(\theta_0) + o_p(n^{-1/2}) .$$

Then using the same argument as given for $n^{1/2}\{\hat{D}(\hat{\theta}) - D_0\}$, one can show that $n^{1/2}\{\hat{D}_{(k)}(\hat{\theta}_{(-k)}) - D_0\}$ is asymptotically equivalent to

$$n^{1/2}\{\hat{D}_{(k)}(\theta_0) - D_0\} + \dot{d}(\theta_0)n^{1/2}(\hat{\theta}_{(-k)} - \theta_0) \simeq n^{1/2}\sum_{i=1}^{n}\eta_{ki} ,$$

where

$$\eta_{ki} = I(\xi_i = k)K\{|Y_i - \hat{Y}(\mathbf{X}_i,\theta_0)| - D_0\} + I(\xi_i \neq k)\frac{K}{K-1}\dot{d}(\theta_0)\mathbf{H}^{-1}\mathbf{M}_i(\theta_0) .$$

It follows that $n^{1/2}(\hat{\mathcal{D}} - D_0) \simeq n^{-1/2}\sum_{i=1}^{n}(\sum_{k=1}^{K} K^{-1}\eta_{ki})$. Since $\sum_{k=1}^{K} I(\xi_i = k) = 1$ and $\sum_{k=1}^{K} I(\xi_i \neq k) = K-1$, it is straightforward to show that

$$\begin{aligned} n^{-1/2}\sum_{i=1}^{n}(\sum_{k=1}^{K} K^{-1}\eta_{ki}) &= n^{-1/2}\sum_{i=1}^{n}\left\{|Y_i - \hat{Y}(\mathbf{X}_i,\theta_0)| - D_0 + \dot{d}(\theta_0)\mathbf{H}^{-1}\mathbf{M}_i(\theta_0)\right\} \\ &= n^{-1/2}\sum_{i=1}^{n}\eta_i . \end{aligned}$$

■

Appendix D. Justification for the Perturbation-Resampling Procedure

Here, we give a brief justification for the perturbation-resampling approach presented in Section 3.2. For formal justification of the approach, please see similar but more rigorous derivations given in Park and Wei (2003) and Cai et al. (2005).

To justify the resampling method, we first note that it follows from the arguments in Appendix B that

$$\sqrt{n}(\hat{\theta} - \theta_0) = -n^{-1/2}\mathbf{H}^{-1}\sum_{i=1}^{n}\mathbf{M}_i(\theta_0) + o_p(1) ,$$

and that

$$\sqrt{n}(\theta^* - \theta_0) = -n^{-1/2}\mathbf{H}^{-1}\sum_{i=1}^{n} G_i\mathbf{M}_i(\theta_0) + o_p(1) ,$$

where $\mathbf{M}_i(\theta_0) = -Y_i I(Y_i f(\mathbf{X}_i;\theta_0) < 1)(\mathbf{X}_i', 1)' + 2\lambda_n(\mathbf{w}_0', 0)'$.

Then, consider the unconditional version of W^*. Let $D^*(\theta) = n^{-1}\sum_{i=1}^{n}\{|Y_i - \hat{Y}(\mathbf{X}_i,\theta)|G_i\}$, $\hat{D}(\theta) = n^{-1}\sum_{i=1}^{n}|Y_i - \hat{Y}(\mathbf{X}_i,\theta)|$, and $D_0 = E|Y_0 - \hat{Y}(\mathbf{X}_0,\hat{\theta})|$, where $(\mathbf{X}_0, Y_0)$ is an independent sample

from the same population of $(\mathbf{X}, Y)$, and the expectation E is with respect to $\{(\mathbf{X}_i, Y_i); i = 1, \ldots, n\}$ and $(\mathbf{X}_0, Y_0)$. As $\hat{\theta}$ converges to θ_0 and $\hat{D}(\hat{\theta})$ converges to D_0, it is straight forward to show that

$$\begin{aligned}
W^* &= n^{1/2}(D^*(\theta^*) - D^*(\theta_0)) - n^{1/2}(\hat{D}(\hat{\theta}) - \hat{D}(\theta_0)) \\
&\quad + n^{1/2}(D^*(\theta_0) - \hat{D}(\theta_0)) - n^{-1/2}\sum_{i=1}^{n} \hat{D}(\hat{\theta})(G_i - 1) \\
&\simeq \dot{d}(\theta_0)n^{1/2}(\theta^* - \theta_0) - \dot{d}(\theta_0)n^{1/2}(\hat{\theta} - \theta_0) \\
&\quad + n^{-1/2}\sum_{i=1}^{n} |Y_i - \hat{Y}(\mathbf{X}_i, \theta_0)|(G_i - 1) - n^{-1/2}\sum_{i=1}^{n} D_0(G_i - 1) \\
&\simeq n^{-1/2}\sum_{i=1}^{n} \{|Y_i - \hat{Y}(\mathbf{X}_i, \theta_0)| - D_0\}(G_i - 1) - \dot{d}(\theta_0)n^{-1/2}\mathbf{H}^{-1}\sum_{i=1}^{n} \mathbf{M}_i(\theta_0)(G_i - 1) \\
&= n^{-1/2}\sum_{i=1}^{n} \eta_i(G_i - 1)\ ,
\end{aligned}$$

where $\eta_i = |Y_i - \hat{Y}(\mathbf{X}_i, \theta_0)| - D_0 - \dot{d}(\theta_0)\mathbf{H}^{-1}\mathbf{M}_i(\theta_0)$.

Conditionally on the data, it follows from the Multiplier Central Limit Theorem (van der Vaart and Wellner, 1996, Chapter 2.9) that the conditional distribution of W^* converges to a normal with mean 0 and variance $n^{-1}\sum_{i=1}^{n} \eta_i^2$, which are the same as the unconditional distribution of W (or its cross-validation counterpart $\mathcal{W}$). This implies that for $\varepsilon > 0$, there exists an N_0 such that when $n > N_0$, the probability, with respect to samples $\mathcal{S} = \{(\mathbf{X}_i, Y_i); i = 1, \ldots, n\}$, of the event

$$\sup_{u \in \mathcal{R}} |P(W^* \leq u|\mathcal{S}) - P(W \leq u)| < \varepsilon\ ,$$

is at least $1 - \varepsilon$.

References

L. Beerenwinkel, B. Schmidt, H. Walter, R. Kaiser, T. Lengauer, D. Hoffmann, K. Korn, and J. Selbig. Diversity and complexity of HIV-1 drug resistance: A bioinformatics approach to predicting phenotype from genotype. *Proceedings of the National Academy of Sciences U.S.A.*, 99:8271–8276, 2002.

Y. Bengio and Y. Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 5:1089–1105, 2004.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

T. Cai, L. Tian, and L. J. Wei. Semiparametric Box-Cox power transformation models for censored survival observations. *Biometrika*, 92:619–632, 2005.

R. E. Campo, J. N. Moreno, G. Suarez, N. Miller, M. A. Kolber, D. J. Holder, M. Shivaprakash, D. M. DeAngelis, J. L. Wright, W. A. Schleif, E. A. Emini, and J. H. Condra. Efficacy of indinavir-ritonavir-based regimens in HIV-1-infected patients with prior protease inhibitor failures. *AIDS*, 17:1933–1939, 2003.

C. C. Chang and C. H. Lin. Libsvm—a library for support vector machine. *Available at http://www.csie.ntu.edu.tw/cjlin/libsvm/*, 2001.

T. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923, 1998.

R. M. Dudley. *Uniform Central Limit Theorems*. Cambridge, U.K.: Cambridge University Press, 1999.

B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7:1–26, 1979.

B. Efron. How biased is the apparent error rate of a prediction rule. *Journal of the American Statistical Association*, 81:461–470, 1986.

B. Efron. Better bootstrap confidence intervals. *Journal of the American Statistical Association*, 82: 171–185, 1987.

B. Efron and R. Tibshirani. Cross-validation and the bootstrap: Estimating the error rate of a prediction rule. Technical report, Stanford University, 1995.

B. Efron and R. Tibshirani. Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, 92:548–560, 1997.

W. J. Fu, R. J. Carroll, and S. Wang. Estimating misclassification error with small samples via bootstrap cross-validation. *Bioinformatics*, 21:1979–1986, 2005.

P. Hall and Y. Maesono. A weighted bootstrap approach to bootstrap iteration. *Journal of the Royal Statistical Society Series B*, 62:137–144, 2000.

P. Hall and E. Mammen. On general resampling algorithms and their performance in distribution estimation. *The Annals of Statistics*, 22:2011–2030, 1994.

N. L. Hjort and D. Pollard. Asymptotics for minimizers of convex processes. Technical report, Yale University, 1993.

M. Kearns and D. Ron. Algorithmic stability and sanity check bounds for leave-one-out cross-validation. *Neural Computation*, 11:1427–1453, 1999.

R. Y. Liu. Bootstrap procedures under some non-i.i.d. models. *The Annals of Statistics*, 16:1696–1708, 1988.

S. Mika, G. R*ä*tsch, J. Weston, B. Sch*ö*lkopf, and K. R. M*ü*ller. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX*, pages 41–48, 1999.

A. Molinaro, R. Simon, and R. Pfeiffer. Prediction error estimation: A comparison of resampling methods. *Bioinformatics*, 21:3301–3307, 2005.

C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52:239–281, 2003.

W. Newey and D. McFadden. Large sample estimation and hypothesis testing. In D. McFadden and R. Engler, editors, *Handbook of Econometrics IV*, pages 2113–2245. Amsterdam: North Holland, 1994.

M. F. Para, D. V. Glidden, R. Coombs, A. Collier, J. Condra, C. Craig, R. Bassett, R. Leavitt, S. Snyder, V. J. McAuliffe, and C. Boucher. Baseline human immunodeficiency virus type I phenotype, genotype, and RNA response after switching from long-term hard-capsule saquinavir to indinavir or soft-gel-capsule saquinavir in AIDS clinical trials group protocol 333. *Journal of Infectious Diseases*, 182:733–743, 2000.

Y. Park and L. J. Wei. Estimating subject-specific survival functions under the accelerated failure time model. *Biometrika*, 90:717–723, 2003.

D. Pollard. *Empirical Process: Theory and Applications*. Hayward, CA: Institute of Mathematical Statistics, 1990.

D. Pollard. Asymptotics for least absolute deviation regression estimators. *Econometric Theory*, 7: 186–199, 1991.

J. Reunanen. Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, 3:1371–1382, 2003.

S. Y. Rhee, M. J. Gonzales, R. Kantor, B. J. Betts, J. Ravela, and R. W. Shafer. Human immunodeficiency virus reverse transcriptase and protease sequence database. *Nucleic Acids Research*, 31: 298–303, 2003.

D. Rubin. The bayesian bootstrap. *The Annals of Statistics*, 9:130–134, 1981.

A. J. Saah, D. W. Haas, M. J. DiNubile, J. Chen, D. J. Holder, R. R. Rhodes, M. Shivaprakash, K. K. Bakshi, R. M. Danovich, D. J. Graham, and J. H. Condra. Treatment with indinavir, efavirenz, and adefovir after failure of nelfinavir therapy. *Journal of Infectious Diseases*, 187:1157–1162, 2003.

J. Shao. Bootstrap model selection. *Journal of the American Statistical Association*, 91:655–665, 1996.

N. Shulman, A. Zolopa, D. Havlir, A. Hsu, C. Renz, S. Boller, P. Jiang, R. Rode, J. Gallant, E. Race, D. J. Kempf, and E. Sun. Virtual inhibitory quotient predicts response to ritonavir boosting of indinavir-based therapy in human immunodeficiency virus-infected patients with ongoing viremia. *Antimicrobial Agents and Chemotherapy*, 146:3907–3916, 2002.

I. Steinwart. Support vector machines are universally consistent. *Journal of Complexity*, 18:768–779, 2002.

A. W. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes*. New York: Springer-Verlag Inc., 1996.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. New York: Springer, 1995.

V. N. Vapnik. *Statistical Learning Theory*. New York: John Wiley and Sons Inc., 1998.

S. Varma and R. Simon. Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics*, 7:91, 2006.

C. F. J. Wu. Jackknife, bootstrap, and other resampling methods in regression analysis (with discussion). *The Annals of Statistics*, 14:1261–1295, 1986.

Journal of Machine Learning Research 9 (2008) 541-558 Submitted 3/07; Revised 11/07; Published 3/08

An Information Criterion for Variable Selection in Support Vector Machines

Gerda Claeskens GERDA.CLAESKENS@ECON.KULEUVEN.BE
Christophe Croux CHRISTOPHE.CROUX@ECON.KULEUVEN.BE
Johan Van Kerckhoven JOHAN.VANKERCKHOVEN@ECON.KULEUVEN.BE
ORSTAT and Leuven Statistics Research Center
Katholieke Universiteit Leuven
B-3000 Leuven, Belgium

Editors: Isabelle Guyon and Amir Saffari

Abstract

Support vector machines for classification have the advantage that the curse of dimensionality is circumvented. It has been shown that a reduction of the dimension of the input space leads to even better results. For this purpose, we propose two information criteria which can be computed directly from the definition of the support vector machine. We assess the predictive performance of the models selected by our new criteria and compare them to existing variable selection techniques in a simulation study. The simulation results show that the new criteria are competitive in terms of generalization error rate while being much easier to compute. We arrive at the same findings for comparison on some real-world benchmark data sets.

Keywords: information criterion, supervised classification, support vector machine, variable selection

1. Introduction

We study classification using the support vector machine (SVM). We start from a training set $\{(x_i, y_i)\}$ containing n observations. Each p-dimensional observation $x_i = (x_{i1}, \ldots, x_{ip})$ has a class label y_i assigned to it, which is either $+1$ or -1. We wish to find a function $f(\cdot)$ such that for an observation x the predicted class $\hat{y} = +1$ if $f(x)$ is positive, and $\hat{y} = -1$ if $f(x)$ is negative. We want this function to assign the correct class labels to the training observations (low training error rate) and to accurately classify new observations (low generalization error rate). Working with a subset of the p variables $x_{i1}, \ldots, x_{ip}$ reduces variability of the class-label estimator and might lead to better out-of-sample predictions.

It is only true to some extent that variable selection would not be necessary in the support vector machine setting since it manages to circumvent the so-called "curse of dimensionality" (see, for example, Cristianini and Shawe-Taylor, 2000; Hastie et al., 2001; Schölkopf and Smola, 2002). While the SVM approach avoids fitting a number of parameters equal to the dimension of the input space, there remains the high probability of a perfect separation in high-dimensional problems. For example, if p is larger than the number of observations, it is always possible to perfectly separate the two classes of training data by a hyperplane. In general, the risk of overfitting will increases with the dimension for most data configurations. Hence, the risk of obtaining a decision rule with poor

generalization properties (high generalization error rate) cannot be avoided. Guyon et al. (2002) illustrate this and show that variable selection can further improve the SVM's performance.

Variable selection techniques can be divided into three categories. Filters select subsets of variables as a pre-processing step, independently of the prediction method. Wrappers use the classification method to score subsets of variables. Finally, embedded methods include variable selection into the construction of the classifier. In this paper we propose new information criteria for SVMs, yielding a wrapper method where we consider the SVM merely as a black box. We refer to Guyon and Elisseeff (2003) for an introduction to variable and feature selection in Machine Learning. Information criteria are a standard tool for model selection in traditional statistics. Information criteria for variable selection assign a numerical value to each subset of the variables under consideration. The subset with the lowest value of the information criterion is then selected. Examples are the Akaike information criterion (AIC, Akaike, 1973) and the Bayesian information criterion (BIC, Schwarz, 1978). Claeskens and Hjort (2008) survey and explain the use of common information criteria for statistical variable selection in likelihood-based models, we refer to there for more references.

For support vector machines only very few information criteria have been developed. The kernel regularisation information criterion (KRIC) of Kobayashi and Komaki (2006) was originally proposed for parameter tuning of the SVM. We apply it for variable selection. However, the KRIC has a complicated definition and is computationally expensive for large sample sizes. In this paper two new information criteria are proposed, one shares properties with AIC, the other with BIC. We want the new criteria to select a preferably compact subset of variables with good predictive properties. We will show that submodels selected by the new criteria are as performing as the ones chosen by the KRIC, while they incur substantially less computational overhead. We also make a comparison with using cross-validated error rate based criteria, as in Kearns et al. (1997). An important contribution of this paper is that our numerical comparisons show that the popular, but time consuming, cross-validation criteria are outperformed in generalization error by the new information criteria, where the latter are coming at almost no additional computational cost.

Alternative approaches perform variable selection in feature space instead of in input space (Shih and Cheng, 2005), or select a set of "maximally separating directions" in the input space (Fortuna and Capson, 2004). These methods, however, do not select a set of original input variables. Various other authors have suggested different formulations for the SVM such that variable selection is performed automatically. Examples of such embedded methods can be found in Bi et al. (2003), Zhu et al. (2004), Neumann et al. (2005), Lee et al. (2006), Wang et al. (2006), Zhang (2006), and Lin and Zhang (2006).

In Section 2 we define the support vector machine setting, we review existing information criteria and we describe ranking techniques to speed up the variable selection process. In Section 3, we define the new information criteria and highlight their advantages. Section 4 contains the results of a simulation study and in Section 5 we compare the different techniques on a few real-world benchmark data sets. Section 6 concludes and gives some directions for further research.

2. Problem Setting

In this Section we first review the definition of a support vector machine. Afterwards some existing variable selection techniques are discussed. Finally we present the ranking techniques that will be used in this paper.

2.1 The Support Vector Machine

We denote the training sample (x_i, y_i), $1 \leq i \leq n$, with x_i a p-dimensional vector of explicative variables, and $y_i \in \{-1, +1\}$ the class label. The goal is to estimate a target function $f(x)$ in the space of explicative variables such that $f(x_i) > 0$ for $y_i = +1$, and $f(x_i) < 0$ for $y_i = -1$.

We start with linear support vector machines, where $f(x)$ is of the form $f(x) = w'x + b$. For binary classification this function is obtained by solving the minimisation problem

$$\min_{w,b,\xi_i} \left\{ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \right\} \text{ subject to } \begin{cases} y_i(w'x_i + b) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = 1, \ldots, n. \end{cases} \tag{1}$$

The ξ_i are slack margin variables, indicating how close a point x_i lies to the separating boundary (if $\xi_i < 1$), or how badly it is misclassified (if $\xi_i > 1$). The tuning parameter C controls how much weight is put on trying to achieve perfect separation.

The dual problem can be solved more easily, and has the following form:

$$\min_{\alpha} \{ \frac{1}{2}\alpha' Q\alpha - \sum_{i=1}^{n} \alpha_i \} \text{ subject to } \begin{cases} 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n, \\ \sum_{i=1}^{n} y_i \alpha_i = 0. \end{cases} \tag{2}$$

Here α_i is the weight given to the observation (x_i, y_i), and Q is a positive semi-definite matrix with entries $Q_{i,j} = y_i y_j x_i' x_j$. The vector w can be found from $w = \sum_{i=1}^{n} y_i \alpha_i x_i$. The negative intercept b is found by computing $b = 0.5(r_2 - r_1)$, where

$$r_1 = \frac{\sum_{0<y_i\alpha_i<C} (Q\alpha)_i - 1}{\sum_{0<y_i\alpha_i<C} 1} \text{ and } r_2 = \frac{\sum_{0>y_i\alpha_i>-C} (Q\alpha)_i - 1}{\sum_{0>y_i\alpha_i>-C} 1}.$$

If no i exist for which $0 < y_i\alpha_i < C$, then define

$$r_1 = \frac{1}{2}\Big(\min_{\alpha_i=0, y_i=1} (Q\alpha)_i - \max_{\alpha_i=C, y_i=1} (Q\alpha)_i \Big),$$

and analogously for r_2, with $y_i = -1$. Note that we can write $\xi_i = [1 - y_i a_i]_+$, where $[x]_+ = \max\{0, x\}$ and where $a_i = f(x_i)$.

The linear SVM can be extended towards more complex decision functions in a rather straightforward way. Therefore we replace the inner products $x_i' x_j$ in the definition of Q by a more general kernel function $K(x_i, x_j)$. See Cristianini and Shawe-Taylor (2000) for the properties that these kernel functions must have. This leads to a more general decision function

$$f(x) = \sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + b. \tag{3}$$

Popular choices for the kernel function in (3) are the linear kernel, where the kernel function is $K(x,z) = x'z$, the polynomial kernel of the form $K(x,z) = (c_0 + \gamma x'z)^d$, and the radial basis kernel $K(x,z) = \exp(-\gamma\|x - z\|^2)$, where c_0, γ and d are regularization parameters that can be tuned for optimal performance of the classifier. In this more general setting, we have

$$\|w\|^2 = \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) = \alpha' Q\alpha$$

for the squared norm of the weight vector, where $Q_{i,j} = y_i y_j K(x_i, x_j)$.

2.2 Existing Variable Selection Techniques for SVM

We compare our new methods (Section 3) to variable selection based on (ten-fold) cross-validation (CV), guaranteed risk minimisation (GRM, Vapnik, 1982) and the kernel regularisation information criterion (KRIC) by Kobayashi and Komaki (2006). Each of these will be explained in more detail below.

Ten-fold cross-validation divides the training data in ten parts of roughly equal size. One part is left out, the other nine parts are the training data and are used to fit the SVM. This SVM is applied to the part that is left out to obtain an estimate of the error rate. This process is repeated ten times (each time a different part is left out) to obtain the CV generalization error rate $\hat{\varepsilon}(S)$ as the average of the ten separate error rates. We select the model with the lowest value of $\hat{\varepsilon}(S)$, where S ranges over all subsets of variables under consideration. Another common method is five-fold CV. The lower the number of folds, the less computing time is required, but the higher the variability of the estimates of the generalization error. Note that $n-$fold CV is the same as the computationally infeasible leave-one-out CV.

General risk minimisation (Vapnik, 1982) is derived from the estimated generalization error rate, using

$$GRM(S) = \hat{\varepsilon}(S) + \frac{|S|}{n}\big(1+\sqrt{1+\hat{\varepsilon}(S)(n/|S|)}\big). \tag{4}$$

Here, $|S|$ stands for the number of input variables in the set S and n is the number of observations in the training sample. We select the model with the lowest value of $GRM(S)$, where S ranges over all subsets of variables under consideration. Kearns et al. (1997) compare CV, GRM and minimum description length (Rissanen, 1989). Their experiments have demonstrated that none of the criteria is consistently better than the others. Note that the computational overhead for computing these measures can be immense, since we need to train ten support vector machines to estimate the generalization error rate for only one submodel.

We now define the KRIC of Kobayashi and Komaki (2006). This criterion was originally developed to tune the constant C in the SVM definition (1), and by extension to tune the kernel parameters. We use it without much adjustment for variable selection. Denote by $x_{i,S}$ the subvector of x_i, consisting of elements x_{ij} with $j \in S$, and similarly for other vectors. We estimate the SVM (1) using the observations $(x_{i,S}, y_i)$, yielding the vectors ω_S, b_S and ξ_S, where the subscript S refers to the subset of variables under consideration. In the dual problem (2), we have $\alpha_S = (\alpha_{S,1}, \ldots, \alpha_{S,n})$ and $[Q_S]_{i,k} = y_i y_k K(x_{i,S}, x_{k,S})$. The decision rule $f_S(x)$ is as in (3), and we set $a_{i,S} = f_S(x_{i,S})$. Next, we define vectors t_S and m_S of length n, with components

$$t_{S,i} = \eta^2 \frac{\exp(-\eta a_{i,S} y_i)}{(1+\exp(-\eta a_{i,S} y_i))^2} \quad \text{and} \quad m_{S,i} = -\eta \frac{y_i \exp(-\eta a_{i,S} y_i)}{1+\exp(-\eta a_{i,S} y_i)}, \;\; i = 1, \ldots, n.$$

Here we choose $\eta = \log(2)$ such that $\log(1+\exp(-\eta x))$ and $\eta[1-x]_+$ coincide for $x = 0$, see Kobayashi and Komaki (2006) for further motivation. With $\lambda = C^{-1}\log 2$ the KRIC for the logistic Bayesian model for SVMs is defined as

$$\begin{aligned} \mathrm{KRIC}(S) \;=\; & 2\Bigg[\sum_{i=1}^{n} \log\big(1+\exp(-\eta a_{i,S} y_i)\big) \\ & + \mathrm{trace}\big((Q_S \mathrm{diag}(t_S) + \lambda I_n)^{-1} Q_S (\mathrm{diag}(m_S)^2 - n^{-1} m_S m_S^t)\big)\Bigg]. \end{aligned} \tag{5}$$

Alternatively, Sollich's Bayesian model for SVMs (Sollich, 2002) leads to a KRIC with a similar form as the one in (5). Using

$$\nu(a_{i,S}) = (1+\exp(-2C))^{-1}(\exp(-C[1-a_{i,S}]_+)+\exp(-C[1+a_{i,S}]_+)),$$

the KRIC for the Sollich Bayesian model for SVMs is defined as

$$\mathrm{KRICS}(S) = \mathrm{KRIC}(S) - 2n\log\sum_{i=1}^{n}\nu(a_{i,S}). \tag{6}$$

The computation of the KRIC includes inverting an $n \times n$-matrix with only a few zeroes. Therefore, the computation is time-consuming if the sample size n is large. Both the CV error rate and the KRIC may require a prohibitive computing time when a large number of different models needs to be evaluated.

2.3 Ranking Techniques

A full subset search is computationally not feasible even not for problems with only a small number of dimensions ($p = 15$ for example). To dramatically reduce the number of models while still selecting a model that is "almost" the best model, Chen et al. (2005) use a genetic algorithm, while Peng et al. (2005) suggest a combined backward elimination/forward selection strategy. However, both of these techniques still suffer from the possibility that a large number of models needs to be checked before arriving at a solution.

Alternatively, variable ranking consists of assigning a "value of importance" to each variable and sorting the variables according to their importance. This results in a series of p stacked models, thus only p evaluations of the variable selection criterion are needed. The most commonly used algorithm is the SVM recursive feature elimination (SVM-RFE) technique from Guyon et al. (2002). For a linear SVM, the variables are ranked by w_j^2, with w_j the j-th component of the weight vector w. This technique assumes that the variables are standardized to have mean 0 and variance 1. The extension proposed by Rakotomamonjy (2003) allows application to SVMs with a non-linear kernel. We use the following SVM-RFE algorithm with variable influence

$$\Delta\|w_S\|^2_{(j)} = \left|\|w_S\|^2 - \|w_{S\setminus\{j\}}\|^2\right|$$

as suggested by Rakotomamonjy (2003).

Step 1: Initialise $S \leftarrow \{1,\ldots,p\}$, the subset of unranked features, and $r \leftarrow ()$, the vector of ranked features.

Step 2: Repeat the following steps until $S = \emptyset$.

(a) Train a SVM on $(x_{i,S}, y_i)$, and compute $\|w_S\|^2 = \alpha_S' Q_S \alpha_S$.

(b) For each $j \in S$, train a new SVM on $(x_{i,S\setminus\{j\}}, y_i)$. This gives a value $\|w_{S\setminus\{j\}}\|^2 = \alpha'_{S\setminus\{j\}} Q_{S\setminus\{j\}} \alpha_{S\setminus\{j\}}$ for each $j \in S$.

(c) Obtain $j_0 = \mathrm{argmin}_j \left|\|w_S\|^2 - \|w_{S\setminus(j)}\|^2\right|$ and set $S \leftarrow S\setminus\{j_0\}$ and $r \leftarrow (j_0, r)$.

The vector r contains the ranked variables, with the first element the most important one. A disadvantage of this method is that the number of SVMs to be trained is $O(p^2)$. This can be overcome by using α_S instead of $\alpha_{S\setminus\{j\}}$ in Step 2*b*, such that $\|w_{S\setminus\{j\}}\|^2 \approx \alpha_S' Q_{S\setminus\{j\}} \alpha_S$. Rakotomamonjy (2003) argues that this will not affect the ranking significantly, while still allowing a major reduction in computational time, bringing the number of SVMs to be estimated to $O(p)$. We employ this approximation in the simulation study in Section 4 and in the real data examples in Section 5.

The most easy way to rank the variables is by filtering methods. Zhang et al. (2006) propose using $s_j = |w_j(m_{j,+1} - m_{j,-1})|$ for ranking, where $m_{j,+1}$ and $m_{j,-1}$ are the within-class means of variable j. Shih and Cheng (2005) use the Fisher score

$$S_j = \frac{|m_{j,+1} - m_{j,-1}|}{\sqrt{\sigma^2_{j,+1} + \sigma^2_{j,-1}}}$$

for a linear SVM, where $\sigma^2_{j,+1}$ and $\sigma^2_{j,-1}$ are the within-class variances of variable j. The main advantage of using S_j is that it is not necessary to train any SVM to rank the variables. The Fisher score ranking is considered in Sections 4 and 5.

3. The New Information Criteria

As stated in the previous section, evaluating the CV error rate or the KRIC of a particular support vector machine model requires a high number of additional computations. For this reason, we propose two new criteria which use information already available in the SVM, without additional complicated computations. The criteria are based on how badly the SVM violates the margin constraints, which are written as $\sum_{i=1}^n \xi_{i,S}$, where $\xi_{i,S}$ is the margin slack of observation i in the support vector machine trained on the variables with indices in S, where S is a subset of $\{1,\ldots,p\}$. Alternatively, we can use the logarithm of this sum, analogous to Bai and Ng (2002) for selecting the number of factors in factor analysis. However, in the SVM setting this has the drawback that the value is undefined if the sum equals zero, which can happen if the data are perfectly separable. Also, Bai and Ng (2002) advise using a log-transform for scalar invariance reasons. Since we follow the advice to standardise the variables before training the SVM, for better ranking as explained in Section 2.3, we automatically have scalar invariance of the sum of the margin slacks. For these reasons, we choose not to take the log-transform.

Generally (but not always), $\sum_i \xi_{i,S}$ will decrease as more variables are added. Therefore we add a penalty term related to the number of included variables to ensure a tradeoff between accuracy and simplicity of the chosen model. We suggest adding a linear penalty term, such that we get an information criterion of the form

$$IC(S) = \sum_{i=1}^n \xi_i + C(n)|S|\,, \tag{7}$$

where S is the set of variables included in the model.

A first choice is to take $C(n)$ constant in (7). It is interesting to note that $IC(S)$ is then, up to constant factors, an easily computable approximation of the KRIC of Kobayashi and Komaki (2006), hereby providing a theoretical justification for its use. To better understand this, note first that $\log\left(1+\exp(-\eta a_{i,S} y_i)\right)$ is a continuous approximation of the hinge loss function

$\eta[1-y_i a_{i,S}]_+ = \eta\xi_{i,S}$ for all $1 \le i \le n$. Hence, the first term of the KRIC can be approximated, up to a constant factor, by $\sum_i \xi_{i,S}$. For the approximation of the second term in (5), rewrite

$$\begin{aligned} W &= (Q_S \mathrm{diag}(t_S) + \lambda I_n)^{-1} Q_S (\mathrm{diag}(m_S)^2 - n^{-1} m_S m_S^t) \\ &= V \mathrm{diag}(t_S)^{-1} (\mathrm{diag}(m_S)^2 - n^{-1} m_S m_S^t), \end{aligned}$$

with $V = (A + \lambda I_n)^{-1} A$ a symmetric, positive semi-definite matrix and $A = Q_S \mathrm{diag}(t_S)$. Denoting A^- the generalised inverse of A, and using a series expansion around $\lambda = 0$, gives that the leading term of $V = A^-(I + \lambda A^-)^{-1} A$ is equal to $A^- A$. This expansion converges as long as the eigenvalues of λA^- are strictly less than one, which can be obtained by taking λ small enough. We now use a singular value decomposition of both A and A^- and use the fact that the singular values of A^- are the reciprocals of the non-zero singular values of A, to obtain that the product $A^- A$ is a $n \times n$ diagonal matrix with on the diagonal $|S|$ ones and the remaining entries zero. Thus, the leading term of $\mathrm{trace}(W)$ equals the sum of $|S|$ diagonal entries of the matrix $\mathrm{diag}(t_S)^{-1}(\mathrm{diag}(m_S)^2 - n^{-1} m_S m_S^t))$. The i-th diagonal element of this matrix is equal to

$$\frac{n-1}{n} t_{S,i}^{-1} m_{S,i}^2 = \frac{n-1}{n} \exp(-\eta a_{i,S} y_i).$$

To further facilitate computations we replace this by 1, motivated by the fact that $\eta a_{i,S} y_i$ is often small. Although this approximation might be crude for a single term, we found empirically that it works well for the summation over the entire training set. Hence, we arrive at the approximation $\mathrm{trace}(W) \approx |S|$ which is the linear penalty term in (7).

Taking the constant value $C(n) = 2$, leads to our first new support vector machine information criterion (SVMIC):

$$\mathrm{SVMICa}(S) = \sum_{i=1}^{n} \xi_i + 2|S|. \tag{8}$$

The newly proposed criterion SVMICa for support vector machines shares the form of the penalty with the well-known Akaike (1973) information criterion. This AIC is defined as minus twice the value of the maximised log likelihood of the model, plus two times the number of parameters to be estimated (that is, $2|S|$). Because the penalty $2|S|$ is not dependent on the sample size n, we expect that both criteria share some properties, such as having the tendency to not select the most parsimonious model. For the AIC, Woodroofe (1982) has shown that in the limit for $n \to \infty$, the expected number of superfluous parameters is less than one.

To support the definition of SVMICa , we ran a simulation experiment and compared the values of KRIC and SVMICa for 100 models. The sample size is $n = 50$, with 10 variables of which only the first 4 variables are different from zero. A detailed description of the simulation setting can be found in Section 4. We used a linear kernel. Figure 1 reports these numerical results and shows a high correlation (0.975) between the values of the two criteria. Other simulation settings gave comparable correlation values.

Our second proposed criterion follows the spirit of Bayesian information criterion (BIC) by Schwarz (1978). This criterion is defined similarly as the AIC, but instead of the penalty $2|S|$, it uses $\log(n)|S|$. The BIC has been shown to be consistent (Haughton, 1988, 1989). This means that if the true model is contained in the search list, the criterion will (in the limit for $n \to \infty$) select this correct model. For a related construction for factor models, see Bai and Ng (2002). This motivates

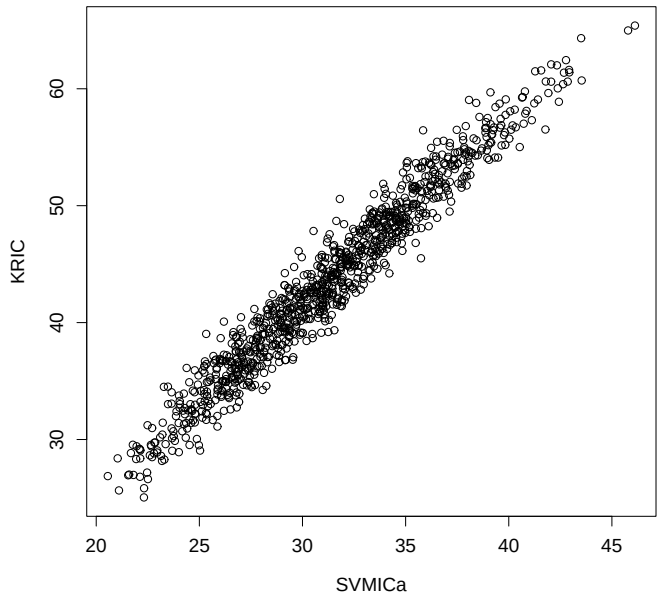


Figure 1: Values of KRIC and SVMICa in a simulation experiment, showing high correlation (0.975).

us to take $C(n) = \log(n)$, and we define our second criterion

$$\text{SVMICb}(S) = \sum_{i=1}^{n} \xi_i + \log(n)|S|. \tag{9}$$

It is immediate that the computational cost of both SVMICs is much lower than of the cross-validated error rate (10 more SVMs to train for 10-fold cross-validation) and of the kernel regularisation information criterion KRIC (which needs computations of the order $O(n^3)$ due to the matrix inversion). The best case is when the $\xi_{i,S}$ are directly available. Computing the SVMICs is only an $O(n)$ computation in that case, and usually even less when employing the property that

$$\xi_{i,S} \neq 0 \Leftrightarrow \alpha_{i,S} = 1.$$

When only α_S and Q_S are available, $\xi_{i,S}$ is computed using the relation

$$\xi_{i,S} = \Big[1 - y_i \sum_{\substack{j=1 \\ \alpha_{j,S}>0}}^{n} \alpha_{j,S}[Q_S]_{ij}\Big]_+.$$

This means that in the worst case, the computation time of the *SVMIC*s is $O(n^2)$, which is still faster than using either CV error rate or KRIC.

4. Simulation Results

We perform $M = 100$ simulation runs with the following settings. We generate $n \in \{25, 50, 100, 200\}$ independent observations x_i, $1 \leq i \leq n$ of dimension $p \in \{25, 50, 100, 200\}$, with distribution $\mathcal{N}(0, \sigma^2 I_p)$ where $\sigma^2 = 1$. For each observation we generate a class label $y_i \in \{-1, +1\}$, with $P(y_i = 1) = 1/2$. Finally, we let $\mu = (1/2, -1/2, -1/2, 1/2, 0, \ldots, 0)$ of dimension p, and set $x_i \leftarrow x_i + y_i \mu$ to separate the two classes to some extent. This implies that the optimal separating hyperplane is $x'\mu = 0$, such that $\hat{y} = +1$ if $x'\mu > 0$, resulting in a generalization error rate of $\Phi(-\|\mu\|_2/\sigma)$, with Φ the cumulative distribution function of a standard normal. In our example, with $\sigma = 1$ and $\|\mu\|_2 = 1$, we find an optimal generalization error rate of 0.159.

During each simulation run, we standardize the variables to improve the numerical performance of the SVM algorithm. The variables are ranked using either the Fisher score or based on the variable influence on w, as described in Section 2.3. For each of the nested models obtained in the variable ranking step, we compute (i) SVMICa and (ii) SVMICb as in (8) and (9). We compare their performance to (iii) ten-fold CV, (iv) Vapnik's GRM as in (4), (v) KRIC for the logistic Bayesian model for SVMs as in (5), and (vi) KRIC for the Sollich model for SVMs as in (6). An important remark is that for ten-fold CV, we employ the CV2 method, which includes the feature selection procedure in each cross-validation step, as suggested by Zhang et al. (2006). Computing the CV error rate in the usual way can lead to a (severely) biased estimate of the generalization error, and using CV2 reduces this bias.

The experiment is repeated with two different kernels (i) a linear kernel $K(x_1, x_2) = x_1' x_2$ leading to a linear decision rule (ii) a quadratic kernel $K(x_1, x_2) = (\gamma x_1' x_2 + 1)^2$, with $\gamma = 1/p$, the inverse of the number of variables, leading to a quadratic decision rule. The tuning parameter C in each SVM that we train is chosen to be $C = 1$, as we standardize the explicative variables a priori. This is also the standard setting for C for the `svm` procedure in the `R` software package. We experimented with other values of C in the range from 0.1 up to 10, and found only minor differences in the simulation outcomes. We test the accuracy of the classifiers computed from the selected input variables by estimating their generalization (out-of-sample) error rate from a test sample of 10000 new observations. These observations are generated in the same way as the training sample.

Table 1 reports the generalization error rates, obtained by averaging over the 100 simulation runs. An overall observation is that the error-rate based selection criteria (CV and GRM) have the worst performance. The performances of the KRICs and the new SVMICs are comparable. More precisely, we observe that the KRICs are better as a variable selection method for small sample sizes ($n = 25$), while the SVMICs give better results for larger sample sizes. This is especially apparent when the quadratic kernel is used. For a small number of observations compared to the number of variables, we also note that SVMICa slightly outperforms SVMICb in terms of generalization error rate, and that the opposite is true with many observations and fewer variables. The differences in generalization error rates become smaller as the number of variables grows. This is particularly true for CV, whose relative performance becomes better at large sample sizes. But SVMICa and SVMICb are still somewhat ahead, and have the advantage that they are much easier (and less time-intensive) to compute than the other criteria, included the KRICs having a computation time of order $O(n^3)$. Note that, as n grows, the generalization error rates of the models obtained by our two suggested criteria are converging towards the theoretically obtained minimal generalization error rate of 15.9%. Investigating which variable ranking criterion is better, results in case of linear

Linear kernel													
n	p	SVMICa		SVMICb		CV		GRM		KRIC		KRICS	
25	25	32.2	29.4	32.6	31.6	33.5	31.8	36.2	34.5	31.3	29.0	31.5	29.9
	50	34.6	31.6	35.3	32.6	35.3	33.5	37.4	35.4	34.4	33.2	34.4	33.2
	100	37.4	33.9	37.3	35.0	37.8	34.4	38.6	35.7	37.0	34.9	37.1	34.9
50	25	24.4	21.6	24.6	23.2	27.1	25.5	31.1	29.6	25.7	24.9	26.0	25.9
	50	28.5	23.3	27.7	24.8	29.5	26.3	31.4	30.5	29.8	28.7	30.2	29.7
	100	30.9	24.6	29.1	25.0	31.0	28.0	32.1	30.9	31.0	30.1	31.3	30.8
100	25	19.9	18.5	19.6	18.9	24.6	23.8	30.1	30.1	21.8	20.6	22.3	21.7
	50	22.9	19.2	20.2	19.0	25.8	25.4	29.9	29.6	26.9	26.8	27.3	27.8
200	25	17.8	17.0	16.9	16.8	22.7	21.5	28.9	29.3	18.7	18.0	19.2	18.9
Quadratic kernel													
n	p	SVMICa		SVMICb		CV		GRM		KRIC		KRICS	
25	25	31.3	30.7	34.2	33.8	33.8	32.9	37.7	36.6	29.5	28.4	30.2	30.1
	50	35.8	35.3	39.3	38.5	39.6	38.5	43.6	42.6	33.3	33.0	33.9	34.1
	100	43.3	43.3	48.3	48.4	42.8	42.7	49.2	48.7	37.1	37.1	37.7	38.2
50	25	22.7	21.3	25.0	24.3	26.7	25.9	31.8	31.7	23.6	22.5	24.8	25.1
	50	24.4	23.0	26.8	26.8	29.8	28.1	33.9	33.5	27.6	27.1	29.1	29.3
	100	26.4	25.6	30.8	30.2	34.1	33.8	40.3	40.1	31.1	30.9	32.5	32.8
100	25	19.4	18.5	19.9	19.1	23.8	19.2	30.6	30.2	20.0	20.0	21.7	22.0
	50	19.7	18.5	19.8	19.5	24.2	22.0	30.5	30.7	22.6	22.6	24.7	25.1
200	25	20.1	20.3	17.1	16.8	22.4	21.4	29.4	29.6	18.3	18.1	20.3	20.6

Table 1: Simulated average generalization error rate (%) for the six methods using two different kernels. For each method, the number on the left resulted from ranking by variable influence on $\|w\|^2$, and the number on the right in each column is from ranking by the Fisher scores S_j.

kernels to a strong preference for ranking with the Fisher score. For the quadratic kernel, it is slightly better to rank the variables based on variable influence on $\|w\|^2$.

Figure 2 presents the values of the 100 simulated generalization errors as boxplots, giving insight in the variability of the variable selection methods. For most of the cases it turns out that cross-validation is highly variable, while GRM has a small variability. This good property of GRM is, however, accompanied by a much higher average generalization error rate. Comparing the different information criteria shows that SVMICa is quite comparable to the KRICs. The SVMICb has a larger variability. In the setting with small sample size ($n = 25$) and relatively large number of variables (100), all methods, except for GRM, are comparable with respect to variability, but GRM has again the largest median error rate. Our main conclusion from this analysis is that SVMICa has a similar variability than the KRIC criteria, but SVMICb has a larger variability. Recall that the average error rates, as reported in Table 1, were of similar magnitude for all the four information criteria. Hence, when needing to choosing between the two newly proposed information criteria, we have a preference for SVMICa.

Given the variability of the generalization errors over the 100 simulation runs, see the boxplots in Figure 2, it is important to test whether the averages reported in Table 1 are also significantly different from each other. We performed standard t-tests, and most difference are indeed significant. For example, for the settings presented in Figure 1, we obtained that, at the 1% level, (a) all differences are significant, except between SVMICb and the 2 KRiCs (b) all differences are significant, except between SVMICa and the 2 KRICs (c) all differences are significant, except between SVMICb and the 2 KRICs (d) the differences with the GRM method are significant, the others not.

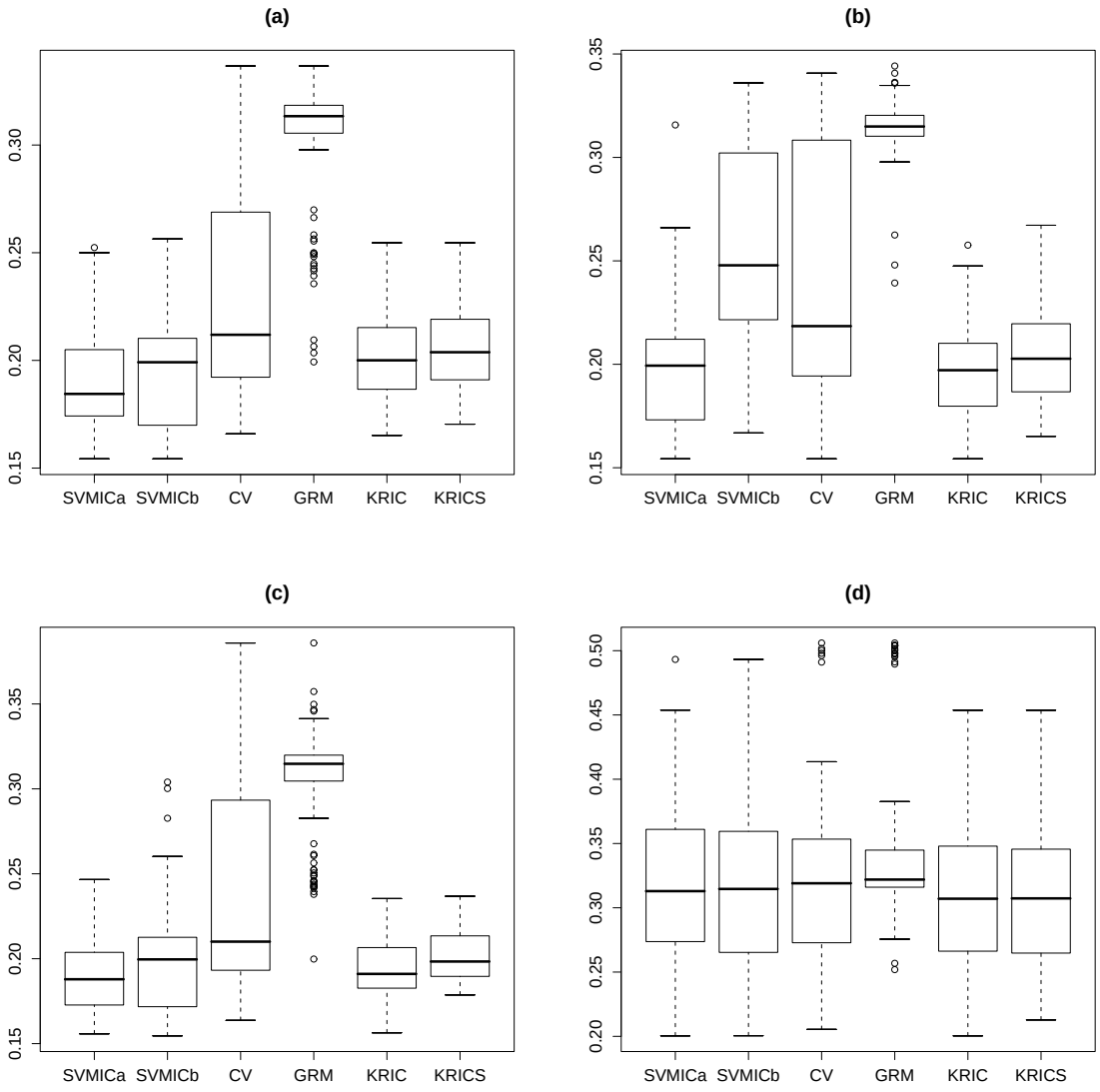


Figure 2: Generalization error rates for 100 simulation experiments, for $n = 100$, $p = 25$ (a) linear kernel, ranking with $\|w\|^2$, (b) linear kernel, ranking with Fisher score, (c) quadratic kernel, ranking with $\|w\|^2$, and for (d) $n = 25$, 100 variables, linear kernel and ranking with $\|w\|^2$.

Furthermore, we investigate which models are actually chosen by the different criteria. This information is reported in Table 2. For each setting, it shows how many times the correct subset of input variables, containing only the first four input variables, was chosen (C, correct). This table also shows how many times a too-sparse group of variables was selected (U, underfitting), and how many times a too-rich group of variables was chosen (O, overfitting). So an overfit means that all correct variables are selected, but in addition some superfluous ones, while an underfit selects a

	Kernel:	Linear				Quadratic			
	Models selected:	C	U	O	R	C	U	O	R
$n = 25$; $p = 25$	SVMICa	1	22	1	76	3	36	0	61
	SVMICb	0	42	0	58	0	64	0	36
	CV	0	38	4	58	1	40	5	54
	GRM	0	77	0	23	0	75	0	25
	KRIC	1	1	7	91	0	1	25	74
	KRICS	0	0	9	91	0	0	49	51
$n = 200$; $p = 25$	SVMICa	22	0	76	2	2	0	98	0
	SVMICb	77	9	10	4	67	14	6	13
	CV	7	48	43	2	4	43	49	4
	GRM	1	98	1	0	1	99	0	0
	KRIC	6	0	93	1	8	0	84	8
	KRICS	1	0	99	0	0	0	100	0
$n = 25$; $p = 100$	SVMICa	0	8	0	92	0	35	0	65
	SVMICb	0	20	0	80	0	63	0	37
	CV	0	23	6	71	0	33	10	57
	GRM	0	56	0	44	0	64	0	36
	KRIC	0	1	0	99	0	0	41	59
	KRICS	0	0	1	99	0	0	56	44

Table 2: Simulated frequencies of selected models, with variable ranking done by influence on $\|w\|^2$. Here 'C' denotes correct selection, 'U' is underfitting, 'O' is overfitting, and 'R' for all other situations.

subset of the important variables, but no irrelevant variables are included. The good performance of SVMICa and SVMICb might be due to the fact that these criteria seem to have the tendency to select a set of variables which includes all significant ones as the number of observations grows. The simulation results indicate that SVMICa behaves like AIC with its tendency to overfit. The SVMICb seems to share the property of BIC that it selects the correct model more often, if at least this true model is one of the possibilities to select from. The cross-validated error rate, and the general risk minimisation in particular, seem to have the tendency to ignore variables which nevertheless are important. As a consequence, the models that these criteria select are of poor predictive quality. The two KRICs of Kobayashi and Komaki (2006) share the overselection property exhibited by SVMICa, but the KRICs select excessive variables even more frequently than SVMICa. This can explain why these criteria perform somewhat worse when the number of observations is large, and why they outperform the proposed SVMICs when the number of observations is small, since the latter tend to underfit the model in the case of few observations.

This concludes the results for the case of two populations coming from an identical distribution, differing only in mean. Another case that we examined is where the variances of the two populations differ from each other. We performed a simulation study, in a similar way as the previous one, where the samples have been drawn from $\mathcal{N}(\mu, I_p)$ for class $+1$, and from $\mathcal{N}(-2\mu, 4I_p)$ for class -1.

The results of this simulation are summarized in Tables 3 and Table 4. We observe similar results as in the case where both populations had equal variance. Selection based on CV error rate and on GRM still perform rather poor. As before, the performances of the KRICs and SVMICs

Linear kernel													
n	p	SVMICa		SVMICb		CV		GRM		KRIC		KRICS	
25	25	28.9	28.0	30.1	29.2	30.4	28.4	32.7	31.6	29.0	27.5	28.8	27.7
	50	33.3	30.2	34.2	31.3	35.1	31.4	35.3	33.1	32.7	30.7	32.5	30.5
	100	35.6	31.5	35.7	32.3	36.0	32.6	36.9	33.7	34.8	32.6	34.8	33.0
	200	36.5	33.2	36.4	34.4	36.4	34.2	36.6	35.6	36.4	33.5	36.1	33.7
50	25	23.3	20.5	23.9	21.9	26.1	24.9	28.9	28.6	24.2	23.6	24.6	24.3
	50	27.1	21.7	25.7	22.7	27.7	25.2	29.1	28.4	27.7	26.8	27.6	27.1
	100	28.3	23.1	27.4	23.7	28.7	25.2	29.9	28.7	28.4	26.7	28.4	27.5
100	25	19.0	17.4	18.1	17.4	22.7	21.5	27.6	27.6	20.5	20.0	21.0	20.9
	50	21.8	17.8	19.3	18.0	23.5	22.7	26.9	27.0	24.8	25.0	25.0	25.5
200	25	17.0	16.1	15.9	15.6	21.4	20.7	27.0	27.0	17.9	17.0	18.3	17.8
Quadratic kernel													
n	p	SVMICa		SVMICb		CV		GRM		KRIC		KRICS	
25	25	29.2	28.9	31.8	31.8	31.8	28.7	35.4	34.7	25.7	24.9	25.8	26.2
	50	35.1	35.8	39.6	40.0	38.1	37.6	42.8	42.4	30.5	30.8	31.3	32.3
	100	42.1	41.7	48.2	48.1	42.2	42.3	49.4	48.7	35.0	36.0	36.2	38.1
	200	50.1	50.1	50.1	50.1	44.7	44.4	50.1	50.1	38.9	40.0	40.4	41.8
50	25	20.5	19.3	23.5	22.2	25.9	24.5	30.6	30.2	19.0	19.1	19.5	19.9
	50	23.1	22.2	26.1	26.2	28.3	27.6	33.2	32.7	23.8	23.9	25.1	26.1
	100	26.5	25.8	30.4	30.4	34.5	33.7	40.5	40.4	28.2	28.8	30.1	32.3
100	25	14.6	15.2	18.5	16.4	20.8	19.9	27.8	27.1	14.2	14.5	14.5	14.9
	50	17.9	17.0	18.4	17.8	22.0	21.5	27.7	28.3	18.1	18.5	19.5	20.3
200	25	9.9	9.8	12.9	13.2	19.6	17.6	29.3	26.8	10.1	10.3	9.7	9.8

Table 3: As Table 1, but now for two populations with different variances.

	Kernel:	Linear				Quadratic			
	Models selected:	C	U	O	R	C	U	O	R
$n = 25$; $p = 25$	SVMICa	0	22	1	77	1	36	0	63
	SVMICb	0	47	0	53	1	57	0	42
	CV	1	40	1	58	1	39	8	52
	GRM	0	76	0	24	0	70	0	30
	KRIC	0	0	6	94	0	0	25	75
	KRICS	0	0	8	92	0	0	50	50
$n = 200$; $p = 25$	SVMICa	11	0	85	4	0	20	0	80
	SVMICb	69	10	16	5	0	45	0	55
	CV	6	56	37	1	0	33	4	63
	GRM	0	100	0	0	0	56	0	44
	KRIC	5	0	93	2	0	0	40	60
	KRICS	0	0	99	1	0	0	53	47
$n = 25$; $p = 200$	SVMICa	0	1	0	99	0	52	0	48
	SVMICb	0	8	0	92	0	54	0	46
	CV	0	22	2	76	0	22	5	73
	GRM	0	46	0	54	0	54	0	46
	KRIC	0	1	0	99	0	0	46	54
	KRICS	0	0	0	100	0	0	56	44

Table 4: As Table 2, but now for two populations with different variances.

are similar. More precisely, the SVMICs have an improved performance with respect to the KRICs when the sample size is large ($n \geq 50$) and the linear kernel is used, and the KRICs work slightly better for small sample sizes ($n = 25$). For the quadratic kernel, we notice a good performance of

the KRICs, which is only matched by SVMICa for larger sample sizes. From Table 4 we can again make the same observations as before when the linear kernel is used. For the quadratic kernel the SVMICs have more difficulty selecting all the relevant variables than the KRICs, which explains why the latter criteria have an improved performance here.

We also conducted a simulation experiment where the input variables were strongly correlated. First, the observations were generated as in the first simulation experiment. Then, we applied the transformation

$$x_{ij} = \rho x_{ik_j} + \varepsilon_{ij} \text{ with } \varepsilon_{ij} \sim \mathcal{N}(0, \rho^2) \text{ i.i.d.}$$

where $i = 1, \ldots, n$, k_j is chosen arbitrarily between 1 and 4, and $4 < j \leq p/2$, such that about half of the unimportant input variables are correlated with the four important ones. The parameter $|\rho| < 1$ controls the degree of correlation. We have chosen $\rho = 0.8$ and found similar results (not reported) as for the case where the variances of both class-population differ.

5. Tests on Real Data Sets

We compare the performance of the new methods with that of the other discussed criteria on several real-world data sets. We use some of the benchmark data sets used in Rakotomamonjy (2003), and in Rätsch et al. (2001). The data sets used are the Pima Indians Diabetes database (768 observations, 8 variables), the Statlog Cleveland Heart Disease database (303 observations, 14 variables), and Leo Breiman's ringnorm and twonorm data sets (both 7400 observations, 20 variables). These data sets are available from the UCI Machine Learning Repository (the first two), and the Delve Repository (last two). We perform 100 random splits of the data in a training sample and a test sample, where the size of the training sample is chosen as $\sqrt{2n}$, with n the total number of observations in the data set. We chose the size of the training set such that there is a sufficient amount of observations in the test sample to estimate the generalization (out-of-sample) error rate. The training sample size is relatively small, such that the computation time for the KRIC remains within bounds. For each of these partitions we perform variable selection on the training sample exactly as in the simulation study. We first rank the variables to retain p stacked subsets of input variables, and then use the information criteria to select the variables that best explain the training data. Then, we predict the class labels for the test sample, and use these predictions to estimate the generalization error rate. We use variable ranking based on variable influence on $\|w\|^2$ as well as on Fisher score, and we use a linear, quadratic and radial kernel.

The estimated generalization error rates are presented in Table 5 for each data set and estimation setting. We observe that the KRICs are the preferred choice of variable selection criterion in terms of generalization error rate for the 'twonorm' and 'heart' data sets. For the 'ringnorm' and 'diabetes' data sets the difference in performance between the KRICs and our newly proposed SVMICs is less pronounced. The predictive performance of the models selected by SVMICa are for most settings comparable to that of the KRIC, while being much faster to compute. These results are consistent across all settings. The CV error rate and especially the GRM have a poor performance, which is in line of the results obtained in the simulation.

From these results, and the results obtained in Section 4, we suggest to use either the SVMICa or the SVMICb if a preliminary analysis of the data or a priori knowledge indicates that the true decision function is almost linear. When it differs strongly from a linear function, the researcher has a choice between the ease of computation of the support vector machine information criteria, or the

	Ranking:	Variable influence on $\|\|w\|\|$			Fisher scores		
Data	Kernel:	Linear	Quadratic	Radial	Linear	Quadratic	Radial
Diabetes	SVMICa	28.6	28.5	29.2	28.0	28.2	28.4
	SVMICb	29.0	28.9	29.2	28.6	28.5	28.9
	CV	28.6	29.1	29.1	28.8	28.5	29.3
	GRM	29.6	29.7	29.6	29.1	29.2	29.3
	KRIC	28.5	28.2	29.4	27.5	28.1	29.6
	KRICS	28.6	28.5	29.7	28.3	28.6	29.7
Heart	SVMICa	27.0	27.4	27.7	27.6	28.0	28.3
	SVMICb	27.6	28.9	28.9	28.2	29.3	29.5
	CV	27.6	28.6	27.2	26.8	28.0	28.8
	GRM	29.3	30.3	29.4	28.8	30.4	30.6
	KRIC	25.4	23.4	23.8	24.5	23.2	23.8
	KRICS	25.3	23.5	25.2	25.2	23.7	25.0
Ringnorm	SVMICa	31.1	16.4	8.4	30.8	15.6	6.5
	SVMICb	34.9	20.2	13.5	35.2	22.4	13.4
	CV	33.9	32.1	26.6	32.8	25.6	21.2
	GRM	39.2	41.3	38.6	39.3	38.4	37.3
	KRIC	30.1	16.3	6.0	29.6	15.9	4.4
	KRICS	29.9	16.0	3.1	29.2	15.4	2.5
Twonorm	SVMICa	9.9	9.3	11.4	10.1	8.9	9.4
	SVMICb	13.5	14.1	15.9	15.0	15.2	16.0
	CV	20.5	21.0	19.8	21.0	21.1	20.8
	GRM	31.4	31.7	31.6	30.8	31.2	31.3
	KRIC	8.0	7.5	11.0	6.8	6.8	9.2
	KRICS	7.5	6.0	4.0	6.6	5.5	4.8

Table 5: Generalization error rates (%) for variable selection applied to four data sets. Two variable ranking schemes and three types of kernel are used for each of the criteria.

somewhat improved predictive performance, though with higher computational cost, of the kernel regularization information criterion.

Finally, we applied the newly proposed information criteria for variable selection to two large data sets, the "Madelon" ($n = 2000, p = 500$) and "Arcene" data ($n = 100, p = 10000$). These data sets were part of the NIPS 2003 feature selection, and are described in detail in Guyon et al. (2006). Given the high dimensionality of these data, the variables were ranked according to the Fisher score. We used a linear kernel and computed balanced error rates (BER), that is the average of the error rate of the positive class and the error rate of the negative class. When using SVMICa we obtain a BER of 43.0% for the Madelon data, and 31.1% for the Arcene data. For SVMICb we get 37.3% and 31.1%, respectively. In Guyon et al. (2006, 2007) the BER of other feature selection methods is presented, and it turns out that several other methods yield much better performance on these data. A possible explication is that we used a standard SVM, without any optimal tuning of the regularization parameters.

6. Conclusions

In this paper we considered the problem of variable selection in support vector machines. We proposed two new information criteria, SVMICa and SVMICb, which allow us to evaluate the suitability of the selected subset of variables for predictive purposes, without much additional computational costs. We provided an argumentation for these criteria, linking SVMICa to the KRIC of Kobayashi and Komaki (2006), and justifying SVMICb with the need for a consistent selection criterion. We demonstrated the effectiveness of these criteria in a simulation study, where we compared their predictive performance to the KRIC, cross-validation and general risk minimization. Especially for decision functions which are close to an affine function, we found that SVMICa and SVMICb performed the best of all tested criteria, and were also the easiest to compute. For more complicated decision functions, we found that SVMICa still performs well for selecting models with good generalization properties. We repeated the experiment on several real data examples, and the result confirmed the good properties of these newly proposed criteria. In particular we showed that cross-validation criteria are outperformed in generalization error by the new information criteria, where the latter are coming at almost no additional computational cost.

The aim of our paper was to propose an information criterion for a standard SVM. We do not claim that the procedure is outperforming other very advanced feature selection methods, which are not relying on a standard SVM. Obtaining information criteria for other machine learners is an interesting topic for future research. Another research question is how suitable the information criteria are for optimal tuning of the regularization and other parameters of the SVM, without necessarily selecting a subset of input variables. Finally, it would be interesting to continue on the theoretical verification of the good performance of our two proposed criteria, and for example try to obtain consistency results for the SVM information criteria.

Acknowledgments

We thank the Editors, Professor Guyon and Professor Alamdari, and the reviewers for their constructive and insightful comments on the first submitted version of this paper.

References

H. Akaike. Information theory and an extension of the maximum likelihood principle. In B. Petrov and F. Csáki, editors, *Second International Symposium on Information Theory*, pages 267–281. Akadémiai Kiadó, Budapest, 1973.

J. Bai and S. Ng. Determining the number of factors in approximate factor models. *Econometrica*, 70:191–221, 2002.

J. Bi, K. P. Bennett, M. Emrechts, C. M. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3:1229–1243, 2003.

S.-W. Chen, Z.-R. Li, and X.-Y. Li. Prediction of antifungal activity by support vector machine approach. *Journal of molecular structure: THEOCHEM*, 731:73–81, 2005.

G. Claeskens and N. L. Hjort. *Model Selection and Model Averaging*. Cambridge University Press, Cambridge, 2008.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, 2000.

J. Fortuna and D. Capson. Improved support vector classification using PCA and ICA feature space modification. *Pattern Recognition*, 37:1117–1129, 2004.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh. *Feature Extraction, Foundations and Applications*. Physica-Verlag, Springer, Berlin, 2006.

I. Guyon, J. Li, T. Mader, P. A. Pletscher, G. Schneider, and M. Uhr. Competitive baseline methods set new standards for the NIPS 2003 feature selection benchmark. *Pattern Recognition Letters*, 28:1438–1444, 2007.

T. J. Hastie, R. J. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, Heidelberg, 2001.

D. Haughton. On the choice of a model to fit data from an exponential family. *The Annals of Statistics*, 16:342–355, 1988.

D. Haughton. Size of the error in the choice of a model to fit data from an exponential family. *Sankhyā, Series A*, 51:45–58, 1989.

M. Kearns, Y. Mansour, A. Y. NG, and D. Ron. Experimental and theoretical comparison of model selection methods. *Machine Learning*, 27:7–50, 1997.

K. Kobayashi and F. Komaki. Information criteria for support vector machines. *IEEE Transactions on Neural Networks*, 17:571–577, 2006.

Y. Lee, Y. Kim, S. Lee, and J.-Y. Koo. Structured multicategory support vector machines with analysis of variance decomposition. *Biometrika*, 93:555–571, 2006.

Y. Lin and H. H. Zhang. Component selection and smoothing in multivariate nonparametric regression. *Annals of Statistics*, 34:2272–2297, 2006.

J. Neumann, C. Schnörr, and G. Steidl. Combined SVM-based feature selection and classification. *Machine Learning*, 61:129–150, 2005.

H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 27:1226–1238, 2005.

A. Rakotomamonjy. Variable selection using SVM-based criteria. *Journal of Machine Learning Research*, 3:1357–1370, 2003.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Machine Learning*, 42:287–320, 2001.

J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Co. Inc., Teaneck, NJ, 1989.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, 2002.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

F. Y. Shih and S. Cheng. Improved feature reduction in input and feature spaces. *Pattern Recognition*, 38:651–659, 2005.

P. Sollich. Bayesian methods for support vector machines: evidence and predictive class probabilities. *Machine Learning*, 46:21–52, 2002.

V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer, New York, 1982.

L. Wang, J. Zhu, and H. Zou. The doubly regularized support vector machine. *Statistica Sinica*, 16: 589–615, 2006.

M. Woodroofe. On model selection and the arc sine laws. *The Annals of Statistics*, 10:1182–1194, 1982.

H. H. Zhang. Variable selection for SVM via smoothing spline ANOVA. *Statistica Sinica*, 16: 659–674, 2006.

X. Zhang, X. Lu, Q. Shi, X.-Q. Xu, H.-C. E. Leung, L. N. Harris, J. D. Iglehart, A. Miron, J. S. Liu, and W. H. Wong. Recursive SVM feature selection and sample classification for mass-spectrometry and microarray data. *BMC Bioinformatics*, 7:197, 2006.

J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. *Advances in Neural Information Processing Systems*, 16:49–56, 2004.

Journal of Machine Learning Research 9 (2008) 559-580 Submitted 12/06; Revised 11/07; Published 4/08

Closed Sets for Labeled Data

Gemma C. Garriga GEMMA.GARRIGA@HUT.FI
Helsinki Institute for Information Technology
Helsinki University of Technology
02015 Helsinki, Finland

Petra Kralj PETRA.KRALJ@IJS.SI
Nada Lavrač NADA.LAVRAC@IJS.SI
Department of Knowledge Technologies
Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia

Editor: Stefan Wrobel

Abstract

Closed sets have been proven successful in the context of compacted data representation for association rule learning. However, their use is mainly descriptive, dealing only with unlabeled data. This paper shows that when considering labeled data, closed sets can be adapted for classification and discrimination purposes by conveniently contrasting covering properties on positive and negative examples. We formally prove that these sets characterize the space of relevant combinations of features for discriminating the target class. In practice, identifying relevant/irrelevant combinations of features through closed sets is useful in many applications: to compact emerging patterns of typical descriptive mining applications, to reduce the number of essential rules in classification, and to efficiently learn subgroup descriptions, as demonstrated in real-life subgroup discovery experiments on a high dimensional microarray data set.

Keywords: rule relevancy, closed sets, ROC space, emerging patterns, essential rules, subgroup discovery

1. Introduction

Rule discovery in data mining mainly explores unlabeled data and the focus resides on finding itemsets that satisfy a minimum support constraint (namely frequent itemsets), and from them, constructing rules over a certain confidence. This is the case of the well-known Apriori algorithm of Agrawal et al. (1996), and its successors, for example, Brin et al. (1997), Han and Pei (2000) and Zaki (2000b) among others. From a different perspective, machine learning is mainly concerned with the analysis of class labeled data, mainly resulting in the induction of classification and prediction rules, and—more recently—also descriptive rules that aim at discovering insightful knowledge from the data (subgroup discovery, contrast set mining). Traditional rule learning algorithms for classification include CN2 (Clark and Niblett, 1989) and Ripper (Cohen, 1995). Other approaches have been proposed that are based on the association rule technology but applied to class labeled data, for example, a pioneer work towards this integration is Liu et al. (1998), and later followed by others, for example, the Apriori-C classifier by Jovanoski and Lavrač (2001), and the Essence algorithm for inducing "essential" classification rules based on the covering properties of frequent itemsets, by Baralis and Chiusano (2004).

Subgroup discovery is a learning task directed at finding subgroup descriptions that are characteristic for examples with a certain property (class) of interest. Special rule learning algorithms for subgroup discovery include Apriori-SD (Kavšek and Lavrač, 2006), CN2-SD (Lavrač et al., 2004) or SD (Gamberger and Lavrač, 2002). The goal of these descriptive mining algorithms is to find characteristic rules as combinations of features with high coverage. If there are several rules with the same coverage, most specific rules (with more features) are appropriate for description and explanation purposes. On the other hand, the closely related task of contrast set mining aims at capturing discriminating features that contrast instances between classes. Algorithms for contrast set mining are STUCCO (Bay and Pazzani, 2001), and also an innovative approach presented in the form of mining emerging patterns (Dong and Li, 1999). Basically, Emerging Patterns (EP) are sets of features in the data whose supports increase significantly from one class to another. Interestingly, also good classifiers can be constructed by using the discriminating power of the mined EPs, for example, see Li et al. (2000). A condensed representation of EPs, defined in terms of a support growth rate measure, has been studied in Soulet et al. (2004).

Indeed, we can see all these tasks on labeled data (learning classification rules, subgroup discovery, or contrast set mining) as a rule induction problem, that is, a process of searching a space of concept descriptions (hypotheses in the form of rule antecedents). Some descriptions in this hypothesis space may turn out to be more relevant than others for characterizing and/or discriminating the target class. The question of relevance has attracted much attention in the context of feature selection for propositional learning (Koller and Sahami, 1996; Liu and Motoda, 1998). This is an important problem since non-relevant features can be excluded from the learning process, thus facilitating the search for the final solution and increasing the quality of the final rules. Feature filtering can be applied during the learning process, or also, by pre-processing the set of training examples (Lavrač et al., 1999; Lavrač and Gamberger, 2005).

Searching for relevant descriptions for rule construction has been extensively addressed in descriptive data mining as well. A useful insight was provided by closure systems (Carpineto and Romano, 2004; Ganter and Wille, 1998), aimed at compacting the whole space of descriptions into a reduced system of relevant sets that formally conveys the same information as the complete space. The approach has successfully evolved towards mining closed itemsets (see, for example, Pasquier et al., 2001; Zaki, 2004). Intuitively, closed itemsets can be seen as maximal sets of items/features covering a maximal set of examples. Despite its success in the data mining community, the use of closed sets is mainly descriptive. For example, they can be used to limit the number of association rules produced without information loss (see, for example, how to characterize rules with respect to their antecedent in Crémilleux and Boulicaut, 2002).

To the best of our knowledge, the notion of closed sets has not yet been exported to labeled data, nor used in the learning tasks for labeled data described above. In this paper we show that raw closed sets can be adapted for discriminative purposes by conveniently contrasting covering properties on positive and negative examples. Moreover, by exploiting the structural properties and the feature relevancy theory of Lavrač et al. (1999) and Lavrač and Gamberger (2005), we formally justify that the obtained closed sets characterize the space of relevant combinations of features for discriminating the target class.

In practice, our notion of closed sets in the labeled context (described in Sections 3 and 4) can be naturally interpreted as non-redundant descriptive rules (discriminating the target class) in the ROC space (Section 5). We also show that finding closed sets in labeled data turns out to be very useful in many applications. We have applied our proposal to reduce the number of emerging

patterns (Section 6.1), to compress the number of essential rules (Section 6.2), and finally, to learn descriptions for subgroup discovery on potato microarray data (Section 6.3).[1]

2. Background

Features, used for describing the training examples, are logical variables representing attribute-value pairs (called items in the association rule learning framework of Agrawal et al., 1996). If $F = \{f_1, \ldots, f_n\}$ is a fixed set of features, we can represent a training example as a tuple of features $f \in F$ with an associated class label. For instance, Table 1 contains examples for the simplified problem of contact lens prescriptions (Witten and Frank, 2005). Patients are described by four attributes: Age, Spectacle prescription, Astigmatism and Tear production rate; and each tuple is labeled with a class label: none, soft or hard. Then, F is the set of all attribute-value pairs in the data, that is, $F = \{$Age=young, $\ldots$, Tear=normal$\}$ (the class label is not included in F), and each example (a patient) corresponds to a subset of features in F with an associated class label. This small data set will be used throughout the paper to ease the understanding of our proposals.

We consider two-class learning problems where the set of examples E is divided into positives (P, target-class examples identified by label $+$) and negatives (N, labeled by $-$), and $E = P \cup N$. Multi-class problems can be translated to a series of two-class learning problems: each class is once selected as the target class (positive examples), while examples of all the other classes are treated as non-target class examples (thus, negative examples). For instance, when class soft of Table 1 is the target class, all examples with label soft are considered as positive, as shown in Table 2, and all examples labeled none and hard are considered as negative.

Given a rule $X \rightarrow +$ formed from a set of features $X \subseteq F$, *true positives* (TP) are those positive examples covered by the rule, that is, $p \in P$ such that $X \subseteq p$; and *false positives* (FP) are those negative examples covered by the rule, that is, $n \in N$ such that $X \subseteq n$; reciprocally, *true negatives* (TN) are those negative examples not covered by X. Later, we will see that some combinations of features $X \subseteq F$ produce more relevant antecedents than others for the rules $X \rightarrow +$. Our study will focus specifically on the combinations of features from the universe F which best define the space of non-redundant rules for the target class. We will do it by integrating the notion of closed itemsets and the concept of feature relevancy proposed in previous works.

2.1 Closed Itemsets

From the practical point of view of data mining algorithms, closed itemsets are the largest sets (w.r.t. set-theoretic inclusion) among those other itemsets occurring in the same examples (Bastide et al., 2000a; Crémilleux and Boulicaut, 2002; Pasquier et al., 2001; Taouil et al., 2000; Zaki, 2000a, 2004; Zaki and Ogihara, 1998). Formally, let *support of itemset* $X \subseteq F$, denoted by $\mathrm{supp}(X)$, be the number of examples in the data where X is contained. Then: a set $X \subseteq F$ is said to be *closed* when there is no other set $Y \subseteq F$ such that $X \subset Y$ and $\mathrm{supp}(X) = \mathrm{supp}(Y)$.

In the example of Table 2, the itemset corresponding to $\{$Age=young$\}$ is not closed because it can be extended to the maximal set $\{$Age=young, Astigmatism=no, Tear=normal$\}$ that has the same support in this data. Notice that by treating positive examples separately, the positive label will be already implicit in the closed itemsets mined on the target class data. So, here we will work by

1. A preliminary version of this work appeared in Garriga et al. (2006). This paper is improved based on the valuable reviewers' comments, incorporates proofs, detailed explanations, extended comparisons with related work and more experiments.

Id	Age	Spectacle prescription	Astig.	Tear prod.	Lens
1	young	myope	no	normal	soft
2	young	hypermetrope	no	normal	soft
3	pre-presbyopic	myope	no	normal	soft
4	pre-presbyopic	hypermetrope	no	normal	soft
5	presbyopic	hypermetrope	no	normal	soft
6	young	myope	no	reduced	none
7	young	myope	yes	reduced	none
8	young	hypermetrope	no	reduced	none
9	young	hypermetrope	yes	reduced	none
10	pre-presbyopic	myope	no	reduced	none
11	pre-presbyopic	myope	yes	reduced	none
12	pre-presbyopic	hypermetrope	no	reduced	none
13	pre-presbyopic	hypermetrope	yes	reduced	none
14	pre-presbyopic	hypermetrope	yes	normal	none
15	presbyopic	myope	no	reduced	none
16	presbyopic	myope	no	normal	none
17	presbyopic	myope	yes	reduced	none
18	presbyopic	hypermetrope	no	reduced	none
19	presbyopic	hypermetrope	yes	reduced	none
20	presbyopic	hypermetrope	yes	normal	none
21	young	myope	yes	normal	hard
22	young	hypermetrope	yes	normal	hard
23	pre-presbyopic	myope	yes	normal	hard
24	presbyopic	myope	yes	normal	hard

Table 1: The contact lens data set, proposed by Witten and Frank (2005).

Id	Age	Spectacle prescription	Astig.	Tear prod.	Class
1	young	myope	no	normal	+
2	young	hypermetrope	no	normal	+
3	pre-presbyopic	myope	no	normal	+
4	pre-presbyopic	hypermetrope	no	normal	+
5	presbyopic	hypermetrope	no	normal	+

Table 2: The set of positive examples when class soft of the contact lens data of Table 1 is selected as the target class. These examples form the set P of positive examples, while instances of classes none and hard are considered non-target, thus treated together as negative examples N. Note that examples are represented here in a simplified tabular form instead of the feature set representation.

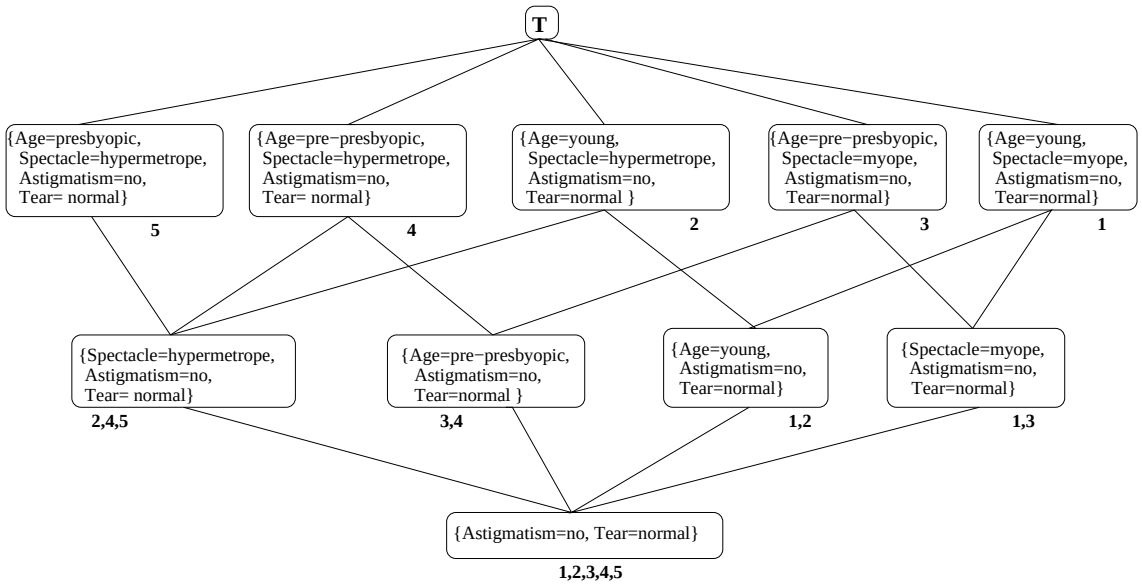


Figure 1: The lattice of closed itemsets for data in Table 2.

constructing the closure system of items on our positive examples and use this system to study the structural properties of the closed sets to discriminate the implicit label. Many efficient algorithms have been proposed for discovering closed itemsets over a certain minimum support threshold; see a compendium of them in Goethals and Zaki (2004).

The foundations of closed itemsets are based on the definition of a closure operator on a lattice of items (Carpineto and Romano, 2004; Ganter and Wille, 1998). The standard closure operator Γ for items acts as follows: the closure $\Gamma(X)$ of a set of items $X \subseteq F$ includes all items that are present in all examples having all items in X. According to the classical theory, operator Γ satisfies the following properties: Monotonicity: $X \subseteq X' \Rightarrow \Gamma(X) \subseteq \Gamma(X')$; Extensivity: $X \subseteq \Gamma(X)$; and Idempotency: $\Gamma(\Gamma(X)) = \Gamma(X)$.

From the formal point of view of Γ, closed sets are those coinciding with their closure, that is, for $X \subseteq F$, X is *closed* iff $\Gamma(X) = X$. Also, when $\Gamma(Y) = X$ for a set $Y \neq X$, it is said that Y is a *generator* of X. By extensivity of Γ we always have $Y \subseteq X$ for Y generator of X. Intensive work has focused on identifying which collection of generators is good to ensure that all closed sets can be produced. The named δ-free sets in Boulicaut et al. (2003) are minimal generators when $\delta = 0$, and these are equivalent to key patterns in Bastide et al. (2000b). Different properties of these δ-free sets generators in Boulicaut et al. (2003) have been studied for different values of δ.

Considering Table 2, we have the following $\Gamma(\{\text{Age=young}\}) = \{\text{Age=young, Astigmatism=no, Tear=normal}\}$. Then, $\{\text{Age=young}\}$ is a generator of this closed set. Note that for $\Gamma(Y) = X$, both Y and X are sets with exactly the same support in the data, but X being a largest set of items, that is, $Y \subset X$ for all Y such that $\Gamma(Y) = X$. This property is ensured by the extensivity of this operator. Moreover, closed sets formalized with operator Γ are exactly those sets obtained in closed set mining process and defined above, which present many advantages (see, for example, Balcázar and Baixeries, 2003; Crémilleux and Boulicaut, 2002).

Closed itemsets are lossless in the sense that they uniquely determine the set of all frequent itemsets and their exact support (cf. Pfaltz, 1996; Zaki and Ogihara, 1998, for more theoretical details). Closed sets of items can be graphically organized in a Hasse diagram, where each node corresponds to a closed itemset, and there is an edge between two nodes if and only if they are comparable (w.r.t. set-theoretic inclusion) and there is no other intermediate closed itemset in the lattice. In this partial order organization, ascending/descending paths represent the subset/superset relation. Typically, the top of this lattice is represented by a constant T corresponding to a set of items not included in any example.

Figure 1 shows the lattice of closed itemsets obtained from data from Table 2. Each node is depicted along with the set of example identifiers where the closed set occurs. Notice that all closed itemsets with the same support cover a different subset of transactions of the original data. In practice, such exponential lattices are not completely constructed, as only a list of closed itemsets over a certain minimum support suffices for practical purposes. Therefore, instead of closed sets one needs to talk about *frequent closed sets*, that is, those closed sets over the minimum support constraint given by the user. Also notice the difference of frequent closed sets from the popular concept of maximal frequent sets (see, for example, Tan et al., 2005), which refers to those sets for which none of their supersets are frequent.

Obviously, imposing a minimum support constraint will eliminate the largest closed sets whose support is typically very low. The impact of such constraint depends on the application. In general, there exists a trade-off between quality and speed up of the process. In the following we consider a theoretical framework with all closed sets; in practice though, we will need a minimum support constraint to consider only the frequent ones.

2.2 Relevant Features for Discrimination

The main aim of the theory of relevancy, described in Lavrač et al. (1999) and Lavrač and Gamberger (2005), is to reduce the hypothesis space by eliminating irrelevant features from F in the pre-processing phase. Other related work, such as Koller and Sahami (1996) and Liu and Motoda (1998), eliminate features in the model construction phase. However, here we concentrate on the elimination of irrelevant features in the preprocessing phase, as proposed by Lavrač and Gamberger (2005):

Definition 1 (Coverage of features) *Feature $f \in F$ covers another feature $f' \in F$ if and only if true positives of f' are a subset of true positives of f, and true negatives of f' are a subset of true negatives of f. In other words,* $\mathrm{TP}(f') \subseteq \mathrm{TP}(f)$ *and* $\mathrm{TN}(f') \subseteq \mathrm{TN}(f)$ *(or equivalently,* $\mathrm{TP}(f') \subseteq \mathrm{TP}(f)$ *and* $\mathrm{FP}(f) \subseteq \mathrm{FP}(f')$*).*

Using the definition of feature coverage, we further define that $f' \in F$ is *relatively irrelevant* if there exists another feature $f \in F$ such that f covers f'. To illustrate this notion we take the data of Table 1: if examples of class none form our positives and the rest of examples are considered negative, then the feature Tear=reduced covers Age=young, hence making this last feature irrelevant for the discrimination of the class none.

Other notions of irrelevancy described in Lavrač and Gamberger (2005) consider a minimum coverage constraint in the true positives or accordingly, on the true negatives.

3. Closed Sets on Target-class Data

Given a set of examples $E = P \cup N$ it is trivial to realize that for any rule $X \rightarrow +$ with a set of features $X \subseteq F$, the support of itemset X in P (target class examples) exactly corresponds to the number of true positives (TP) of the rule; reciprocally, the support of X in N (non-target class examples) is the number of false positives (FP) of the rule. Also, because of the anti-monotonicity property of support (i.e., $Y \subseteq X$ implies $\text{supp}(X) \leq \text{supp}(Y)$) the following useful property can be easily stated.

Proposition 2 *Let $X, Y \subseteq F$ such that $Y \subseteq X$, then* $\text{TP}(X) \subseteq \text{TP}(Y)$ *and* $\text{FP}(X) \subseteq \text{FP}(Y)$.

Proof The anti-monotonicity property of support on the set of positive examples ensures that $|\text{TP}(X)| \leq |\text{TP}(Y)|$. Since $Y \subseteq X$, we necessarily have $\text{TP}(X) \subseteq \text{TP}(Y)$. The same reasoning applies to the set of negative examples. ■

For convenience, let $\text{supp}^+(X)$ denote the support of the set X in the positive set of examples P, and $\text{supp}^-(X)$ the support in the negative set of examples N. Notice that for a rule $X \rightarrow +$ we indeed have that $\text{supp}^+(X) = |\text{TP}(X)|$ and $\text{supp}^-(X) = |\text{FP}(X)|$. In the following we will use one notation or the other according to the convenience of the context.

Following from the last proposition, the next property can be readily seen.

Lemma 3 *Feature $f \in F$ covers another feature $f' \in F$ (as in Definition 1), iff* $\text{supp}^+(\{f'\}) = \text{supp}^+(\{f, f'\})$ *and* $\text{supp}^-(\{f\}) = \text{supp}^-(\{f, f'\})$.

Proof That f covers f' can be formulated as $\text{TP}(f') \subseteq \text{TP}(f)$ and $\text{FP}(f) \subseteq \text{FP}(f')$. Because all the true positives of f' are also covered by f, it is true that $\text{TP}(f') = \text{TP}(f, f')$; similarly, because all the false positives of f are also covered by f' we have $\text{FP}(f) = \text{FP}(f, f')$. These two facts directly imply that $\text{supp}^+(\{f'\}) = \text{supp}^+(\{f, f'\})$ and $\text{supp}^-(\{f\}) = \text{supp}^-(\{f, f'\})$.

The other direction is proved as follows. The anti-monotonicity property of Proposition 2 applied over $\{f'\} \subseteq \{f, f'\}$ leads to $\text{TP}(f, f') \subseteq \text{TP}(f')$. Indeed, from $\text{supp}^+(\{f'\}) = \text{supp}^+(\{f, f'\})$ we have $|\text{TP}(f')| = |\text{TP}(f, f')|$, which along with $\text{TP}(f, f') \subseteq \text{TP}(f')$ implies an equivalence of true positives between these two sets: that is, $\text{TP}(f, f') = \text{TP}(f')$. From here we deduce $\text{TP}(f') \subseteq \text{TP}(f)$. Exactly the same reasoning applies to the negatives. Proposition 2 ensures that $\text{FP}(f, f') \subseteq \text{FP}(f)$ because $\{f\} \subseteq \{f, f'\}$. But from $\text{supp}^-(\{f\}) = \text{supp}^-(\{f, f'\})$ we have $|\text{FP}(f)| = |\text{FP}(f, f')|$, which together with $\text{FP}(f, f') \subseteq \text{FP}(f)$ leads to the equivalence of the false positives between these two sets: that is, $\text{FP}(f) = \text{FP}(f, f')$. Then, we deduce $\text{FP}(f) \subseteq \text{FP}(f')$. That is f covers f' as in Definition 1. ■

Indeed, this last result allows us to rewrite, within the data mining language, the definition of relevancy proposed by Lavrač et al. (1999) and Lavrač and Gamberger (2005): a feature f is *more relevant* than f' when $\text{supp}^+(\{f'\}) = \text{supp}^+(\{f, f'\})$ and $\text{supp}^-(\{f\}) = \text{supp}^-(\{f, f'\})$. For instance, the support of {Age=young} over the class none of data from Table 1 is equal to the support of {Age=young, Tear=reduced} in this same class none ; at the same time, the support of {Tear=reduced} is zero in the negatives (formed here by the classes soft and hard together), thus equal to the support in the negatives of {Age=young, Tear=reduced}. So, the feature Age=young is irrelevant with respect to Tear=reduced, as we identified in Section 2.1. In other words, f' is

irrelevant with respect to f if the occurrence of f' always implies the presence of f in the positives, and at the same time, f always implies the presence of f' in the negatives.

To the effect of our later arguments it will be useful to cast the result of Lemma 3 in terms of the formal closure operator Γ. This will provide the desired mapping from relevant sets of features to the lattice of closed itemsets constructed on target class examples. Again, because we need to formalize our arguments against positive and negative examples separately, we will use Γ^+ or Γ^- for the closure of itemsets on P or N respectively.

Lemma 4 *A feature f is more relevant than f' iff $\Gamma^+(\{f'\}) = \Gamma^+(\{f, f'\})$ and $\Gamma^-(\{f\}) = \Gamma^-(\{f, f'\})$.*

Proof It follows immediately from Lemma 3 and the formalization of operator Γ. A feature f is more relevant than f' when f covers f' according to Definition 1. Then, by Lemma 3 we have that $\text{supp}^+(\{f'\}) = \text{supp}^+(\{f, f'\})$ and $\text{supp}^-(\{f\}) = \text{supp}^-(\{f, f'\})$. By construction of Γ, this means that the sets $\{f'\}$ and $\{f, f'\}$ have the same closure on the positives, and the sets $\{f\}$ and $\{f, f'\}$ have the same closure on the negatives. That is: because Γ is an extensive operator, we can rewrite it as $\Gamma^+(\{f'\}) = \Gamma^+(\{f, f'\})$ and $\Gamma^-(\{f\}) = \Gamma^-(\{f, f'\})$. ■

Interestingly, operator Γ is formally defined for the universe of sets of items, so that these relevancy results on single features can be directly extended to sets of features. This provides a proper generalization, which we express in the following definition.

Definition 5 (Relevancy of feature sets) *Set of features $X \subseteq F$ is more relevant than set $Y \subseteq F$ iff $\Gamma^+(Y) = \Gamma^+(X \cup Y)$ and $\Gamma^-(X) = \Gamma^-(X \cup Y)$.*

To illustrate Definition 5 take the positive examples from Table 2, with negative data formed by classes none and hard together. Feature Spectacle=myope alone cannot be compared to feature Astigmatism=no alone with Definition 1 (because Astigmatism=no does not always imply Spectacle=myope in the negatives). For the same reason, Spectacle=myope cannot be compared to feature Tear=normal alone. However, when considering these two features together, then Spectacle=myope turns out to be irrelevant w.r.t. the set {Astigmatism=no, Tear=normal}. So, the new semantic notion of Definition 5 allows us to decide if a set of features is structurally more important than another for discriminating the target class. In the language of rules: rule $Y \rightarrow +$ is *irrelevant* if there exists another rule $X \rightarrow +$ satisfying two conditions: first, $\Gamma^+(Y) = \Gamma^+(X \cup Y)$; and second, $\Gamma^-(X) = \Gamma^-(X \cup Y)$. E.g., when soft is the target class: the rule Spectacle=myope $\rightarrow +$ is not relevant because at least the rule {Astigmatism=no, Tear=normal} $\rightarrow +$ will be more relevant.

Finally, from the structural properties of operator Γ and from Proposition 2, we can deduce that the semantics of relevant sets in Definition 5 is consistent.

Lemma 6 *A set of features $X \subseteq F$ is more relevant than set $Y \subseteq F$ (Definition 5) iff* $\text{TP}(Y) \subseteq \text{TP}(X)$ *and* $\text{FP}(X) \subseteq \text{FP}(Y)$.

Proof That X is more relevant than Y means $\Gamma^+(Y) = \Gamma^+(X \cup Y)$ and $\Gamma^-(X) = \Gamma^-(X \cup Y)$. Proposition 2 ensures that $\text{TP}(X \cup Y) \subseteq \text{TP}(Y)$ because $Y \subseteq X \cup Y$. Then, from $\Gamma^+(Y) = \Gamma^+(X \cup Y)$ we naturally have that $|\text{TP}(Y)| = |\text{TP}(X \cup Y)|$ (by formalization of Γ), which together with $\text{TP}(X \cup Y) \subseteq \text{TP}(Y)$ leads to the equality of the true positives between the following sets: $\text{TP}(X \cup Y) = \text{TP}(Y)$.

From here, $\mathrm{TP}(Y) \subseteq \mathrm{TP}(X)$. On the other hand, it is implied by the definition of relevancy that $Y \subseteq X$, thus directly from Proposition 2 we have that $\mathrm{FP}(X) \subseteq \mathrm{FP}(Y)$.

The other direction is proved as follows. Let X and Y be two sets such that $\mathrm{TP}(Y) \subseteq \mathrm{TP}(X)$ and $\mathrm{FP}(X) \subseteq \mathrm{FP}(Y)$. As all the true positives of Y are also covered by X, it is true that $\mathrm{TP}(Y) = \mathrm{TP}(X \cup Y)$; similarly, as all the false positives of X are also covered by Y we have that $\mathrm{FP}(X) = \mathrm{FP}(X \cup Y)$. This directly implies that $\mathrm{supp}^+(Y) = \mathrm{supp}^+(X \cup Y)$ and $\mathrm{supp}^-(X) = \mathrm{supp}^-(X \cup Y)$. By construction of Γ, this means we can directly rewrite this as $\Gamma^+(Y) = \Gamma^+(X \cup Y)$ and $\Gamma^-(X) = \Gamma^-(X \cup Y)$. That is: set X is more relevant than Y by Definition 5. ■

In the language of rules, Lemma 6 implies that when a set of features $X \subseteq F$ is more relevant than $Y \subseteq F$, then rule $Y \rightarrow +$ is less relevant than rule $X \rightarrow +$ for discriminating the target class. Moreover, Lemma 6 proves the consistency of Definition 5. If we consider $X = \{f\}$ and $Y = \{f'\}$, then the definition is simply reduced to the coverage of Definition 1. Yet, the interestingness of Definition 5 is that we can use this new concept to study the relevancy of itemsets (discovered in the mining process) for discrimination problems. Also, it can be immediately seen that if X is more relevant than Y in the positives, then Y will be more relevant than X in the negatives (by just reversing Definition 5).

Next subsection characterizes the role of closed itemsets to find relevant sets of features for discrimination. Notice that the first condition to consider a set X more relevant than Y in the discrimination of target class examples is that $\Gamma^+(Y) = \Gamma^+(X \cup Y)$. So, the closure system constructed on the positive examples will be proved to be structurally important for inducing target class rules.

3.1 Closed Sets for Discrimination

Together with the result of Lemma 6, it can be shown that only closed itemsets mined in the set of positive examples suffice for discrimination.

Theorem 7 *Let $Y \subseteq F$ be a set of features such that $\Gamma^+(Y) = X$ and $Y \neq X$. Then, set Y is less relevant than X (as in Definition 5).*[2]

Proof By the extensivity property of Γ we know $Y \subseteq X$. Then, Proposition 2 ensures that $\mathrm{TP}(X) \subseteq \mathrm{TP}(Y)$ and $\mathrm{FP}(X) \subseteq \mathrm{FP}(Y)$. However, by hypothesis we have $\Gamma^+(Y) = X$, which by construction ensures that $|\mathrm{TP}(Y)| = |\mathrm{TP}(X)|$; but because $Y \subseteq X$, it must be true that $\mathrm{TP}(Y) = \mathrm{TP}(X)$. In all, we obtained that $\mathrm{TP}(Y) = \mathrm{TP}(X)$ and $\mathrm{FP}(X) \subseteq \mathrm{FP}(Y)$, and from Lemma 6 we have that X is more relevant than Y. ■

Typically, in approaches such as Apriori-C (Jovanoski and Lavrač, 2001), Apriori-SD (Kavšek and Lavrač, 2006) or RLSD (Zhang et al., 2004), frequent itemsets with very small minimal support constraint are initially mined and subsequently post-processed in order to find the most suitable rules

2. We are aware that some generators Y of a closed set X might be exactly equivalent to X in terms of TP and FP, thus forming equivalence classes of rules (i.e., $Y \rightarrow +$ might be equivalent to $X \rightarrow +$). The result of this theorem characterizes closed sets in the positives as those representatives of relevant rules; so, any set which is not closed can be discarded, and thus, efficient closed mining algorithms can be employed for discrimination purposes. The next section will approach the notion of the shortest representation of a relevant rule, which will be conveyed by these mentioned equivalent generators.

for discrimination. The new result presented here states that not all frequent itemsets are necessary: as shown in Theorem 7 only the closed sets have the potential to be relevant.

To illustrate this result we use again data in Table 2, where $\Gamma^{+}(\{\text{Astigmatism=no}\}) = \{\text{Astigmatism=no, Tear=normal}\}$. Thus, rule Astigmatism=no $\rightarrow +$ can be discarded: it covers exactly the same positives as $\{$Astigmatism=no, Tear=normal$\}$, but more negatives. Thus, a rule whose antecedent is $\{$Astigmatism=no, Tear=normal$\}$ would be preferred for discriminating the class soft.

However, Theorem 7 simply states that those itemsets which are not closed in the set of positive examples cannot form a relevant rule to discriminate the target class, thus they do not correspond to a relevant combination of features. In other words, closed itemsets suffice but some of them might not be necessary to discriminate the target class. It might well be that a closed itemset is irrelevant with respect to another closed itemset in the system.

As illustrated above, when considering class soft as the target class (identified by $+$), we had that feature Spectacle=myope is irrelevant with respect to set $\{$Astigmatism=no, Tear=normal$\}$; yet, set $\{$Spectacle=myope, Astigmatism=no, Tear=normal$\}$ is closed in the system (see the lattice of Figure 1). Indeed, this latter closed set is still irrelevant in the system according to our Definition 5 and can be pruned away. The next section is dedicated to the task of reducing the closure system of itemsets to characterize the final space of relevant sets of features.

4. Characterizing the Space of Relevant Sets of Features

This section studies how the dual closure system on the negative examples is used to reduce the lattice of closed sets on the positives. This reduction will characterize a complete space of relevant sets of features for discriminating the target class. First of all, we raise the following two important remarks following from Proposition 2.

Remark 8 *Given two different closed sets on the positives X and X' such that $X \nsubseteq X'$ and $X' \nsubseteq X$ (i.e., there is no ascending/descending path between them in the lattice), then they cannot be compared in terms of relevancy, since they cover different positive examples.*

We exemplify Remark 8 with the lattice in Figure 1. The two closed sets: $\{$Age=young, Astigmatism=no, Tear=normal$\}$ and $\{$Spectacle=myope, Astigmatism=no, Tear=normal$\}$, are not comparable with subset relation: they cover different positive examples and they cannot be compared in terms of relevance.

Remark 9 *Given two closed sets on the positives X and X' with $X \subset X'$, we have by construction that* $\text{TP}(X') \subset \text{TP}(X)$ *and* $\text{FP}(X') \subseteq \text{FP}(X)$ *(from Proposition 2). Notice that because X and X' are different closed sets in the positives,* $\text{TP}(X')$ *is necessarily a proper subset of* $\text{TP}(X)$*; however, regarding the coverage of false positives, this inclusion is not necessarily proper.*

To illustrate Remark 9 we use the lattice of closed itemsets in Figure 1. By construction the closed set $\{$Spectacle=myope, Astigmatism=no, Tear=normal$\}$ from Figure 1 covers fewer positives than the proper predecessor $\{$Astigmatism=no, Tear=normal$\}$. However, both closed sets cover exactly one negative example. In this case $\{$Astigmatism=no, Tear= normal$\}$ is more relevant than $\{$Spectacle=myope, Astigmatism=no, Tear=normal$\}$.

Remark 9 points out that two different closed sets in the positives, yet being one included in the other, may end up covering exactly the same set of false positives. In this case, we would like

Transaction occurrence list	Closed Set
$1,2,3,4,5$	{Astigmatism=no, Tear=normal }
$2,4,5$	{Spectacle=hypermetrope, Astigmatism=no, Tear=normal }
$3,4$	{Age=pre-presbyopic, Astigmatism=no, Tear=normal }
$1,2$	{Age=young, Astigmatism=no, Tear=normal }

Table 3: The four closed sets corresponding to the space of relevant sets of features for data in Table 2.

to discard the closed set covering less true positives. Because of the anti-monotonicity property of support, the smaller one will be the most relevant.

From these two remarks we obtain the following result.

Theorem 10 *Let $X \subseteq F$ and $X' \subseteq F$ be two different closed sets in the positives such that $X \subset X'$. Then, we have that X' is less relevant than X (as in Definition 5) iff $\Gamma^-(X) = \Gamma^-(X')$.*

Proof That X' is less relevant than X is defined as: $\Gamma^+(X') = \Gamma^+(X' \cup X)$ and $\Gamma^-(X) = \Gamma^-(X' \cup X)$. Since $X \subset X'$ by hypothesis, we always have that $X' = X' \cup X$, so that the above two conditions can be rewritten as $\Gamma^+(X') = \Gamma^+(X')$ (always true) and $\Gamma^-(X) = \Gamma^-(X')$, as we wanted to prove.

In the backward direction we start from $\Gamma^-(X) = \Gamma^-(X')$, where $X \subset X'$ as stated by hypothesis of the theorem. Because $X \subset X'$ it is true that $X' = X' \cup X$. Then, we can rewrite $\Gamma^-(X) = \Gamma^-(X')$ as $\Gamma^-(X) = \Gamma^-(X' \cup X)$, thus satisfying already the first condition of Definition 5. Also, $\Gamma^+(X')$ is simply the same as $\Gamma^+(X') = \Gamma^+(X' \cup X)$, thus satisfying the second condition of Definition 5. ■

Thus, by Theorem 10 we can reduce the closure system constructed on the positives by discarding irrelevant nodes: if two closed itemsets are connected by an ascending/descending path on the lattice of positives (i.e., they are comparable by set inclusion $\subset$), yet they have the same closure on the negatives (i.e., they cover the same false positives, or equivalently, their support on the negatives is exactly the same), then just the shortest set is relevant.

Finally, after Theorem 7 and Theorem 10, we can characterize the space of relevant sets of features for discriminating the selected target class as follows.

Definition 11 (Space of relevant sets of features) *The space of relevant combinations of features for discriminating the target class is defined as those sets X for which it holds that: $\Gamma^+(X) = X$ and there is no other closed set $\Gamma^+(X') = X'$ such that $\Gamma^-(X') = \Gamma^-(X)$.*

It is trivial to see after Remarks 8 and 9, that by construction, any two sets in this space always cover a different set of positives and a different set of negatives. These final sets can be directly interpreted as antecedents of rules for classifying the target class (i.e., for each relevant $X \subseteq F$ in the space, we have a relevant rule $X \rightarrow +$ for classifying the positives).

The four closed sets forming the space of relevant sets of features for the class soft are shown in Table 3. It can be checked that the CN2 algorithm (Clark and Niblett, 1989) would output a single

rule whose antecedent corresponds to the closed set in the first row of Table 3. On the other hand, Ripper (Cohen, 1995) would obtain the most specific relevant rules, that is, those corresponding to the three last rows from Table 3. Finally, other algorithms such as Apriori-C would also output rules whose antecedents are not relevant as such, for example, Astigmatism=no $\rightarrow$ Lenses= soft.

To complete the example of the contact lenses database: the lattice of closed itemsets on the class hard contains a total of 7 nodes, which is reduced to only 3 relevant sets; on the other hand, the lattice of closed itemsets on the class none contains a total of 61 nodes, which is reduced to 19 relevant sets.

The space of relevant combinations defines exhaustively all the relevant antecedents for discriminating the target class. Not to generate this space completely, in large sets of data a minimum support threshold will be usually imposed (see more details in the experimental section). As expected, too large relevant sets will be naturally pruned by the minimum support constraint, which might have an undesired effect depending on the application. Still, it is known that very long closed sets, that is, too specific sets of features in our contribution, tend to overestimate when constructing a classifier or learning a discriminative model. In general, it will be up to the user to find a proper trade off between quality of the results and speed up of the process.

4.1 Shortest Representation of a Relevant Set

Based on Theorem 7 we know that generators Y of a closed set X are characterized to cover exactly the same positive examples, and at least the same negative examples. Because of this property, any generator will be redundant w.r.t. its closure. That is:

Remark 12 *Let Y be a generator of X in the closure system on the positives; then, $\Gamma^+(Y) = X$ always implies $\mathrm{TP}(Y) = \mathrm{TP}(X)$ and $\mathrm{FP}(X) \subseteq \mathrm{FP}(Y)$ (from Lemma 6 and Theorem 7). However, note that the inclusion between the set of false positives is not necessarily proper.*

However, we have $\mathrm{FP}(X) \subseteq \mathrm{FP}(Y)$ for Y generator of X; so, it might happen that some generators Y are equivalent to their closed set X in that they cover exactly the same true positives and also the same false positives.

Definition 13 (Equivalent generators) *Let $\Gamma^+(Y) = X$ and $Y \neq X$. We say that a generator Y is equivalent to its closure X iff $\mathrm{FP}(X) = \mathrm{FP}(Y)$.*

The equivalence between true positives of Y and X is guaranteed because $\Gamma^+(Y) = X$. Therefore, it would be only necessary to check if generators cover the same false positives than its closure to check equivalence. Generators will provide a more general representation of the relevant set (because $Y \subset X$ by construction). So, $Y \rightarrow +$ is shorter than the rule $X \rightarrow +$ and it is up to the user to choose the more meaningful to her or to the application. For example, this may depend on a minimum-length criterion of the final classification rules: a generator Y equivalent to a closed set X satisfies by construction that $Y \subset X$, so $Y \rightarrow +$ is shorter than the rule $X \rightarrow +$. Then, the minimal equivalent generators of a closed itemset X naturally correspond to the minimal representation of the relevant rule $X \rightarrow +$.

In terms of the closure operator of negatives, we have the following way of characterizing these equivalent generators.

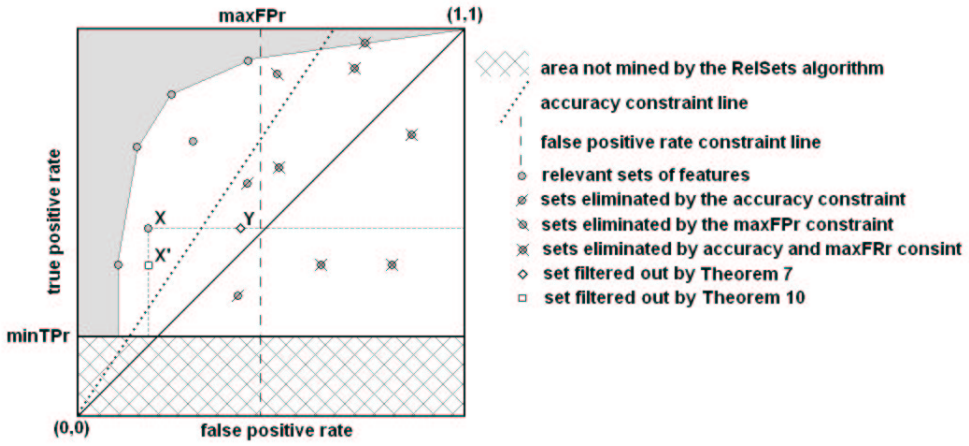


Figure 2: The evaluation of relevant combinations of features in the ROC space.

Proposition 14 *Let* $\Gamma^+(Y) = X$ *and* $Y \neq X$. *Then* Y *is an equivalent generator of* X *iff* $\Gamma^-(X) = \Gamma^-(Y)$.

Proof It is defined that the generator Y is equivalent to its closure X when $\mathrm{FP}(X) = \mathrm{FP}(Y)$, which directly implies $\Gamma^-(X) = \Gamma^-(Y)$ by construction of Γ. On the other direction: $\Gamma^-(X) = \Gamma^-(Y)$ implies $|\mathrm{FP}(Y)| = |\mathrm{FP}(X)|$, but because $Y \subseteq X$ by the extensivity of Γ, we necessarily have that $\mathrm{FP}(Y) = \mathrm{FP}(X)$. ■

It is well-known that minimal generators of a closed set X can be computed by traversing the hypergraph of differences between X and their proper predecessors in the system (see, for example, Pfaltz and Taylor, 2002). In practice, efficient algorithms have been designed for computing free sets and their generalizations (see, for example, Calders and Goethals, 2003).

5. Evaluation of Relevant Sets in the ROC Space

The ROC space (Provost and Fawcett, 2001) is a 2-dimensional space that shows a classifier (rule/ruleset) performance in terms of its *false positive rate* (also called 'false alarm'), $\mathrm{FPr} = \frac{|\mathrm{FP}|}{|\mathrm{TN}|+|\mathrm{FP}|} = \frac{|\mathrm{FP}|}{|N|}$ plotted on the X-axis, and *true positive rate* (also called 'sensitivity') $\mathrm{TPr} = \frac{|\mathrm{TP}|}{|\mathrm{TP}|+|\mathrm{FN}|} = \frac{|\mathrm{TP}|}{|P|}$ plotted on the Y-axis. The ROC space is appropriate for measuring the quality of rules since rules with the best covering properties are placed in the top left corner, while rules that have similar distribution of covered positives and negatives as the distribution in the entire data set are close to the main diagonal.

A set of features from Definition 5 can be interpreted as a condition part of a rule or also as a subgroup description. A set of relevant sets of features from Definition 11 can therefore be visualized and evaluated in the ROC space as a ruleset.

Relevant sets are induced with a minimum support constraint on the positives (as discussed in Section 4). This means that in the ROC space they all lie above the minimum true positive rate constraint line (in Figure 2 denoted as minTPr). Relevant sets are depicted in Figure 2 as circles.

Sometimes, depending on the application, additional filtering criteria are applied. In such cases a maximum false positive rate constraint can be imposed (in Figure 2 this constraint is represented by a dashed line, rules eliminated by this constraint are shown as circles with backslash), or we can apply a minimum confidence constraint (represented by a dotted line, rules eliminated by this constraint are shown as slashed circles in Figure 2). Alternatively we may simply select just the rules on the convex hull.

Let us interpret and visualize Theorems 7 and 10 in the ROC space. According to Theorem 7, sets of features Y, s.t. $Y \subset X$, that cover the same positives as X (i.e., $TP(Y) = TP(X)$), are filtered out. Since Y and X have the same true positive rate (i.e., $TPr(Y) = TPr(X)$), both lie on the same horizontal line in the ROC space. Since Y is a subset of X, which in rule learning terminology translates into "rule X is a specialization of rule Y", $FPr(X) \leq FPr(Y)$ so Y is located at the right hand side of X. In Figure 2, a sample feature set filtered out according to Theorem 7 is depicted as a diamond. Note that this captures exactly the notion of relevancy defined by Lavrač and Gamberger (2005) and Lavrač et al. (1999).

According to Theorem 10, sets of features X', s.t. $X \subset X'$, that cover the same negatives as X (i.e., $FP(X') = FP(X)$), are filtered out. Since X' and X have the same false positive rate (i.e., $FPr(X') = FPr(X)$), both lie on the same vertical line in the ROC space. Since X is a subset of X', which in rule learning terminology translates into "rule X' is a specialization of rule X ", $TPr(X) \geq TPr(X')$, therefore X is located above X' in the ROC space. In Figure 2, a sample feature set filtered out according to Theorem 10 is depicted as a square.

Note that the feature sets filtered out by the relevancy filter are never those on the ROC convex hull. Furthermore, it can be proved that there are no sets of features outside the convex hull (grey area on Figure 2 denotes an area without sets/rules).

6. Experimental Evaluation

The results presented above lead to the concept of closed sets in the context of labeled data. In practice, closed sets can be discovered from labeled data as follows.

1. First, mining the set $S = \{X_1, \ldots, X_n\}$ of frequent closed itemsets from the target class (Theorem 7). This requires a minimum support constraint on positives. For our experiments we will use the efficient LCM algorithm by Uno et al. (2004).

2. Second, reducing S to the space of relevant set of features by checking the coverage in the negatives (Theorem 10). Schematically, for any closed set $X_i \in S$, if there exists another closed set $X_j \in S$ such that both have the same support in the negatives and $X_j \subset X_i$, then X_i is removed.

The first step of this process usually requires a minimum support constraint on true positives, while the second step can be computed automatically without any constraints. However, depending on the purpose of the application we can apply an extra filtering criterion (such as forcing a maximum false positive constraint on the negatives, or a minimum accuracy constraint), or compute minimal equivalent generators of the relevant sets as described above. For short, we will name this computing process as *RelSets* (i.e., the process of discovering the Relevant Sets of features of Definition 5).

			Emerging Patterns					
			Growth rate > 1.5			Growth rate ∞		
Data set	Class	Distrib. %	EPs	RelSets	CF%	EPs	RelSets	CF%
Lenses	soft	20.8	31	4	87.10	8	3	62.5
	hard	16.9	34	3	91.18	6	2	66.67
	none	62.5	50	12	76.00	42	4	90.48
Iris	setosa	33.3	83	16	80.72	71	7	90.14
	versicolor	33.3	134	40	70.15	63	10	84.13
	virginica	33.3	92	16	82.61	68	6	91.18
Breast-w	benign	65.5	6224	316	94.92	5764	141	97.55
	malignant	34.5	3326	628	81.12	2813	356	87.34
SAheart	0	34.3	4557	1897	58.37	2282	556	75.64
	1	65.7	9289	2824	69.60	3352	455	86.43
Balance-scale	B	7.8	271	75	72.32	49	49	0.00
	R	46	300	84	72.00	90	90	0.00
Yeast	MIT	16.4	3185	675	78.81	250	40	84.00
	CYT	31.2	3243	808	75.08	68	16	76.47
	ERL	0.3	1036	5	99.52	438	4	99.09
Monk-1	0	64.3	1131	828	26.79	321	18	94.39
	1	35.7	686	9	98.69	681	4	99.41
Lymphography	metastases	54.72	36435	666	98.17	10970	90	99.18
10% min supp.	malign	41.21	61130	740	98.79	19497	55	99.72
Crx	$+$	44.5	3366	782	76.76	304	26	91.44
10% min supp.	$-$	55.5	3168	721	77.24	12	5	58.33

Table 4: Compression factor (CF% = $(1 - \frac{|RelSets|}{|EPs|}) \times 100$) of EPs in several UCI data sets. Note that we did not impose any minimum true positive threshold on any data set, except for Lymphography and Crx, where all EPs and RelSets were discovered with a 10% threshold on true positives.

As discussed above, the minimum support constraint on the first phase will tend to prune too long closed sets and this might have an impact in the application. In practice however, it is known that the longest sets of features are sometimes too specific, thus leading to overfitting problems. It is up to the user to trade off between the specificity of the closed sets and the speed up of the process. Also notice that the lowest the minimum support constraint, the largest the number of closed sets, and thus, the most expensive it becomes to compute the second phase of the approach. Our goal is not to present efficient algorithms but to illustrate the concept of relevancy.

Still we find important to point out that the notion of relevancy explored in the paper prefers typically the shortest closed sets. This is obvious by the second reduction phase shown in Theorem 10, where the shortest sets are always more relevant than the longest ones if they cover the same negative examples. Thus, finding a proper threshold level for the minimum support is not critical in our experiments as different minimum support thresholds lead to very similar results.

6.1 Emerging Patterns on UCI data

Emerging Patterns (EP) (Dong and Li, 1999; Li et al., 2000; Dong et al., 1999) are sets of features in the data whose supports change significantly from one class to another. More specifically, EPs are itemsets whose growth rates (the ratio of support from one class to the other, that is, $\frac{\text{TPr}}{\text{FPr}}$ of the pattern) are larger than a user-specified threshold. In this experimental setting we want to show that some of the EPs mined by these approaches are redundant, and that our relevant sets correspond to the notion of compacted data representation for labeled data. Indeed, EPs are a superset of the result returned by RelSets.

In our comparisons we calculate relevant sets over a certain growth rate threshold (1.5 and infinite), and we compare this with the number of EPs by using the same growth rate constraint. Numerical attributes in the data sets are discretized when necessary by using four equal frequency intervals. Although being a very simple discretization scheme, we want to point out that our goal in this experiment is to compare the number of EPs with our relevant sets, and thus, any preprocessing decision on the original data will affect in the same way the two methods we wish to compare.

Results are shown in Table 4. We observe that compression factor may vary according to the data set. When data is structurally redundant, compression factors are higher since many frequent sets are redundant with respect to the closed sets. However, in data sets where this structural redundancy does not exist (such as the Balance-scale data), the compression factor is zero, or close to zero.

A set of relevant properties of EPs have been studied in Soulet et al. (2004). This latter work also identifies condensed representations of EPs from closed sets mined in the whole database. Our approach is different in that we deal with pieces of the data for each class separately, and this allows for a reduction phase given by Theorem 10. Indeed, the amount of compression that this second phase provides in our approach depends on the distribution of the negative examples in the data, but at least, the number of relevant sets obtained by RelSets will be always smaller than the number of condensed EPs from Soulet et al. (2004).

6.2 Essential Rules on UCI Data

Essential rules were proposed by Baralis and Chiusano (2004) to reduce the number of association rules to those with nonredundant properties for classification purposes. Technically, they correspond to mining all frequent itemsets and removing those sets X such that there exists another frequent Y with $Y \subset X$ and having both the same support in positives and negatives. This differs from our proposal in the way of treating the positive class with closed sets. The compression factor achieved for these rules is shown in Table 5. Note that essential rules are not pruned by growth rate threshold, and this is why their number is usually higher than the number of emerging patterns shown in previous subsection.

6.3 Subgroup Discovery in Microarray Data Analysis

Microarray gene expression technology offers researchers the ability to simultaneously examine expression levels of hundreds or thousands of genes in a single experiment. Knowledge about gene regulation and expression can be gained by dividing samples into control samples (in our case mock infected plants), and treatment samples (in our case virus infected plants). Studying the differences between gene expression of the two groups (control and treatment) can provide useful insights into complex patterns of host relationships between plants and pathogens (Taiz and Zeiger, 1998).

Data set	Class	Distrib. %	Essential rules	RelSets	CF%
Lenses	soft	20.8	43	4	90.69
	hard	16.9	39	3	92.30
	none	62.5	89	19	78.65
Iris	setosa	33.3	76	20	73.68
	versicolor	33.3	111	41	63.06
	virginica	33.3	96	27	71.87
Breast-w	benign	65.5	3118	377	87.90
	malignant	34.5	2733	731	73.25
SAheart	0	34.3	6358	4074	35.92
	1	65.7	9622	4042	58
Balance-scale	B	7.8	415	147	88.67
	R	46	384	364	5.20
Yeast	MIT	16.4	2258	1125	50.17
	CYT	31.2	2399	1461	80.78
	ERL	0.3	417	5	98.80
Monk-1	0	64.3	1438	1135	21.07
	1	35.7	1477	363	75.42
Lymphography	metastases	54.72	1718	369	78.52
10% min supp.	malign	41.21	2407	476	80.22
Crx	$+$	44.5	2345	1091	53.47
10% min supp.	$-$	55.5	2336	1031	55.86

Table 5: Compression factor (CF% = $(1 - \frac{|RelSets|}{|EPs|}) \times 100$) of essential rules in UCI data sets. Note that essential rules and RelSets are not pruned by any growth rate threshold.

Microarray data analysis problems are usually addressed by statistical and data mining/machine learning approaches (Speed, 2003; Causton et al., 2003; Parmigiani et al., 2003). State-of-the-art machine learning approaches to microarray data analysis include both supervised learning (learning from data with class labels) and unsupervised learning (such as conceptual clustering). A review of these various approaches can be found in Molla et al. (2004). It was shown by Gamberger et al. (2004) that microarray data analysis problems can be approached also through subgroup discovery, where the goal is to find a set of subgroup descriptions (a rule set) for the target class, that preferably has a low number of rules while each rule has high coverage and accuracy (Lavrač et al., 2004; Gamberger and Lavrač, 2002).

The goal of the real-life experiment addressed in this paper is to investigate the differences between virus sensitive and resistant transgenic potato lines. For this purpose, 48 potato samples were used, leading to 24 microarrays. The laboratory experiment was carried out at the National Institute of Biology, Ljubljana, Slovenia.

Our data set contains 12 examples. Each example is a pair of microarrays (8 and 12 hours after infection) from the same transgenic line. All the data was discretized by using expert background knowledge. Features of the form $|\textit{gene expression value}| > 0.3$ were generated and enumerated. Three groups of features were generated: first group corresponding to gene expression levels 8 hours after infection (feature numbers $\in [1, 12493]$); second group corresponding to gene expression levels 12 hours after infection (feature numbers $\in [12494, 24965]$); finally, a third group corresponding

Data set	Class	Num. of rules			AUC		Time	
		RelSets	RelSets-ROC	SD	RelSets	SD	RelSets	SD
potatoes	sensitive	1	1	20	100%	100%	<1s	>1h
	resistant	1	1	20	100%	91%	<1s	>1h

Table 6: Comparison of algorithms RelSets and SD on the potato microarray data. Column RelSets-ROC shows the number of RelSets rules on the ROC convex hull.

to the difference between gene expression levels 12 and 8 hours after infection (feature numbers $\in [24966, 37559]$).

We used the RelSets algorithm to analyze the differences between gene expression levels characteristic for virus sensitive potato transgenic lines, discriminating them from virus resistant potato transgenic lines and vice versa. We ran it twice: once the sensitive examples were considered positive and once the resistant ones were considered positive. In both cases the constraint of minimal true positive count was set to 4, and in the first phase the algorithm returned 22 closed sets on positives. Rule relevancy filtering according to Definition 5, filtered the rules to just one relevant rule with a 100% true positive rate and a 0% false positive rate for each class. The results gained are shown below, where features are represented by numbers.

Twelve features determine the virus sensitive class for the potato samples used:

$\{$*13031, 13066, 19130, 23462, 24794, 25509, 29938, 33795, 33829, 35003, 35190, 36266*$\}$ $\rightarrow$ *sensitive*

Sixteen features determine the virus resistant class for the potato samples used:

$\{$*16441, 20474, 20671, 24030, 25141, 29777, 30111, 32459, 33225, 33248, 33870, 34108, 34114, 34388, 37252, 37484*$\}$ $\rightarrow$ *resistant*

When comparing our results with the SD algorithm for subgroup discovery (Gamberger and Lavrač, 2002), we observe that the running time of SD degrades considerably due to the high dimensionality of this data set. Moreover, SD obtains a larger set of rules which are less interpretable and do not have the same quality as the rules obtained with RelSets. Table 6 shows the numbers of discovered rules, area under ROC curve and the running time of both algorithms.

The results obtained with RelSets were validated by the experts from the National Institute of Biology, Ljubljana, Slovenia, and evaluated as insightful. Based on the tested samples, the experts have observed that the response to the infection after 8 hours is not strong enough to distinguish between resistant transgenic lines and sensitive ones. None of the gene expression changes after 8 hours appeared significant for the RelSets algorithm. However, selected gene expression levels after 12 hours and the comparison of gene expression difference (12-8) characterize the resistance to the infection with potato virus for the transgenic lines tested.[3]

3. Details of this analysis are beyond the scope of this paper: first qualitative analysis results have appeared in Kralj et al. (2006), while a more thorough analysis is to appear in a biological journal.

7. Conclusions

We have presented a theoretical framework that, based on the covering properties of closed itemsets, characterizes those sets of features that are relevant for discrimination. We call them closed sets for labeled data, since they keep similar structural properties of classical closed sets, yet taking into account the positive and negative labels of examples. We show that these sets define a nonredundant set of rules in the ROC space.

This study extends previous results where the notion of relevancy was analyzed for single features (Lavrač and Gamberger, 2005; Lavrač et al., 1999), and it provides a new formal perspective for relevant rule induction. In practice the approach shows major advantages for compacting emerging patterns and essential rules and solving hard subgroup discovery problems. Thresholds on positives make the method tractable even for large databases with many features. The application to potato microarray data, where the goal was to find differences between virus resistant and virus sensitive potato transgenic lines, shows that our approach is not only fast, but also returns a small set of rules that are meaningful and easy to interpret by domain experts.

Future work will be devoted to adapting efficient algorithms of emerging patterns by Dong and Li (1999) for the discovery of the presented relevant sets.

Acknowledgments

This work was partially funded by the Pascal Network of Excellence through a visit of the first author to the Jožef Stefan Institute, Ljubljana, Slovenia, and the Slovenian Research Agency grant Knowledge Technologies (2004–2008). We wish to thank Ana Rotter, Nataša Toplak and Kristina Gruden from the National Institute of Biology, Ljubljana, Slovenia, for the biological data and the joint research on the application of closed sets in functional genomics.

References

R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. *Advances in Knowledge Discovery and Data Mining*, pages 307–328, 1996.

J.L. Balcázar and J. Baixeries. Discrete deterministic datamining as knowledge compilation. In *SIAM Int. Workshop on Discrete Mathematics and Data Mining*, 2003.

E. Baralis and S. Chiusano. Essential classification rule sets. *ACM Trans. Database Syst.*, 29(4): 635–674, 2004.

Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. *Lecture Notes in Computer Science*, 1861:972–986, 2000a.

Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explor. Newsl.*, 2(2):66–75, 2000b.

S.D. Bay and M.J. Pazzani. Detecting group differences: Mining contrast sets. *Data Min. Knowl. Discov.*, 5(3):213–246, 2001. ISSN 1384-5810.

J.F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Min. Knowl. Discov.*, 7(1):5–22, 2003. ISSN 1384-5810.

S. Brin, R. Motwani, J.D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings ACM SIGMOD Int. Conference on Management of Data*, pages 255–264, 1997.

T. Calders and B. Goethals. Minimal *k*-free representations of frequent sets. In *Proceedings of the 7th European Conference on Principles and Knowledge Discovery in Data mining*, pages 71–82, 2003.

C. Carpineto and G. Romano. *Concept Data Analysis. Theory and Applications.* Wiley, 2004.

H.C. Causton, J. Quackenbush, and A. Brazma. *Microarray Gene Expression Data Analysis: A Beginner's Guide*. Blackwell Publishing, Oxford, United Kingdom, 2003.

P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

W. W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, 1995.

B. Crémilleux and J. F. Boulicaut. Simplest rules characterizing classes generated by delta-free sets. In *Proceedings of the 22nd Annual International Conference Knowledge Based Systems and Applied Artificial Intelligence*, pages 33–46, 2002.

G. Dong and J. Li. Efficient mining of emerging patterns: discovering trends and differences. In *Proceedings of the 5th Int. Conference on Knowledge discovery and data mining*, pages 43–52, 1999.

G. Dong, X. Zhang, L. Wong, and J. Li. CAEP: classification by aggregating emerging patterns. In *Proceedings of the 2nd In. Conference on Discovery Science*, pages 30–42, 1999.

D. Gamberger and N. Lavrač. Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17:501–527, 2002.

D. Gamberger, N. Lavrač, F. Železný, and J. Tolar. Induction of comprehensible models for gene expression datasets by subgroup discovery methodology. *Journal of Biomedical Informatics*, 37 (4):269–284, 2004.

B. Ganter and R. Wille. *Formal Concept Analysis. Mathematical Foundations*. Springer, 1998.

G.C. Garriga, P. Kralj, and N.Lavrač. Closed sets for labeled data. In *Proceedings of the 10th Int. Conference on Principles and Knowledge Discovery on Databases*, pages 163–174, 2006.

B. Goethals and M. Zaki. Advances in frequent itemset mining implementations: report on FIMI'03. *SIGKDD Explor. Newsl.*, 6(1):109–117, 2004.

J. Han and J. Pei. Mining frequent patterns by pattern-growth: methodology and implications. *SIGKDD Explor. Newsl.*, 2(2):14–20, 2000.

V. Jovanoski and N. Lavrač. Classification rule learning with APRIORI-C. In *Proceedings of the10th Portuguese Conference on Artificial Intelligence on Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving (EPIA '01)*, pages 44–51. Springer-Verlag, 2001.

B. Kavšek and N. Lavrač. APRIORI-SD: Adapting association rule learning to subgroup discovery. *Applied Artificial Intelligence*, To appear, 2006.

D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the 13th Int. Conference on Machine Learning*, pages 284–292, 1996.

P. Kralj, A. Grubešič, K. Gruden N. Toplak, N. Lavrač, and G.C. Garriga. Application of closed itemset mining for class labeled data in functional genomics. *Informatica Medica Slovenica*, 2006.

N. Lavrač and D. Gamberger. Relevancy in constraint-based subgroup discovery. *Constraint-Based Mining and Inductive databases*, 3848:243–266, 2005.

N. Lavrač, D. Gamberger, and V. Jovanoski. A study of relevance for learning in deductive databases. *Journal of Logic Programming*, 40(2/3):215–249, 1999.

N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004.

J. Li, G. Dong, and K. Ramamohanarao. Instance-based classification by emerging patterns. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 191–200, 2000.

B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of the 4th Int. Conference on Knowledge Discovery and Data Mining*, pages 571–574, 1998.

H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, 1998.

M. Molla, M. Waddell, D. Page, and J. Shavlik. Using machine learning to design and interpret gene-expression microarrays. *AI Magazine*, 25(1):23–44, 2004.

G. Parmigiani, E.S. Garrett, R.A. Irizarry, and S.L. Zeger, editors. *The Analysis of Gene Expression Data: Methods and Software*. Springer-Verlag, New York, 2003.

N. Pasquier, Y. Bastide, R. Taouil L., and Lakhal. Closed set based discovery of small covers for association rules. *Networking and Information Systems*, 3(2):349–377, 2001.

J.L. Pfaltz. Closure lattices. *Discrete Mathematics*, 154:217–236, 1996.

J.L. Pfaltz and C.M. Taylor. Scientific knowledge discovery through iterative transformations of concept lattices. In *SIAM Int. Workshop on Discrete Mathematics and Data Mining*, pages 65–74, 2002.

F.J. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.

A. Soulet, B. Crémilleux, and F. Rioult. Condensed representation of eps and patterns quantified by frequency-based measures. In *Proceedings of Knowledge Discovery in Inductive Databases Workshop*, pages 173–190, 2004.

T.P. Speed, editor. *Statistical Analysis of Gene Expression Microarray Data*. Chapman & Hall/CRC, Boca Raton, 2003.

L. Taiz and E. Zeiger. *Plant Physiology*. Sinauer Associates, second edition (372:374) edition, 1998.

P-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Mining bases for association rules using closed sets. In *Proceedings of the 16th Int. Conference on Data Engineering*, page 307. IEEE Computer Society, 2000.

T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *Discovery Science*, pages 16–31, 2004.

I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2005.

M. Zaki. Generating non-redundant association rules. In *Proceedings of the 6th Int. Conference on Knowledge Discovery and Data Mining*, pages 34–43, 2000a.

M. Zaki. Mining non-redundant association rules. *Data Mining and Knowledge Discovery: An International Journal*, 4(3):223–248, 2004.

M. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000b.

M. Zaki and M. Ogihara. Theoretical foundations of association rules. In *SIGMOD-DMKD Int. Workshop on Research Issues in Data Mining and Knowledge Discovery*, 1998.

J. Zhang, E. Bloedorn, L. Rosen, and D. Venese. Learning rules from highly unbalanced data sets. In *Proceedings of the 4th. IEEE Int. Conference on Data Mining (ICDM'04)*, pages 571–574, 2004.

Journal of Machine Learning Research 9 (2008) 581-621 Submitted 1/07; Revised 2/08; Published 4/08

Learning Reliable Classifiers From Small or Incomplete Data Sets: The 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: Charles Elkan

Abstract

In this paper, the naive credal classifier, which is a set-valued counterpart of naive Bayes, is extended to a general and flexible treatment of incomplete data, yielding a new classifier called *naive credal classifier 2* (NCC2). The new classifier delivers classifications that are reliable even in the presence of small sample sizes and missing values. Extensive empirical evaluations show that, by issuing set-valued classifications, NCC2 is able to isolate and properly deal with instances that are hard to classify (on which naive Bayes accuracy drops considerably), and to perform as well as naive Bayes on the other instances. The experiments point to a general problem: they show that with missing values, empirical evaluations may not reliably estimate the accuracy of a traditional classifier, such as naive Bayes. This phenomenon adds even more value to the robust approach to classification implemented by NCC2.

Keywords: naive Bayes, naive credal classifier, imprecise probabilities, missing values, conservative inference rule, missing at random

1. Introduction

Is it possible to draw credible conclusions about a domain only looking at some data produced within the domain itself?

The answer to this question appears to be related to the problem of modeling *ignorance*.[1] In fact, there are at least two kinds of ignorance involved in the process of learning from data. The first is *prior ignorance* about the domain, as we are assuming that data are our only source of information. The second is ignorance arising from missing values, as data are often incomplete; in this case, ignorance is about the process that originates the missing values: that is, the *missingness process*. So, in principle, we should model both ignorances properly in order to deliver credible conclusions.

Let us consider *pattern classification*, which is the focus of this paper. Considering *Bayesian classifiers*, we see that prior ignorance is modeled in common practice by so-called non-informative *prior densities* (or just *priors*, for short). But such an approach can lead, especially when the learning set is *small*, to the known problem of prior-dependent classifications, whose reliability is

1. When we use the word "ignorance" in this paper, we actually mean a condition of *near-ignorance*. Indeed, full ignorance is not compatible with learning, as it is well known (e.g., see Section 7.3.7 of Walley, 1991, Section 2.3 of Zaffalon, 2005b).

questionable. This appears to indicate that non-informative priors do not model prior ignorance satisfactorily.

A more objective-minded[2] model of prior ignorance has been proposed through a classifier called *naive credal classifier* (NCC), see Zaffalon (2001), which is an extension of *naive Bayes classifier* (NBC) to *imprecise probabilities* (Walley, 1991). NCC models prior ignorance by a *set* of prior densities (also called prior *credal set*), which is turned into a set of posteriors by element-wise application of Bayes' rule. The classification is eventually issued by returning all the classes that are *non-dominated* by any other class according to the posterior credal set, where class c_i is said to dominate c_j if *for all* the posteriors it holds that the probability of c_i is larger than that of c_j. This makes NCC naturally issue *set-valued* classifications (i.e., classifications made by more than one class) when faced with instances that are hard to classify, due to a combination of prior ignorance and poor information *about those specific instances* in the learning set. The shift of paradigm based on set-valued classifications allows NCC to deliver robust classifications in spite of small learning sets. NCC has indeed shown excellent accuracy in real-world case studies (Zaffalon, 2005a; Zaffalon et al., 2003), thus demonstrating the usefulness, for classification purposes, of modeling prior ignorance via a credal set. In the following, set-valued classifications are also called *indeterminate*. *Determinate* classifications correspond instead to the set being a singleton, and hence to the case usually considered by more traditional classifiers. Similarly, we say that a classifier is determinate when it outputs a single class and indeterminate otherwise.

As for the ignorance arising from missing data, we can think of the missingness process (MP) as a process that takes in input the complete data, which we cannot usually observe, and outputs the incomplete data, which we do observe. If data are our only source of information, we are ignorant about the MP because it is usually not possible to learn how it operates, from the observed, incomplete data.

In common practice, missing values are often *ignored*; this entails the idea that the MP is non-selective in producing them, or, in other words, that it is a *missing at random* (MAR) process (Little and Rubin, 1987; Jaeger, 2005). However, if one is ignorant about the MP, assuming MAR cannot be regarded as an objective-minded approach, as is well documented, for instance, by Manski (2003).

In its original formulation, NCC introduced also an initial attempt to deal with ignorance about the MP. The idea was to model ignorance about it by using a set of *likelihoods*: a likelihood per each complete learning set consistent with the incomplete one. A similar avenue was also implemented by *robust Bayes classifier* (Ramoni and Sebastiani, 2001).[3] These approaches are indeed valuable, but have two problems: (i) they implicitly still assume MAR for the missing values in the instance to classify, thus creating a peculiar asymmetry between learning and test set that is not of general validity in applications; (ii) they may well be too conservative, because for some feature variables one might know that the missingness is MAR, and they do not allow this information to be incorporated in the model. Furthermore, their treatment of missing values rests on intuitive arguments rather than on a principled derivation.

2. Although we base our results on Walley's theory, which is a *subjective* theory of probability, we sometimes use the terminology "objective-minded." We do so to stress that using (very) weak assumptions leads to results that are much more determined by the data than by our prior beliefs (intended in a loose way, not only as prior probabilities), and in this sense are more objective. Yet, in the paper we deliberately avoid using the word "objective" alone, just because it is improper within the considered theory.
3. When used with the so-called *strong dominance score*.

By this paper we extend NCC to a very general and flexible treatment of incomplete data, both in learning and testing. We call the resulting classifier *naive credal classifier 2* (NCC2), in order to emphasize the advancement made to deal with incomplete data, while keeping the original benefits of NCC on the front of prior ignorance.

By NCC2, it is possible to declare that some (possibly all or none) of the feature variables are subject to a MAR process, and the remaining ones are automatically assumed to be subject to an MP that is unknown to us. Remarkably, the set of feature variables subject to a MAR MP can be chosen differently from the learning to the test set. This is a key characteristic of NCC2: in fact, if the MP is unknown, it may well change its behavior from unit to unit for all we know (i.e., it may not be *identically distributed*), and we should act accordingly.

The development of NCC2 is based on a recently derived so-called *conservative inference rule* (CIR) to compute (imprecise) conditional expectations with incomplete data (Zaffalon, 2005b). After giving some notation and briefly recalling CIR in Section 2, we derive NCC2 in Section 3 by specializing CIR to the case of naive classification. In the end we obtain procedures to learn NCC2 and to do classifications with it that do not involve approximations and are computationally fast. (The software which implements NCC2 is released as open source; more details are provided in Section 3.5.)

Next, we concentrate on empirical evaluations: in Section 4 we analyze the behavior of NCC2 from a number of angles and on a number of publicly available data sets. The analysis turns out to be particularly meaningful when we compare NCC2 with its precise-probability counterpart, that is, naive Bayes. We do this by evaluating the accuracy of NBC on the instances of the test set where NCC2 issues a determinate classification separately from those where it does not. In fact, NCC2 is indeterminate on an instance when it deems that there is not enough knowledge in the learning set to make a determinate classification reliably; NBC, on the other hand, issues a determinate classification on such an instance (as well as on any other). Therefore we expect NBC to have different behaviors on the two kinds of instances isolated by NCC2. And this is indeed the case: the experiments show that NBC undergoes a major drop in accuracy moving from the instances classified in a determinate way by NCC2 to the indeterminate ones. The drop is observed on every data set, with no exception.

It is important to realize that such a drop points out a key question: the usual way to measure the performance of a classifier, that is, its predictive accuracy, which is an average over all the instances of the test set, may not help uncover a possible bad performance of the classifier on a subset of the test instances. These instances are precisely those that are hard to classify and that NCC2 isolates by delivering set-valued classifications.

But set-valued classifications help NCC2 to do more than just isolating the hard instances, they enable it to cope effectively with them: in fact, we show that set-valued classifications are often informative, as they usually lead to drop some unlikely classes; and that the measured *set-based* accuracy of NCC2 (i.e., the proportion of times the true class is contained in the output set) is often similar to the accuracy obtained on the instances classified in a determinate way.

At this point we should say that the mentioned experiments have been carried out with a variety of settings, obtained considering both MAR processes and non-MAR ones, and the mentioned outcomes are been confirmed over all of them (although sometimes this is due more to prior ignorance and some others more to the missing data).

To make our results stronger, we have also investigated whether NBC could take advantage of the posterior probabilities it computes in order to deal more successfully with the hard instances.

We have considered such posterior probabilities again by separating the cases where NCC2 is determinate from the others, and by comparing those probabilities with the measured accuracy that NBC actually achieves on the data. What we show is that the NBC probabilities are (also very) unreliable on the instances that are hard to classify, as isolated by NCC2, and definitely more unreliable than on the remaining instances. In other words, we observe another kind of drop that now is related to the quality of the posterior probabilities computed by NBC.

Overall, we show that NBC may well be too optimistic in dealing with small data sets and missing data, thus yielding unreliable predictions. It is useful to recall that NBC is known to be very robust to missing data. Therefore, it is not unlikely that the optimism on the front of missing data is even greater with more complex classifiers. This point appears to be worth of serious consideration on its own.

At the same time, and in contrast with naive Bayes, our experiments show that NCC2 may sometimes be too pessimistic (i.e., conservative) especially when dealing with missing data. This happens because by construction NCC2 implicitly considers the worst possible MP to have acted on the non-MAR part of the data, and in some cases this hypothesis may be too far from the MP that has actually produced the missing values.

In most of our experiments, for instance, we have deliberately used very simple MPs, and this has favored some excess of caution to show up. Yet, we have also considered an illustrative example of a more elaborated MP, in Section 4.6. In this case we show that the indeterminacy of NCC2 is fully justified, the NBC being completely unreliable in the area of indeterminacy. Even more important, we show that such an unreliability *cannot be uncovered by making empirical evaluations*: despite the predictive accuracy of NBC on a certain instance is measured properly by cross-validation, the actual accuracy on new instances of the same type can be significantly worse. This highlights the fact that modeling ignorance properly is important, even if there are data available for empirical evaluations. Indeed, in such an experiment NCC2 does not decrease its performance, nor do its empirical evaluations fail.

This is not to say that one should abuse assuming ignorance about the MP: when there are many missing values, especially in the instance to classify, one would obtain conclusions much too weak. This is shown in a setup where missingness increases in Section 4.5. Our view is that information about the MP should be incorporated in a model when available. In fact, we regard as an important research avenue the definition of classification models able to flexibly incorporate MP-related knowledge, which is usually not conveyed by the data at hand. With NCC2, incorporating knowledge is done by declaring that some variables are subject to a MAR MP, and we actually recommend doing so whenever possible; this has the potential, shown also experimentally in the above section, to yield strong enough conclusions in many cases. Eventually, in Section 4.7 we analyze the results obtained on the eucalyptus data sets; in fact, such results are quite peculiar and therefore worthy of a separate investigation. The conclusion we achieve in this case is that the use of coarsened rather than missing observations is another very effective means to incorporate knowledge, in case there is no support for declaring some variables as subject to a MAR MP.

2. Setup

In this paper, the variables that refer to complete data, which are in general not observable, are called *latent*, while those referring to incomplete data, which are the ones to which we have usually access, are called *manifest*. A given manifest value is identical to the corresponding latent one, unless the

latent value has been turned into missing by the MP; in this case, the manifest value is actually the symbol of missing value. Therefore, in a case where the data at hand are complete, the instances of the latent and the manifest variables coincide.

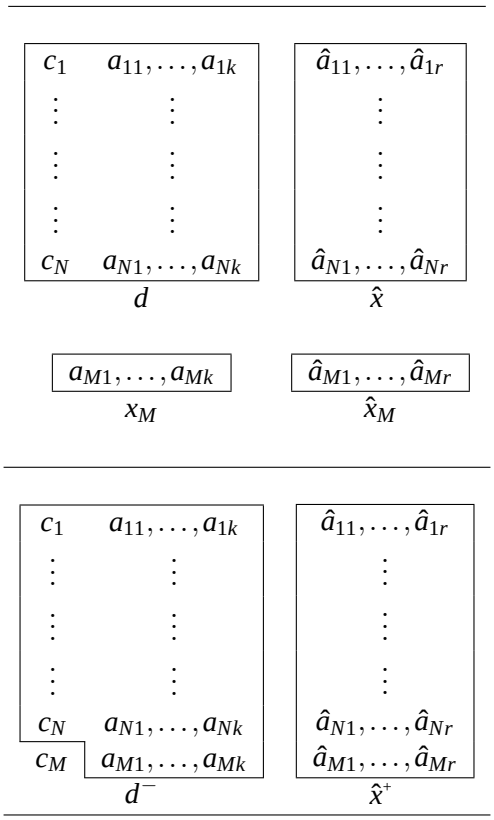


Figure 1: Graphical representation of some vectors of latent variables. Rows $1,\ldots,N$ constitute the training set, while the M-th unit is a new instance to be classified.

In the following, i indexes a given unit (i.e., a certain row) of the data set: the *learning set* (or *training set*) is made up of the units for which $1 \leq i \leq N$, while the unit to classify (not belonging to the learning set) is indexed by $M := N+1$. A set of units to classify is referred to as *test set*.

In a classification problem there are typically *class variables* and *attribute variables*. We denote: (i) the latent *class variable* as C_i, and we assume that it is always observed; (ii) the latent attribute variables affected by an unknown MP (i.e., to be conservatively modeled as non-MAR) as $A_{i1},\ldots,A_{ik}$; (iii) the latent attributes affected by a MAR MP as $\hat{A}_{i1}$, …, $\hat{A}_{ir}$. The two MPs are assumed to be independent of each other and their coarsening behavior is allowed to vary with different units, that is, they are *not* assumed to be identically distributed.[4]

For all i, C_i takes generic value c_i in the finite set $\mathcal{C}$, called *set of latent classes*, while A_{ij} ($\hat{A}_{il}$) take generic values a_j ($\hat{a}_l$) in the finite sets $\mathcal{A}_j$ ($\hat{\mathcal{A}}_l$), called *sets of latent attributes*.

We define the following groups of latent variables: $X_i := (A_{i1},\ldots,A_{ik})$, $D_i := (C_i, X_i)$, and $\hat{X}_i := (\hat{A}_{i1},\ldots,\hat{A}_{ir})$. We then extend such grouped variables to span the whole training set, instead than

4. See Zaffalon (2005b), Section 5, for a discussion about this point.

just the *i*-th unit, defining the vector $C := (C_1,\ldots,C_N)$ and the matrices $X := (X_1,\ldots,X_N)$, $\hat{X} := (\hat{X}_1,\ldots,\hat{X}_N)$, $D := (D_1,\ldots,D_N)$. The same grouped variables but with a "+" superscript, include also data of the M-th unit (i.e., the instance to be classified): $C^+ := (C_1,\ldots,C_M)$, $X^+ := (X_1,\ldots,X_M)$, $\hat{X}^+ := (\hat{X}_1,\ldots,\hat{X}_M)$,$D^+ := (D_1,\ldots,D_M)$. Let also define $D^- := (C,X^+)$.[5]

Observe that realizations of the random matrix $(D^+,\hat{X}^+)$ represent the possible complete data sets, which we cannot observe directly, and that realizations of $(D,\hat{X})$ represent the possible complete learning sets, while those of $(X_M,\hat{X}_M)$ represent the possible complete units to classify.

To complete the notation regarding latent variables, we assume that the generic latent unit $(d,\hat{x}) \in \mathcal{D}\times\hat{\mathcal{X}}$ is generated in *independently and identically distributed* (IID) way according to the *aleatory probability* (or *chance*) $\vartheta_{(d,\hat{x})}$. The vector of such chances is denoted by ϑ, which belongs to Θ, that is, a (non-empty) subset of the unitary $|\mathcal{D}\times\hat{\mathcal{X}}|$-dimensional simplex. Let θ denote the random variable of which ϑ is a generic value. Knowledge about θ is expressed by $p(\theta)$, which denotes an imprecise prior density for θ. This means that $p(\theta)$ is known to belong to a non-empty set $\mathcal{P}(\theta)$ of precise prior densities for θ. $\mathcal{P}(\theta)$ is referred to as *prior credal set*.

As for the manifest variables, we assume that we either observe a precise value, or we do not observe it at all. Manifest variables are denoted by the letter O followed by the latent variable they refer to, written as a subscript. We define hence the following manifest variables: $O := O_D = (O_1,\ldots,O_N)$, $O^+ := O_{D^+} = (O_1,\ldots,O_M)$, $O^- := O_{D^-} = (O_1,\ldots,O_N,X_M)$, $\hat{O} := O_{\hat{X}} = (\hat{O}_1,\ldots,\hat{O}_N)$, $\hat{O}^+ := O_{\hat{X}^+} = (\hat{O}_1,\ldots,\hat{O}_M)$.

2.1 Classification With Imprecise Probabilities and Conservative Inference Rule

The goal of classification is to predict the class of the M-th unit, given the previous units $(1,\ldots,N)$ and the values of the M-th attribute variables.

To this extent, a traditional probabilistic classifier outputs what it deems to be the optimal prediction: that is, the class with the highest probability (in the case of 0-1 loss function) on the basis of a uniquely computed posterior density. In the imprecise setting, however, the optimality criterion has to be extended to manage a set of posterior densities (derived from a set of priors and a set of likelihoods), instead of a single posterior; in particular, according to Section 3.9.2 of Walley (1991), the optimality criterion in the imprecise setting prescribes to return the *non-dominated* classes. The definition of dominance is as follows: class c_i dominates c_j if for all the computed posteriors densities, the posterior probability of c_i is greater than that of c_j; clearly, c_j is non-dominated if no class dominates c_j. The second procedure of Figure 2, based on pairwise comparison of classes, identifies the non-dominated classes. Observe that, as a result of the uncertainty arising from both prior specification and non-MAR missing values, there can be several non-dominated classes; in this case, the classifier returns an indeterminate (or set-valued) classification. Classifiers that issue set-valued classifications are called *credal classifiers* by Zaffalon (2002).

A key point is that non-dominated classes are *incomparable*;[6] this means that there is no information in the model that allows us to rank them. In other words, credal classifiers are models that allow us to drop the dominated classes, as sub-optimal, and to express our indecision about the optimal class by yielding the remaining set of non-dominated classes.

5. Since we assume the class variable to be always observed, C is a complete vector. Hence, it is possible to derive the NCC2 algorithms also by grouping C with $\hat{X}$.
6. If we exclude the classes that are non-dominated due to indifference rather than incomparability, and that constitute a very special case in the imprecise setting.

In the setup of this paper, the test of dominance can be re-written as follows: c'' is dominated by c' if and only if it holds that

$$1 < \min_{x_M \in o_M} \min_{d \in o} \inf_{p(\theta) \in \mathcal{P}(\theta)} \frac{p(c'_M | d^-, \hat{x}^+ \in \hat{o}^+)}{p(c''_M | d^-, \hat{x}^+ \in \hat{o}^+)}. \quad (1)$$

Actually, Equation (1) is the general form of the test of dominance for any classifier based on the conservative inference rule presented in Zaffalon (2005b); CIR is a conditioning rule (i.e., a rule for computing conditional expected values) that generalizes the traditional conditioning; it assumes that prior beliefs are dealt with via a credal set $\mathcal{P}(\theta)$ and it accounts for data sets in which the missingness process is MAR for some variables (the term $\hat{x}^+ \in \hat{o}^+$ refers indeed to the missing data of MAR feature variables in the training set), and unknown for some others. Moreover, CIR is able to manage variables whose MP is MAR in learning and unknown in testing, or vice versa. Equation (1) follows almost immediately from Theorem 4 in Zaffalon (2005b) after considering that for c'_M to dominate c''_M, it must hold that $p(c'_M | d^-, \hat{x}^+ \in \hat{o}^+) > p(c''_M | d^-, \hat{x}^+ \in \hat{o}^+)$ for all the possible posteriors, which we obtain considering any precise prior in the prior credal set, as well as any completion of the non-MAR missing values both in the sample and in the unit to classify.

CIR can be regarded as unifying two rules (Zaffalon, 2005b): a *conservative learning rule*, which prescribes how to learn the classifier from an incomplete training set, and a *conservative updating rule*, which prescribes how to classify a novel instance that contains missing values. Such a distinction is made clear by two distinct optimization loops of Equation (1); the middle optimization loop ($\min_{d \in o}$) realizes the conservative learning rule, by prescribing to loop on the completions of the non-MAR part of the learning set, that is, $d \in o$, while the outer minimum implements the conservative updating rule, prescribing to loop on the replacements for the non-MAR missing values of the unit to classify. The inner loop, which minimizes over the prior credal set, is common to both learning and updating rules.

NCC2 specializes the test of Equation (1) to the case of naive classification. In the following, we will move from the precise setting (corresponding in fact to naive Bayes) to NCC2 in four steps: in Section 3.1 we describe the precise setting (assuming hence that there is a single prior, and that there is a single likelihood as non-MAR data are complete); in Section 3.2 we extend the computation to manage a set of priors; in Sections 3.3 and 3.4 we finally relax the assumptions of completeness about non-MAR data in learning and testing respectively, thus managing a set of likelihoods and instances to classify.

3. Introducing NCC2

In this section, we introduce the NCC2 framework. In particular, we derive an expression that specializes the test of Equation (1) to the case of naive classification; such an expression realizes both the minimizations over possible priors and over possible unobserved values by distributing some pseudo-counts in a way that minimizes the ratio of the posterior probabilities of the two competing classes c'_M and c''_M.

Formal proofs of the findings derived in this section are provided in Appendix B.

3.1 Updating Precise Beliefs

In this section, we assume that a single prior is specified and that only the observations of the features affected by the MAR MP contain some missing data; the observations of the feature affected by the

unknown MP, instead, do not contain any missing data. In practice, this corresponds to the naive Bayes setting, with the difference that we explicitly separate variables affected by the MAR MP and the unknown MP. In our setting, the naive hypothesis (i.e, the assumption of mutual independence of the latent attribute variables $A_{i1}, \ldots, A_{ik}, \hat{A}_{i1}, \ldots, \hat{A}_{ir}$ conditional on the class variable C_i) can be formalized as follows:

$$\vartheta_{(d_i,\hat{x}_i)} = \vartheta_{c_i} \prod_{j=1}^{k} \vartheta_{a_{ij}|c_i} \prod_{l=1}^{r} \vartheta_{\hat{a}_{il}|c_i} \quad \forall (d_i,\hat{x}_i) \in \mathcal{D} \times \hat{\mathcal{X}}, \tag{2}$$

where ϑ_c denotes the chance of $(C_i = c_i)$; $\vartheta_{a_{ij}|c}$ and $\vartheta_{\hat{a}_{il}|c}$ denote the chances of $(A_{ij} = a_j | C_i = c_i)$ and $(\hat{A}_{il} = \hat{a}_l | C_i = c_i)$, respectively.

We focus on the precise probability $p(c_M | d^{\text{-}}, \hat{x}^{+} \in \hat{o}^{+})$. Let us consider the probability $p(c_M, d^{\text{-}}, \hat{x}^{+} \in \hat{o}^{+})$, which is proportional to the previous one. Write $p(c_M, d^{\text{-}}, \hat{x}^{+} \in \hat{o}^{+})$ as $\int_\Theta p(\vartheta) p(c_M, d^{\text{-}}, \hat{x}^{+} \in \hat{o}^{+} | \vartheta) d\vartheta$. Observe that $p(c_M, d^{\text{-}}, \hat{x}^{+} \in \hat{o}^{+} | \vartheta)$ is equal to $p(d, \hat{x} \in \hat{o} | \vartheta) p(c_M, x_M, \hat{x}_M \in \hat{o}_M | \vartheta)$ because of the IID assumption about the data generation mechanism. We obtain that

$$p(c_M | d^{\text{-}}, \hat{x}^{+} \in \hat{o}^{+}) \propto \int_\Theta p(\vartheta) p(d, \hat{x} \in \hat{o} | \vartheta) p(c_M, x_M, \hat{x}_M \in \hat{o}_M | \vartheta) d\vartheta. \tag{3}$$

The term $p(d, \hat{x} \in \hat{o} | \vartheta)$ in (3) is called *likelihood function*.

By sticking to the naive hypothesis, the likelihood can be expressed as follows:

Lemma 1
$p(d, \hat{x} \in \hat{o} | \vartheta) = \prod_{c \in \mathcal{C}} \{ \vartheta_c^{n(c)} [\prod_{j=1}^{k} \prod_{a_j \in \mathcal{A}_j} \vartheta_{a_j|c}^{n(a_j,c)}] [\prod_{l=1}^{r} \prod_{\hat{a}_l \in \hat{\mathcal{A}}_l} \vartheta_{\hat{a}_l|c}^{n(\hat{a}_l,c)}] \}$.

Here $n(c)$ resp. $n(a_j, c)$ denote the number of occurrences of c resp. of joint occurrences of (a_j, c) in d, and $n(\hat{a}_l, c)$ denotes the number of joint occurrences of $(\hat{a}_l, c)$ in the learning set after dropping the units with missing values of $\hat{A}_l$. Technically, the likelihood function of Lemma 1 has the same functional form as a product of Dirichlet densities; in particular, the frequencies $n(\cdot)$ correspond to the Dirichlet hyperparameters usually denoted as $\alpha(\cdot) - 1$.

With similar arguments to those used with Lemma 1, and assuming that the MAR attribute variables have been re-ordered so as to index the non-missing ones in the instance to classify from 1 to $r' \leq r$, we obtain:

Lemma 2 $p(c_M, x_M, \hat{x}_M \in \hat{o}_M | \vartheta) = \vartheta_{c_M} \prod_{j=1}^{k} \vartheta_{a_{Mj}|c_M} \prod_{l=1}^{r'} \vartheta_{\hat{a}_{Ml}|c_M}$.

Note that restricting the second product between $l = 1$ and $l = r'$ prevents the inclusion in the expression of the attributes that are missing in the unit to classify.

3.1.1 IMPRECISE PRIOR (PRIOR CREDAL SET)

The remaining term in (3) is $p(\vartheta)$, that is, the prior. We define it so as to be *conjugate* to the likelihood, according to the following expression:

$$p(\vartheta | s, t) d\vartheta \propto \prod_{c \in \mathcal{C}} \{ \vartheta_c^{st(c)-1} d\vartheta_C [\prod_{j=1}^{k} \prod_{a_j \in \mathcal{A}_j} \vartheta_{a_j|c}^{st(a_j,c)-1} d\vartheta_{A_j|c}] \cdot \cdot [\prod_{l=1}^{r} \prod_{\hat{a}_l \in \hat{\mathcal{A}}_l} \vartheta_{\hat{a}_l|c}^{st(\hat{a}_l,c)-1} d\vartheta_{\hat{A}_l|c}] \}, \tag{4}$$

which is a product of Dirichlet densities. The density is named $p(\vartheta|s,t)$, so as to make it explicitly the hyperparameters on which it depends (here t denotes the vector of $t(\cdot)$-hyperparameters). In other words, the prior is defined by an expression that is similar to the likelihood function except that the frequencies $n(\cdot)$ are replaced everywhere by $st(\cdot)-1$. The real hyperparameter s can be regarded as the size of the *hypothetical sample*, in the common interpretation of conjugate Bayesian priors as additional sample units; Walley (1996) gives arguments to choose s in the interval $[1,2]$, and we will adopt $s := 1$ for the empirical experiments of Section 4. The real hyperparameter $t(\cdot)$ can instead be regarded as the proportion of units in which the variables in question take certain values (e.g., $t(c)$ is the proportion of units where the class variable is equal to c) in the hypothetical sample.

Now, remember that we want the prior to be imprecise, that is, to be a set of priors. We define the set by imposing a system of constraints on the t-hyperparameters that resemble the structural constraints of the observed frequencies $n(\cdot)$: in particular, $\sum_{c\in\mathcal{C}} t(c) = 1$, $\sum_{a_j\in\mathcal{A}_j} t(a_j,c) = t(c)$, $\sum_{\hat{a}_l\in\hat{\mathcal{A}}_l} t(\hat{a}_l,c) = t(c)$. We moreover impose the conditions $t(a_j,c) > 0$, $t(\hat{a}_l,c) > 0$. The credal set $\mathcal{P}(\theta)$ is defined as the set of all the precise priors that satisfy these constraints. The construction of $\mathcal{P}(\theta)$ is similar to the approach implemented by Walleys' *imprecise Dirichlet model* (Walley, 1996).

Informally, one could interpret $\mathcal{P}(\theta)$ in the following way: say that any single, *precise* Dirichlet prior, is a possible state of information about the process generating (latent) data. Then, our ignorance about this process is modeled by considering the set of all the states of information about the process, that is, by considering that all of them are actually possible.[7]

3.1.2 PROBABILITY OF THE NEXT CLASS

The tools introduced so far lead us to the following result.

Theorem 3 *Consider Expression* (3)*. It holds that:*

$$p(c_M|d^{\cdot},\hat{x}^{+}\in\hat{o}^{+},s,t) = p(c_M|d,\hat{x}\in\hat{o},s,t)\prod_{j=1}^{k} p(a_{Mj}|c_M,d,\hat{x}\in\hat{o},s,t)\cdot \cdot\prod_{l=1}^{r'} p(\hat{a}_{Ml}|c_M,d,\hat{x}\in\hat{o},s,t), \tag{5}$$

where

- $p(c_M|d,\hat{x}\in\hat{o},s,t) := [n(c_M)+st(c_M)]/(N+s)$*;*
- $p(a_{Mj}|c_M,d,\hat{x}\in\hat{o},s,t) := [n(a_{Mj},c_M)+st(a_{Mj},c_M)]/[n(c_M)+st(c_M)]$ *($j=1,\ldots,k$);*
- $p(\hat{a}_{Ml}|c_M,d,\hat{x}\in\hat{o},s,t) := [n(\hat{a}_{Ml},c_M)+st(\hat{a}_{Ml},c_M)]/[n_l(c_M)+st(c_M)]$ *($l=1,\ldots,r'$);*
- $n_l(c_M) := \sum_{\hat{a}_l\in\hat{\mathcal{A}}_l} n(\hat{a}_l,c_M)$.

Note that MAR attributes that are missing in the unit to be classified do *not* affect the posterior probabilities of the class.

7. To be more precise, we do not consider all of them, only those with s fixed; this corresponds to fix the "strength" of our prior ignorance.

3.2 Credal Dominance Tests with an Imprecise Prior

Let us now address the computation of the inner optimization problem in (1), under the choice of the imprecise prior made in Section 3.1.1.

Lemma 4 *Consider the problem* $\inf_{p(\theta)\in\mathcal{P}(\theta)} p(c'_M|d^-,\hat{x}^+\in\hat{o}^+)/p(c''_M|d^-,\hat{x}^+\in\hat{o}^+)$*, with the set of prior densities described in Section 3.1.1, and the probabilities in the function to optimize (also called* objective function *in the following) defined as in* (5)*. Such a problem is equivalent to the following:*

$$\begin{aligned}&\inf_{0<t(c_{M''})<1}\{[\frac{n(c''_M)+st(c''_M)}{n(c'_M)+s-t(c''_M)}]^{k-1}\prod_{j=1}^{k}\frac{n(a_{Mj},c'_M)}{n(a_{Mj},c''_M)+st(c''_M)}\\&\cdot\prod_{l=1}^{r'}[\frac{n_l(c''_M)+st(c''_M)}{n_l(c'_M)+s-t(c''_M)}\cdot\frac{n(\hat{a}_{Ml},c'_M)}{n(\hat{a}_{Ml},c''_M)+st(c''_M)}]\}\\&=:\inf_{0<t(c_{M''})<1}h(t(c_{M''})).\end{aligned}\tag{6}$$

The problem of finding the prior $p(\theta)\in\mathcal{P}(\theta)$ that minimizes the ratio of the posterior probabilities $p(c'_M|d^-,\hat{x}^+\in\hat{o}^+)/p(c''_M|d^-,\hat{x}^+\in\hat{o}^+)$ can be hence be solved by finding the value $0<t(c_{M''})<1$ that minimizes $h(t(c_{M''}))$, as follows.

Theorem 5 *The infimum of* $h(t(c_{M''}))$ *over* $(0,1)$ *is determined by the following procedure:*

- *if there is* j *such that* $n(a_{Mj},c'_M)=0$ *or* l *such that* $n(\hat{a}_{Ml},c'_M)=0$*,* $\inf h(t(c_{M''}))=0$*;*
- *if* $k=0$ *and* $r'=0$*,* $\inf h(t(c_{M''}))=h(1)$*;*
- *otherwise,* $h(t(c_{M''}))$ *can be shown to be convex over* $(0,1)$*; hence, it can be minimized by a convex optimization procedure (detailed in the proofs).*

3.3 Credal Dominance Test with an Incomplete, Non-MAR, Learning Set

In case the learning data produced by the unknown MP are incomplete, the problem to be solved is

$$\min_{d\in o}\inf_{p(\theta)\in\mathcal{P}(\theta)} p(c'_M|d^-,\hat{x}^+\in\hat{o}^+)/p(c''_M|d^-,\hat{x}^+\in\hat{o}^+).\tag{7}$$

Theorem 6 *The procedure of Theorem 5 solves Problem* (7) *after renaming* $n(a_{Mj},c'_M):=\underline{n}(a_{Mj},c'_M)$ *and* $n(a_{Mj},c''_M):=\overline{n}(a_{Mj},c''_M)$*, where* $\underline{n}(a_{Mj},c'_M):=\min_{d\in o}n(a_{Mj},c'_M)$ *and* $\overline{n}(a_{Mj},c''_M):=\max_{d\in o}n(a_{Mj},c''_M)$*.*

In other words, to solve Problem (7) we have to select the realization of the non-MAR part of the learning set, among the possible realizations $d\in o$ consistent with our observations, which minimizes the probability ratio.

For a non-MAR feature variable A_j, one can prove that:

- the probability of class c'_M is minimized by assuming the value of $\mathcal{A}_j$ to be different from a'_{Mj} whenever A_j is missing and the class of the instance is c'_M; this is the meaning of the lower counts $\underline{n}(a_{Mj},c'_M)$;[8]

8. Note that, despite the different meaning, the counts $\underline{n}(a_j,c)$ for non-MAR feature variables are computed identically to the counts $n_l(a_j,c)$ for MAR ones.

- the probability of class c''_M is maximized by assuming the value of A_j to be a_{Mj} whenever A_j is missing and the class of the instance is c''_M; this is the meaning of the upper counts $\overline{n}(a_{Mj}, c''_M)$.

Once the realization of the non-MAR missing values is identified by upper and lower counts, the most unfavorable complete realization of the learning set has been chosen (i.e., the minimization over $d \in o$ has been accomplished), and we are in the case of Problem (6).

3.4 Credal Dominance Test with an Incomplete, Non-MAR, Unit to Classify

Finally, we consider the case when the unit to classify is missing some of the attributes subject to the unknown MP. We need to address the following problem:

$$\min_{x_M \in o_M} \min_{d \in o} \inf_{p(\theta) \in \mathcal{P}(\theta)} \frac{p(c'_M | d^-, \hat{x}^+ \in \hat{o}^+)}{p(c''_M | d^-, \hat{x}^+ \in \hat{o}^+)}. \tag{8}$$

Problem (8) can be trivially solved as follows: (i) considering all the possible realizations x_M of the non-MAR part of the instance (this is accomplished by considering all the possible replacements for the missing values); (ii) for each $x_M \in o_M$, assuming x_M to be the realization of the non-MAR part of the unit to classify, and then solving Problem (7); (iii) if the computed solution is smaller than or equal to 1, c''_M is not dominated by c'_M (and the computation can be interrupted); instead, if the solution is greater than 1 for each x_M, c''_M is dominated by c'_M.

Although this procedure leads to the exact solution, it takes exponential time due to the number of the replacements of missing values in the instance to classify. A more efficient polynomial-time procedure, which is still exact, can be designed to solve Problem (8) and is in fact given in Appendix A.

The underlying idea of such a procedure is to split the outer minimum in (8) in many minima, each one related to a different feature variable, and to distribute them at different places into the objective function. This makes is clear that the function to optimize is the lower envelope of a set of convex functions. In Appendix A we show that it is easy to compute the points where the function that determines the envelope changes, thus in fact obtaining a partition of the envelope function's domain with the property that in each of its elements the envelope function is convex. At that point, the function can be locally optimized efficiently on every element of the partition; and the global optimum is then just the minimum of the local optima so computed. The fact that the overall procedure is polynomial follows because the size of the partition is bounded by a polynomial.

A final remark is that, to make notation simpler, we refer to the possible realizations of the non-MAR variables in the instance to be classified as o_M; such a notation implies however that the non-MAR variables of the test set are the same of the training set. If instead the non-MAR variables of the test set are different from those of the training set, o_M should contain all the possible realizations of the variables which are non-MAR in the test set.

The NCC2 procedures are summarized in Fig. 2. Learning has linear complexity with respect to the number of attributes, while testing has roughly quadratic complexity, if the procedure carried out in Appendix A is adopted. Please refer to Appendix A also for the exact expression for the complexity.

3.5 Software Availability

The software JNCC2, implemented by the authors of this paper, implements the naive credal classifier 2. JNCC2 is open source, released under the GNU GPL license; it is hence freely available

LEARNING

- list the attributes affected by a MAR MP or by an unknown MP on the learning set;
- compute on the learning set the counts $n(\hat{a}_l, c)$, $n_l(c)$ for MAR attributes and the counts $\underline{n}(a_j, c)$, $\overline{n}(a_j, c)$, $n(c)$ for non-MAR attributes.

CLASSIFICATION OF AN INSTANCE

1. set NonDominatedClasses := $\mathcal{C}$;
2. for class $c' \in \mathcal{C}$
   - for class $c'' \in \mathcal{C}$, $c'' \neq c'$
     - if c'' is dominated by c' (to be assessed via the below procedure), drop c'' from NonDominatedClasses;
     - exit;
   - exit
3. return NonDominatedClasses.

DOMINANCE TEST BETWEEN TWO CLASSES (c', c'')

- list the attributes affected by a MAR MP or by an unknown MP in the instance;
- for each $x_M \in o_M$ (i.e., for each possible realizations of the non-MAR part of the unit to classify):
  - assume x_M to be the realization of the non-MAR part of the instance;
  - solve Problem (7) via Theorem 6;
  - if the computed solution is smaller than 1, c'' is not dominated by c'. STOP.
- if, after having tried every $x_M \in o_M$, no solution greater than 1 has been found, c' dominates c''.

/**alternatively to the above exhaustive procedure (for each $x_M \in o_M$) , the minimum can be computed more efficiently via the procedure presented in Appendix A.*/

Figure 2: Summary of NCC2 procedures.

together with user manual, sources and documentation. Being written in Java, it runs under any operating system; it has a command-line interface.

JNCC2 loads data stored in the ARFF format, which is a textual format (designed for classification problem), originally developed for WEKA (Witten and Frank, 2005), an open source software for data mining. Hence, the large public repositories of ARFF files can be used to extensively test NCC2.

For downloads and further information about JNCC2, see `www.idsia.ch/~giorgio/jncc2.html`.

4. Experiments

We consider 18 data sets from the UCI repository, and available in the ARFF format from the WEKA data sets page.[9] All the data sets are complete, that is, they do not contain any missing data.

We discretized the numerical features via MDL-based discretization (Fayyad and Irani, 1993), and then we split each data set into a training and a test set. Feature discretization is necessary, as NCC2 is designed to work with categorical variables. Discretizing the features on the entire data set introduces a slight optimistic bias in the evaluation of the classifiers accuracy (in principle, the discretization intervals should be computed on the training set, and then applied unchanged in the test set); yet, this is not a problem as our goal is to compare NBC and NCC2 in identical conditions, rather than to compare our findings with previous results obtained on the same data sets.

Also, since our goal is to fairly compare NBC and NCC2, and not to finely tune them for maximum performance, we did not perform feature selection (although, in fact, we remove numerical feature variables discretized into a single bin); yet, as both NBC and NCC2 are based on the naive hypothesis, redundant or mutually dependent feature variables might significantly bias the learning process; hence, in order to achieve maximum performance, one should consider selecting feature variables.

In the experiments presented in the following we generate artificial missingness on the original, complete data sets by using different MPs and then we compare NBC and NCC2 accuracy on the incomplete data sets.

4.1 Missingness Generation

We consider two different artificial MPs: (i) a MAR one, and (ii) a non-MAR, non-identically distributed one (nonMAR). The MPs we consider do not affect the class variable, as the class variable has been assumed to be always observed.

Note that we do not test mixed cases of MAR and non-MAR feature variables, which NCC2 is actually designed to treat. In fact, such settings would simply lead to results intermediate between those obtained under the MAR and the non-MAR settings. Yet, mixed settings are valuable and should be considered whenever possible when operating the classifier, as they allow for finely tuning the treatment of missing data to the characteristics of the MP.

The MAR MP turns into missing, with 5% probability, all the features of both the training and test sets. Such a missingness process actually meets not only the definition of MAR, but also the more restrictive definition called MCAR, that is, *missing completely at random*, see Little and Rubin (1987). We have taken into consideration also a non-MCAR, MAR MP; however, the results do not differ significantly from those obtained with the MAR MP (which satisfies also MCAR) described above.

The non-MAR MP works as follows : (i) it splits the categorical values of each feature variable into two halves; (ii) for each feature variable, it turns into missing, with probability 5%, the observations falling in the first half of values, on the training set; (ii) for each feature variable, it turns

9. The URL is `http://www.cs.waikato.ac.nz/ml/weka/`.

into missing, with probability 5%, the observations falling in the second half of values, in the test set. Such MP is not identically distributed, as it follows a different pattern from training to test set.

We split each data set into equally sized training and test subsets. Using this training/test split, for each data set and for each MP we generate artificial missingness 100 times, producing hence 100 different training and test sets. The results we present are obtained as an average over 100 runs for each data set-MP pair.

4.2 Performance Measures

The performance of NBC is measured by its *accuracy*, that is, the percentage of correct classifications, while the evaluation of NCC2 requires a larger number of indicators: *determinacy*, that is, the percentage of classifications having as output a unique class; *single accuracy*, that is, accuracy of NCC2 when it is determinate; *indeterminate output size*, that is, the average number of classes returned when NCC2 is indeterminate; *set-accuracy*, that is, the percentage of indeterminate classifications that contain the true class. Note that if a data set has two classes, the output size is necessarily 2 and set-accuracy 100%; therefore, such two indicators are meaningful on data sets with more than two classes only.

We consider three classifiers: (i) NBC (with standard Laplace prior); (ii) NCC2-MAR, that is, NCC2 which assumes all missing values to be generated by a MAR MP; (iii) NCC2-nonMAR, that is, NCC2 which assumes all missing values to be generated by an unknown MP. Indeterminate classifications of NCC2-MAR are in fact due to prior uncertainty only, while indeterminate classifications of NCC2-nonMAR are due to both the prior and the missing values.

We then consider an additional set of indicators, which refer to the NBC accuracy on specific subsets of instances (note that such indicators, referring to subsets of instances, are written in italic):

- *NCC2-MAR D* and *NCC2-MAR I*, as the NBC accuracies on the instances classified respectively in a determinate and indeterminate way by NCC2-MAR; such indicators point out the effects of the prior specification on NBC;

- *NCC2-nonMAR D* and *NCC2-nonMAR I*, as the NBC accuracies on the instances classified respectively in a determinate and indeterminate way by NCC2-nonMAR; such indicators point out the joint effect of prior specification and missing values on NBC;

- Δ*NCC2-nonMAR*, as the NBC accuracies on the instances over which NCC2-nonMAR outputs a larger number of classes than NCC2-MAR; such an indicator points out the effect of missing values on NBC.

It turns out however that *NCC2-MAR D* coincides with the single-accuracy of NCC2-MAR, and that *NCC2-nonMAR D* coincides with the single-accuracy of NCC2-nonMAR: in fact, when determinate, NCC2-MAR and NCC2-nonMAR return the same output as NBC.[10] In the following we will hence only mention *NCC-MAR D* and *NCC-nonMAR D*; it is understood that the single-accuracy of respectively NCC2-MAR and NCC2-nonMAR coincides with such values.

10. Differences might arise only in case that the NBC prior is not included in the credal set, and that such a prior leads to a classification which is different from the classification to which leads any single prior of the credal set. The frequency of such an event is however negligible and therefore the above indicators can be considered to be the same.

4.3 Overview of the Results

Table 1 reports the performance of NBC, NCC2-MAR and NCC2-nonMAR under the different MPs, averaged over all the data sets. The average of set-accuracy and indeterminate output size have been computed considering only data sets with more than two classes. The detailed results data set by data set are instead shown in Tables 6 and 7, which refer respectively to the MAR and non-MAR setting.

NBC achieves an average accuracy of about 80%; although its accuracy seems to be insensitive to the MP, we show later however, through a deeper analysis, that this is not the case.

The average determinacy of NCC2-MAR is about 91%, that is, NCC2-MAR yields set-valued classification in 9% of cases; this indicator is mainly influenced by the data set size and therefore it does not show great differences between the different MP settings. The point here is that when we declare all the feature variables to be MAR, imprecision in the probabilities is only originated by the imprecise prior density, as it follows from (1). Such a kind of imprecision strictly decreases with the size of the learning set because the prior, as in the precise Bayesian setting, counts less and less with more data.

The results detailed data set by data set (Tables 6 and 7) show that NCC2-MAR has considerably lower determinacy on the glass data set than on any other data set. This is a consequence of the low number of training instances (about 100) and high number of output classes (i.e., 7). In these conditions the prior has much weight and it originates more imprecision.

The determinacy of NCC2-nonMAR ranges from 33% under the MAR setting, to 50–55% under the non-MAR settings. The higher indeterminacy under the MAR setting for NCC2-nonMAR is due to the higher number of missing data: in fact, the MAR MP produces twice as many missing data as the non-MAR MPs, because it turns *all* the values of the feature variables into missing, with probability 5%, instead of half the values as the non-MAR MP does.

Looking at the results data set by data set, we can detect three factors that increase the indeterminacy of NCC2-nonMAR: (a) high prior uncertainty, that is, data sets over which NCC2-MAR is already quite indeterminate, and to which the uncertainty coming from missing data adds up; this is the case of glass; (b) high number of features variables (40–60), as in the case of kr-kp, optdigits, waveform, splice; in the case of the letter data set, the cause is instead (c) the high number of categories (10 on average) in which feature variables are discretized. Factors (b) and (c) increase the indeterminacy as they increase the number of complete data sets $d \in o$ consistent with the incomplete one. Increasing the number of the complete data sets makes it more likely to obtain lower values of the optimum in (1), eventually yielding a larger number of non-dominated classes. It is interesting to observe that the factors (a)–(c) above are the same that can lead NBC to overfitting. Hence, we can conjecture that classifiers based on precise and imprecise probability, respectively react in different ways to such critical characteristics of the data, the first ones by losing reliability, the second ones by becoming excessively cautious.

Let us focus now on the the size of the indeterminate output, which is the average number of classes in the set-valued classifications returned by NCC2 in a certain setting, and on set-accuracy, which denotes the percentage of times when set-valued classifications contain the actual class (we recall that the indeterminate output size and set-accuracy are measured only on data sets with more than two classes). The size of the indeterminate output should be compared with the (average) number of classes of the data sets under considerations, which is 8.8; hence, NCC2-MAR returns on average less than half the classes, and NCC2-nonMAR slightly more than half the classes. This

	MAR	non-MAR
	NBC	
Accuracy (%)	80.5	80.9
	NCC2-MAR	
Determinacy (%)	91.1	91.1
Set-Acc. (%)	84.0	84.3
Indeterminate Output Size	2.6	2.6
	NCC2-nonMAR	
Determinacy (%)	33.0	49.4
Set-Acc. (%)	97.9	96.6
Indeterminate Output Size	6.0	5.3
NBC accuracy (%) on subsets of instances		
	MAR	non-MAR
NCC2-MAR D	83.2	83.5
NCC2-MAR I	48.4	48.0
NCC2-nonMAR D	92.2	90.4
NCC2-nonMAR I	74.2	71.3
ΔNCC2-nonMAR	75.1	72.7

Table 1: Measured performance, averaged over all the 18 data sets. The average of set-accuracy and indeterminate output size has been computed considering only the data sets with more than two classes; the average number of classes of such data sets is 8.8.

shows that set-valued classifications are informative: they lead us to drop on average half of the classes, as sub-optimal, for the more doubtful instances, thus preventing an over-confident use of the issued judgment. Moreover, indeterminate classifications allow very high set-accuracy under any MP setting; about 85% for NCC2-MAR and about 96% for NCC2-nonMAR. Set-valued classifications appear then as a very effective way to maintain reliability while conveying informative content. This is especially appealing for contexts in which the classification outcome is very sensitive, such as, for instance, the medical area.

Probably the most interesting part of our results concerns the analysis of the accuracy of NBC made separately for the instances classified in a determinate way by NCC2 and for those whose classification is set-valued. Here the intuition is that we expect NBC to perform worse in the latter set of instances, as NCC2 deems that they are harder to classify. Indeed, by averaging over the two different MPs, we obtain that the accuracy of NBC drops of 33 points from *NCC2-MAR D* to *NCC2-MAR I*; of 20 points from *NCC2-nonMAR D* to *NCC2-nonMAR I* and of 19 points from *NCC2-nonMAR D* to Δ*NCC2-nonMAR*.

The overall performance of NBC can hence be regarded as the average of a good accuracy on the instances which are easier to classify, and a much lower accuracy on the instances which are harder, due to the fact that the available information about them is reduced with respect to the former ones. Such a reduction depends in part on the learning set, which in general contains different degrees of information in relationship with different units to classify; and in part on the number and type

of missing feature variables in those units, which contributes to determine the information that the classifier can exploit to classify them.

We regard the alternate good-bad performance of NBC on different units as an important finding, which should be brought to light and properly taken into account. In contrast, such a finding can well remain hidden if we only measure the predictive accuracy, as it is common with traditional classifiers, because it is an average over all the instances in the test set.[11] NCC2, instead, recognizes the more difficult instances and preserves its reliability by issuing indeterminate classifications. (NCC2-MAR does the same, but not with respect to the hardness originated by missing data, both in learning and test set.)

In fact, the following relationships hold on the average indicators, but also on *every* data set-MP pair:

- *NCC2-MAR D* $>$ *NCC2-MAR I*,
- *NCC-nonMAR D* $>$ *NCC2-nonMAR I*,
- Δ*NCC2-nonMAR* $>$ *NCC2-nonMAR I*.

The intuition behind these inequalities is, as before, that on the instances where NCC2 is indeterminate, there is a drop of accuracy of NBC compared to the remaining instances. This happens not only on every data set but also under every experimental setup that we considered. This enforces the idea that NCC2 truly isolates instances that are hard to classify and where, as a consequence, NBC is less reliable.

Despite the inequalities remain the same in the different settings, one should not overlook the fact that the amount of instances isolated by NCC2 may be also very different in the different cases. In particular, looking at Table 1, we see that NCC2-nonMAR may produce even 5–6 times as much indeterminacy as NCC2-MAR. This is not surprising as prior uncertainty tends to vanish with larger samples whereas this does not need to be the case for the uncertainty originated by missing values. We see then that the large majority of indeterminate classifications issued by NCC2-nonMAR are due to missing data rather than prior uncertainty.[12]

It is useful to remark also that the indeterminacy generated by NCC2-nonMAR may sometimes be greater than necessary, or, in other words, that NCC2-nonMAR may show an excess of caution. This makes the drop in NBC accuracy be usually smaller from *NCC2-MAR D* to *NCC2-MAR I*, than from *NCC2-nonMAR D* to Δ*NCC2-nonMAR*. The excess of caution appears because the MP acting on the data is simpler than what NCC2-nonMAR expects. One reason is that for the MP described in Section 4.1 only half the values contained in $\mathcal{A}_j$ are actually possible replacements. This information is not accessible to NCC2-nonMAR, which then considers all the values in $\mathcal{A}_j$ as possible replacement for the missing values of A_j. The other, more important, reason is that we have

11. There are some reasons why computing in addition the standard deviation of the predictive accuracy would not be very helpful either: (i) the standard deviation might be small simply because so is the subset of test-set instances on which the classifier performs badly; (ii) the standard deviation cannot be reliably estimated in the case of small samples; (iii) finally, even if computing the standard deviation might point to the problem, it would not help to isolate the critical instances, nor to know what to do with them, as opposed to what NCC2 makes it possible to do.
12. With respect to the variability of the empirical measures collected, we can observe that *NCC2-MAR I* has usually larger standard deviation than the others (see Tables 6 and 7); this is expected, as NCC2-MAR tends to generate indeterminacy on relatively few instances. Anyway, the difference between *NCC2-MAR D* and *NCC2-MAR I*, and between *NCC2-NonMAR D* and *NCC2-NonMAR I*, is much larger than the standard deviations of the estimate of then indicators (actually, is it well above twice the standard deviation in most cases).

deliberately designed the non-MAR MPs so as to make them act in quite a naive way. We have done so to show that even those MPs can lead to appreciable problems for NBC, as confirmed by the fact that for the large majority of data sets it holds that the NBC accuracy on the ΔNCC2-nonMAR area decreases, moving from the MAR to the non-MAR setting. And, finally, when we consider MPs that are not naive, as the one described in Sect. 4.6, we see that the caution of NCC2-nonMAR is fully justified, NBC being totally unreliable on the area of indeterminacy.

4.4 NBC Probabilities Vs. Set-valued Classifications

We have shown that, thanks to imprecise probabilities, NCC2 delivers set-valued classifications on hard-to-classify instances, over which the accuracy of NBC clearly drops. In the following, we further compare the predictions of NBC and NCC2 by taking into consideration also the posterior probabilities computed by NBC for the classes. To make things clearer we carry out the analysis in two steps.

The first step, described in the next two sections, focuses on a simplified setup that allows us to make a very detailed analysis: a single data set containing two classes, and subject only to a MAP MP. The focus in this case is on prior ignorance, as the only possible source of indeterminacy for NCC2; we can therefore understand how strong assumptions about the prior can affect the ability of NBC to compute posterior probabilities. Thanks to the restriction to the binary case (i.e., two classes), we can then also clearly see that there is a difference between the cases that are deemed doubtful by NBC (i.e., with probability close to 50%) and those that are deemed so by NCC2 (i.e., that lead it to indeterminacy).

The second step extends the analysis to all the data sets and to ignorance originated by non-MAR missing data. In this case there is necessarily less detail but we have the advantage of having the full comparison between NBC probabilities and NCC2 indeterminacy, and where we observe that the situation basically resembles the one observed in the first step. This is done in Section 4.4.3.

4.4.1 AN ILLUSTRATIVE EXAMPLE: THE SPECT DATA SET

We focus on the spect data set, which is made of 2 classes and 267 instances.

For the following analysis, we split the data set into training set of 67 instances and a test set of 200 instances. We choose a 25-75% split in this example because we want to clearly see the effect of the prior, which is well known to decrease with increasing sizes of the learning set.

We run the experimental framework of Section 4.3 with the MAR MP; that is, we generate 100 pairs of training and test sets (made artificially incomplete by the MP) from the original training/test split. The findings presented in the following are obtained by analyzing the predictions issued over the 100 test sets.

In particular, we consider four pieces of information for each classified instance: the actual class, the class returned by NBC and its associated probability, and whether or not the instance has been classified by NCC2 in a determinate way. The instances are then partitioned into subsets, according to the probability estimated by NBC for the returned class, that is, instances for which NBC estimates a probability in the range 50–55%, 55–60%, and so on (i.e., we use a step of 5% in probability to define the subsets). On each subset of instances, we measure: (a) the determinacy of NCC2-MAR; (b) the accuracy achieved by NBC on the instances classified determinately and (c) indeterminately by NCC2-MAR.

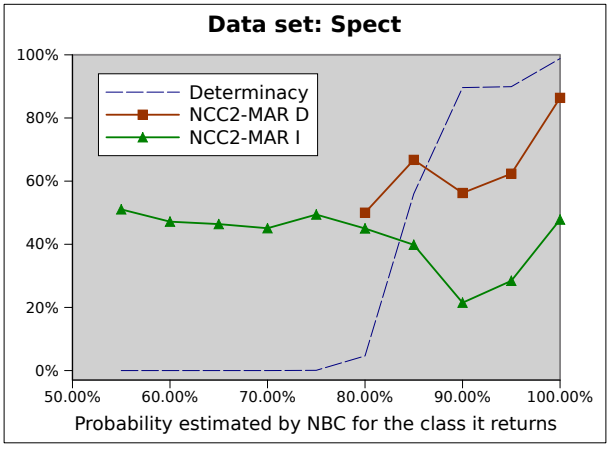


Figure 3: Relationship between the posterior probabilities computed by NBC and the output of NCC2-MAR on the spect data set.

The results are reported in Figure 3. The dashed line represents the percentage of instances in the subset (related to a certain range of probabilities) under consideration that is classified determinately by NCC2. We see then that there is a positive association between higher posterior probabilities computed by NBC and higher determinacy. This is a natural effect but it needs some words to be explained, and we postpone this task to the next section.

An interesting point is that the output of NCC2 is indeterminate for all the instances as long as the probability estimated by NBC for the returned class is lower than 75%. In other words, here NCC2 deems that there is very little information about those instances in the learning set and suspends the judgment. Remarkably, such caution is justified as on those instances NBC is just guessing (uniformly) at random, at is follows from the line for NCC2-MAR I. The point here is that NBC believes to be quite confident on a number of instances (assigning probabilities that go as up as 75%) but in practice it is not. Stated differently, it believes it knows but it does not. NCC2 simply knows not to know from the very beginning and this enables it to avoid losing credibility.

Moving on to greater probabilities, we see that the determinacy of NCC2 rises substantially when the probability estimated by NBC exceeds 80%; in this region NCC2 returns a mix of determinate and indeterminate classifications, and the drop of accuracy between *NCC2-MAR D* and *NCC2-MAR I* is large at any level of posterior probability. In fact, NCC2 returns indeterminate classification also on a non-negligible number of instances classified very confidently (for instance, with probability higher than 85%) by NBC, and over which the accuracy of NBC is bad indeed.

The last discussion points to an important question, which is useful to stress: that is, to the fact that NCC2 does not suspend the judgment only on instances that are deemed doubtful by NBC, that is, those whose probability is for instance less than 55%. Moreover, while on the spect data set NCC2 is fully indeterminate on the instances on which NBC is doubtful, this is not always the case. For instance, the same analysis performed on the spambase data set (2 classes, 1300 training instances) shows that about 40% of instances in the range 50–55% of probability are classified

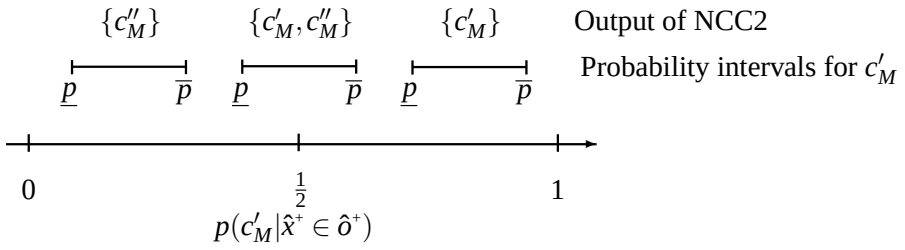


Figure 4: A graphical view of the test of dominance in the binary case. The output of the classifier is indeterminate if and only if the posterior probability interval for c'_M contains $\frac{1}{2}$.

determinately by NCC2 (while the behavior on the remaining ranges of probabilities followed a pattern similar to the one shown in Fig. 3).

4.4.2 ASSOCIATION BETWEEN NBC PROBABILITIES OF NCC2 DETERMINACY

In the following, we explain, with reference to a generic data set with two classes, why there is a positive association between increasing posterior probabilities computed by NBC and increasing determinacy of NCC2.

Re-consider the notation that we have introduced for the test of dominance in (1). Define the posterior *lower* and *upper probabilities* of class c'_M respectively as

$$\underline{p}(c'_M|\hat{x}^+ \in \hat{o}^+) := \inf_{p(\theta)\in\mathcal{P}(\theta)} p(c'_M|\hat{x}^+ \in \hat{o}^+)$$

and

$$\overline{p}(c'_M|\hat{x}^+ \in \hat{o}^+) := \sup_{p(\theta)\in\mathcal{P}(\theta)} p(c'_M|\hat{x}^+ \in \hat{o}^+).$$

Compared to (1), here we have removed the parts related to the non-MAR missingness process as in this moment we prefer to keep things simple by focusing on the MAR case, as in the previous section.

With the new definitions, the test of dominance (1) can be easily re-written in the case of a data set with two classes as a simple test that checks the position of the interval $[\underline{p}(c'_M|\hat{x}^+ \in \hat{o}^+), \overline{p}(c'_M|\hat{x}^+ \in \hat{o}^+)]$ relative to the probability value $\frac{1}{2}$. This is shown graphically in Fig. 4. Such a figure shows three possible positions of the posterior intervals, which are represented as segments. When the interval contains the value $\frac{1}{2}$, as in the middle case, then the output of the classifier is indeterminate: both classes are returned. Here we have complete indeterminacy because it is not possible to know if one of the two classes has posterior probability larger than the other (note that in this case the interval for c''_M contains $\frac{1}{2}$ as well). In the remaining two cases, the value $\frac{1}{2}$ is either greater or less than all the points in the interval and this allows the classifier to know that one of the two classes is dominated, thus returning a determinate classification, as shown above the intervals. Now, assume temporarily the width of the NCC2 interval to be constant across all the instances. By construction, the posterior interval computed by NCC2 is typically "close" to the posterior probability given by NBC.[13] Hence, the NBC probability and the NCC2 interval tend to move together; if the NBC

13. In our experiments the posterior probability is actually very often in the interval, but we cannot say this is always the case because the Laplace prior used for NBC is not contained in the prior credal set of NCC2.

probability grows, then the NCC2 interval reaches a point that makes it exclude the value $\frac{1}{2}$, leading to a determinate classification. This explains the basic mechanism behind the association between NBC probabilities and NCC2 determinacy.

Let us now move on to the realistic scenario that involves dropping the simplifying assumption about the width of the NCC2 interval to be constant.[14] In fact, it practice it happens that the length of the NCC2 interval varies from unit to unit, as it depends on the amount of information that the learning set contains about the *specific unit under consideration*: the less the information the wider the interval, and vice versa. The fact that the learning set contains different amounts of information for different units to classify should not be surprising: a unit to classify can be put in correspondence with the subset of the learning set characterized by the values taken by the feature variables for the unit under consideration; and these subsets have (also very) different sizes (remember that we are dealing only with the MAR case; in the non-MAR case, we should consider also the effect of missing data and the interpretation would be slightly more complicated, but still of a similar kind).

Therefore, it may well happen that for a certain unit to classify to which NBC assigns a high posterior probability, the width of the NCC2 interval dilates so much that $\frac{1}{2}$ gets included in the interval, leading to an indeterminate classification. In this case, NCC2 deems that there is little information in the learning set about the unit under consideration, and expresses this by a large width of the interval. NBC does not have such expressive means and, moreover, may generate such a high probability mostly because of its prior density, drawing then conclusions that are not supported by the data.

In any case, it is clear that increasing the NBC probability makes it less likely that the NCC2 interval is as large as to be able to contain the value $\frac{1}{2}$. This is the reason why we observe the, very natural at this point, association between increasing NBC probabilities and increasing NCC2 determinacy.

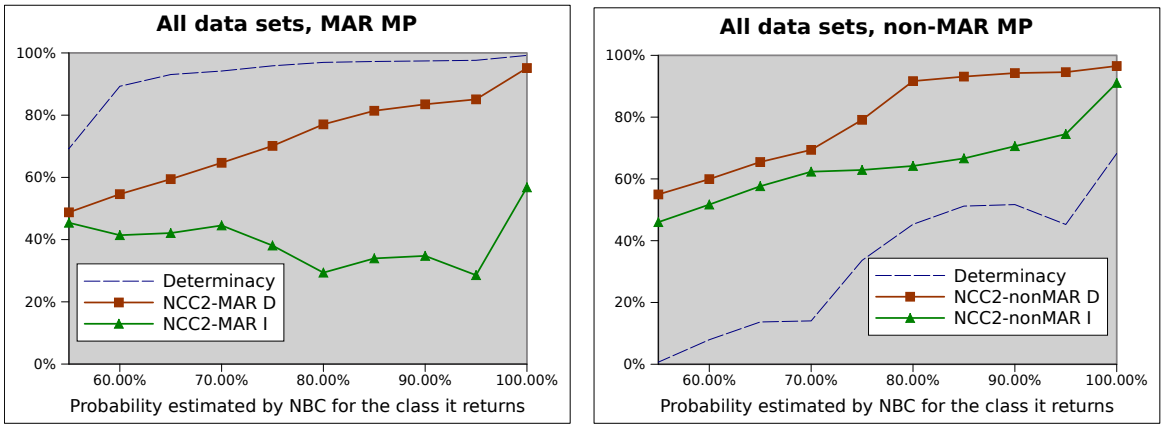
4.4.3 RESULTS ON ALL THE DATA SETS

In this section we consider all the usual 18 data sets with the same training/test splits of Section 4, that is, 50%-50%. We consider two settings: the first compares NBC and NCC2-MAR when the MP is MAR, and the second compares NBC and NCC2-nonMAR when the MP is non-MAR. We analyze jointly the predictions issued on all the data sets, respectively under the first (Figure 5a) and second setting (Figure 5b).

One point to consider is that in data sets with two classes, one can spot the instances doubtful for NBC looking at those with probability around 50% for the returned class. However, for data sets with more than two classes, the instances that are doubtful for NBC are not as easy to recognize as before; for instance, with four classes both the posterior mass functions [40%, 40.5%, 15%, 4.5%] and [24.5%, 25.5%, 25%, 25%], lead to doubtful classifications for NBC. For this reason, we only focus now on the case when NBC is confident, that is, on the instances with probability for the returned class greater than or equal to 55%.

Looking at the figures we see that the determinacy is much higher in the MAR setup than in the non-MAR; in fact, we have already seen that the non-MAR case leads in general to a much larger amount of instances that are isolated as difficult ones. Nevertheless, both Figures 5a and 5b show a clear drop of accuracy of NBC, for the same level of posterior probability of the predicted class, from

14. If it were really so, the behavior of NCC2 would be quite trivial, and could be reproduced by that of an NBC that uses a threshold larger that $\frac{1}{2}$ to decide when a class should be said to dominate the other.



(a) NBC, NCC2-MAR and MAR MP

(b) NBC, NCC2-nonMAR and non-MAR MP

Figure 5: Relationship between the posterior probabilities computed by NBC and the output of NCC2.

the instances classified in a determinate way by NCC2 to the others. The drop is especially striking on the instances classified confidently by NBC, when the computed probability is for instance larger than 70%. This is more evident in Fig. 5a; on the other hand, the drop observed in Fig. 5b applies to a much larger set of instances. In fact, NCC2-nonMAR suspends the judgment frequently also on the instances classified by NBC with probability greater than 80%; the reason is that a replacement for missing data especially unfavorable for the class predicted by NBC, can well change the outcome of the classification with respect to the output of NBC, which instead marginalizes out the missing feature variable.

The instances for which NBC returns a probability higher than 90% are of particular interest, also because they constitute 62% of the total instances. On this area, NCC2-MAR returns 1.5% of indeterminate classifications on which it achieves the following remarkable performance: *NCC2-MAR D*: 94%; *NCC2-MAR I*: 54%. The performance of NCC2-nonMAR in this area is as follows: determinacy 66%, *NCC2-NonMAR D*: 96%; *NCC2-NonMAR I*: 89%. The drop is in this case significant, yet smaller than under MAR: in fact, as already pointed out NCC2-nonMAR is more conservative than necessary under this setting.

Summing up, hence, uncertainty related to the choice of the prior manifest itself more evidently, but not only, on the instances in which NBC is less confident; yet, only part of such doubtful instances lead to indeterminate classifications. On the other hand, missing data treated as non-MAR can lead to many indeterminate classifications even if the probability computed by NBC for the returned class is high. This is clear by comparing the dashed lines in Figures 5a and 5b.

4.5 Results with Increasing Missingness

Next, we analyze how NCC2-nonMAR deals with very high levels of missingness. We adopt the non-MAR MP, with missingness rate at 15% per feature variable; given how the MP works, it will produce about 7.5% missing data on each feature. We focus on two data sets: segment-challenge

		NBC	NCC2-nonMAR				Subsets	
	MP rate (%)	Accuracy (%)	Determinacy (%)	Accuracy (%)	Set-Accuracy (%)	Output Size (%)	NCC2-nonMAR D	NCC2-nonMAR I
segment	5	91.6	54.8	97.7	99.4	5.5/7	97.7	84.1
segment	15	91.4	18.3	99.4	99.9	6/7	99.4	89.6
segment-pruned	5	93.8	77.8	98.0	99.7	4.6/7	98.0	79.1
segment-pruned	15	93.0	48.8	99.1	99.5	5.1/7	99.1	87.7

Table 2: *Segment-challenge* data set: impact of feature selection on the performance of NBC and NCC2-nonMAR. Standard deviation of the indicators is always smaller than 2%, that is, all indicators $\pm$ (at maximum) 2 percentage points.

(classes: 7; feature variables: 19; average number of bins per discretized feature variable: 5.6)[15] and waveform (classes: 3; feature variables: 40; average number of bins per feature variable: 8.4).[16]

Let us stress that in our intentions the results we present constitute a kind of worst-case analysis in terms of achieved determinacy, since (a) the amount of missing data generated by the MP is far higher than those usually found in real data sets; (b) NCC2-nonMAR treats all feature variables as non-MAR, both in training and testing (while usually, the investigator is able to isolate some MAR feature variables); (c) the considered data sets contain a high number of feature variables, classes and bins per feature variable, and these characteristics work together towards rising the indeterminacy of NCC2-nonMAR. Moreover, as already discussed, given how the non-MAR MP works, NCC2-nonMAR is more conservative than necessary under this setting.

Tables 2 and 3 report the results, displaying also the outcomes previously obtained with missingness rate 5%. In fact, while the performance of NBC is substantially insensitive to the amount of missing data, the determinacy of NCC2-nonMAR clearly decreases; with rate of missingness per feature variable equal to 15%, the determinacy of NCC2-nonMAR drops to about 1% on waveform and 18% on segment. As a side-effect, we see also that the indicator NCC2 I increases with the level of missingness; this is due to the fact that the number of instances on which NCC2 I is computed approaches the overall test set, so that NCC2 I approximates the average accuracy of NBC. Remarkably, however, even in such a difficult setting, there is a clear drop of accuracy (some 15 points), on both data sets, between NCC2 D and NCC2 I.

Is there a way to reduce the indeterminacy of NCC2-nonMAR? A first way to improve the situation is feature selection. On the one hand, feature selection is necessary if one wants to get maximum performance from NBC and NCC2, since redundant feature variables, which violate the naive assumption, might skew the learning process of both classifiers; on the other hand, reducing the set of feature variables reduces the amount of missing data, which is beneficial for the deter-

15. Two feature variables, out of the 19, are discretized into a single bin; they are not considered when computing the average number of bins.

16. Twenty feature variables, out of the 40, are discretized into a single bin; they are not considered when computing the average number of bins.

		NBC	NCC2-nonMAR				Subsets	
	MP rate (%)	Accuracy (%)	Determinacy (%)	Accuracy (%)	Set-Accuracy (%)	Output Size (%)	NCC2-nonMAR D	NCC2-nonMAR I
waveform	5	81.3	54.3	89.5	99.9	2.1/3	89.5	71.7
waveform	15	81.1	1.2	100	100	2.7/3	100	81.1
waveform-pruned	5	82.3	65.9	88.1	99.9	2.1/3	88.2	70.9
waveform-pruned	15	81.7	10.2	97.0	100	2.5/3	96.9	79.9

Table 3: *Waveform* data set: impact of feature selection on the performance of NBC and NCC2-nonMAR. Standard deviation of the indicators is always smaller than 2%, that is, all indicators $\pm$ (at maximum) 2 percentage points.

minacy of NCC2-nonMAR. We performed feature selection by cross-checking the suggestions of different methods implemented in WEKA (Witten and Frank, 2005); eventually, we selected 6 feature variables for segment-challenge and 14 for waveform.

In Table 2 and 3, the data sets after feature selection are referred to as *pruned*. The effectiveness of the feature selection we performed is demonstrated by a slight yet sensible improvement of NBC accuracy; as for NCC2-nonMAR, it achieves a satisfactory 50% of determinate classifications on segment-challenge, and a less satisfactory 10% on waveform. Still, it is remarkable that such a 10% of instances is naturally classified in a determinate way under such weak assumptions and with such an amount of missing data; this should make us quite confident about the reliability of the classification, and indeed the measured accuracy is 97%. The drop between NCC2 D and NCC2 I after feature selection does not show major changes compared to before feature selection.

So far the analysis in the most conservative possible conditions. Let us recall at this point that a strength of NCC2 is the possibility of declaring that some feature variables are MAR (possibly changing this from learning to test set). This is likely to be often possible to do in practice if we only consider that currently most classifiers take for granted that all the feature variables should be MAR. We considered then a new setup for waveform-pruned obtained by declaring that only half the feature variables are non-MAR both in training and testing; and the determinacy of NCC2 raised up to 58.5%.

4.6 A More Sophisticated MP

So far, we have considered a very simple MP, which in particular works independently on each feature variable. It is interesting at this point to consider a more complex MP, working for instance on the joint values of the feature variables. We show that this can have much bigger effects: in fact, it can lead not only to severe misclassifications, but also to erroneous empirical evaluations of the accuracy of classifiers which assume MAR.

We consider now the vote data set, also taken from the UCI repository. The data set contains information about the United States Congressional Voting; the classification task is to guess whether

a certain congressman is republican or democrat, by looking at his/her votes about some critical laws. The data set has 435 instances, 16 features variables and 2 classes; by feature selection, we reduce to 3 the number of features variables. The pruned data set is complete, that is, it does not contain any missing data. In the following, we denote by the symbol "*" a missing value generated by the non-MAR MP. Let us name as "type A" the instances with values (*n, y, n, class1*) and as "type B" the instances with values (*y, n, n, class0*). The data set contains about 26% type A and 26% type B instances.

The malicious MP that we consider turns the type A instances of the training set into (*, *, n, class1*), and the type B instances of the test set into (*, *, n, class0*); it hence affects about 26% of the training set and 26% of the test set.

We measured the following on the test set:

- NBC accuracy 71.4%;
- NCC2-MAR: determinacy 100%; single accuracy 71.4%;
- NCC2-nonMAR: determinacy 55.3%, single accuracy 99.2%.

Remarkably, in this case NCC2-MAR is always determinate, and therefore it behaves identically to NBC. With reference to the previous results on the same data set, obtained with a simpler MP, there is a major drop of accuracy (some 25 points) for NBC (and consequently for NCC2-MAR, too); on the other hand, NCC2-MAR becomes more indeterminate, but it preserves a high single-accuracy.

Going more deeply in the analysis, the accuracy of NBC and NCC2-MAR is 99.2% on the instances NCC2-nonMAR D, and 37.1 only (lower than random guessing!) on the instances NCC2-nonMAR I. Hence, this MP heavily deteriorates the reliability of the classifiers which assume MAR. NCC2-nonMAR, being based on CIR, behaves instead robustly even against such a malicious MP.

However, there is a further point of interest, concerning the failure of the empirical evaluations of classifiers that are always determinate, when data are made missing by a non-MAR MP. Let us assume that the only available data are the instances of the training set. Evaluating by cross-validation the classifiers on the instances of the training set, we measure:

- NBC accuracy 88.2%;
- NCC2-MAR: determinacy 89.9%; single accuracy 93.9%;
- NCC2-nonMAR: determinacy 49.5%, single accuracy 100%.

The evaluation of the accuracy of NBC and NCC2-MAR is hence heavily biased when compared against their actual performance on the test set. This phenomenon is discussed at some length in Section 6 of Zaffalon (2005b). On the other hand, NCC2-nonMAR is reliably evaluated.

Of course, this kind of "extreme" example heavily relies on the fact that the MP is not identically distributed. Yet, note that if one is ignorant about the MP, such a behavior should be considered as a possibility, which is just what NCC2 does. On the other hand, imposing the assumption that the MP is identically distributed should be done on a case-by-case basis, as doing it in general is questionable (unlike imposing the data generation process to be so, given that MPs are usually processes of a very different kind). Alternative assumptions, between the two extremes of assuming ignorance and assuming that the MP is identically distributed, are not so easy to envisage if we also require that they are widely applicable as one would like them to be in the field of data mining.

Algorithm	Determinacy (%)	Accuracy (%)	Output Size	Set-accuracy (%)
NBC	-	62.4 ± 5.1	-	-
NCC2-MAR	91.8 ± 3.3	64.2 ± 5.5	2.2 ± 0.2	83.9 ± 15.4
NCC2-nonMAR	65.1 ± 5.5	60.1 ± 7.7	3.3 ± 0.3	89.8 ± 5.7

Table 4: Results on the eucalyptus data set. Accuracy refers to determinate classifications only, while output size to indeterminate classifications only.

	Values of feature variable "vigour"						
Output class	*1*	*2*	*3*	*4*	*5*	*6*	*7*
c_1 ("*none*")	24	10	14	1	6	1	0
c_2 ("*low*")	0	3	24	17	8	2	0
c_3 ("*average*")	0	0	2	17	31	14	1
c_4 ("*good*")	0	0	0	6	28	57	16
c_5 ("*best*")	0	0	0	0	8	21	23

Table 5: Counts $n(a_j, c_k)$ for feature variable "vigour" in the eucalyptus data set.

In summary, we conclude that the conservative approach provided by NCC2 can be considered as a valuable avenue to reliably cope with situations where we miss substantial information about (part of) the MP. Such an approach might become even more effective by exploiting coarsened observations, as a way to insert knowledge about the MP. This is discussed in the next section.

4.7 The Eucalyptus Data Set: A Difficult One For NCC2-nonMAR

The *eucalyptus* data set is available from the Weka (Witten and Frank, 2005) repository[17] of public data sets. It contains 736 instances; there are 19 feature variables and 5 classes. After feature selection, there are 8 feature variables left; about 3% of the values in the data set are missing. NBC, NCC2-MAR and NCC2-nonMAR have been validated via 10 runs of 10 folds cross-validation; feature variables have been discretized via MDL-based discretization (Fayyad and Irani, 1993). Results are reported in Table 4.

Strikingly, NCC2-MAR has higher single-accuracy than NCC2-nonMAR. To investigate this issue, we split in a stratified way (i.e., keeping the proportion of the classes as close as possible between training and test set) the data set into a training and a test set; this enables us to detect a set of 35 critical instances, which are classified accurately and in a determinate way by NCC2-MAR, and in a totally indeterminate way by NCC-nonMAR, which returns five classes out of five.

17. The URL is `http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html`.

A key point is that all these instances lack the value of feature variable vigour. The frequencies $n(a_j, c_k)$, reported in Table 5, show that, for any class c_k, there is at least a value $\tilde{a}_j$ such that $n(\tilde{a}_j, c_k)$=0. This is due to different reasons, such as the high number of classes in which the feature variable is discretized; the skew of the distribution of the counts $n(a_j, c_k)$, for any class c_k, towards certain values of the feature variable; the high number of output classes (i.e., 5), and also the relatively small size of the data set.

In fact, if the value of vigour is missing and the MP is supposed to be non-MAR, for any pair of classes (c_i, c_j) there exists a replacement $\tilde{a}_j$ for the missing value such that $n(\tilde{a}_j, c_i) = 0$, and hence $p(c'_M|d^-, \hat{x}^+ \in \hat{o}^+)/p(c''_M|d^-, \hat{x}^+ \in \hat{o}^+) = 0$, thus preventing the existence of any credal-dominance relationship between any two classes. In this condition, hence, NCC2-nonMAR has to output all the classes.

On the other hand, the conditional counts of the remaining feature variables, strongly skewed towards class c_1, allow NCC2-MAR (and NBC as well) for predicting all these instances accurately and in a determinate way. This phenomenon is the reason why NCC2-MAR is both more accurate and determinate than NCC2-nonMAR on the eucalyptus data set.

4.7.1 SOLUTION VIA COARSENING (OR SET-BASED OBSERVATIONS)

Yet, NCC2-nonMAR is processing correctly the eucalyptus instances: if the MP affecting the vigour feature variable is non-MAR, by definition all the possible replacements for the missing values should be taken into consideration. However, NCC2-nonMAR is unnecessarily cautious, as demonstrated by both NCC2-MAR and NBC, which are able to classify the critical instances both accurately and in a determinate way. It appears hence that some further knowledge should be put into the classifier, to mitigate its unnecessary caution; yet, we try to avoid assuming MAR. A step forward in this direction can be accomplished by introducing *set-based* (or *coarsened*) observations.

Up to now, we have assumed that manifest values are either equal to the latent ones, or set equal to the symbol of missing data. In the former case, the value is known exactly; in the latter, it is known in the most indeterminate way, that is, we only know that it belongs to the set of possible values of the feature variable. Coarsened observations represent an intermediate situation of knowledge, in which the observation, for instance referring to feature variable A_j, is known to belong to a set $\mathcal{A}'_j \subseteq \mathcal{A}_j$. The size and composition of $\mathcal{A}'_j$ can vary between different set-based observations of the same feature variable, thus allowing for a high degree of flexibility. Actually, the possible values of $\mathcal{A}'_j$ are all the possible non-empty subsets of $\mathcal{A}_j$.

If A_j is assumed to be affected by a non-MAR MP, set-based observations restrict the set of possible replacements from $\mathcal{A}_j$ to $\mathcal{A}'_j$ (we recall that $\mathcal{A}'_j$ will generally be different for each set-based observation). Therefore, set-based observations provide, without implying any strong assumption about the MP, a higher degree of knowledge than missing data, and they tend to generate less indeterminacy than missing data.

Remarkably, real MPs often produce set-based observations rather than missing data. For instance, let us consider a doctor that recommends a patient to go through exam B only if exam A is positive. In fact, the domain knowledge of the doctor makes it possible to conclude that if A is negative, the outcome of B will range within a restricted set of values, not influential for the diagnosis. On the contrary, if A is positive, the value of B cannot be predicted. However, when B is not performed because A is negative, we do have a set-based observation rather than a missing value.

In many cases, it is possible to provide coarsening sets rather than missing values at the time data are collected, by excluding unsuitable values from the set of the possible replacements.

Let us stress that using set-based observations and assuming MAR are two fundamentally different ways of inserting knowledge into the classifier. In fact, set-based observations incorporate specific domain knowledge (obtained at data collection or data validation time) into the data set and eventually into the classifier. On the other hand, MAR is an assumption typically done by the modeler, whose tenability is only rarely checked with domain experts.

What happens using set-based observations on the eucalyptus data set? On the 35 critical instances we tried to substitute the missing values of vigour with set-based observations, using a coarsening set $\mathcal{A}_j^{'} = \{1,2,3,4,5\}$, which contains 5 values out of the 7 possible. Using such set-based observations, NCC-nonMAR classifies 32/35 critical instances correctly and in a determinate way; on the remaining 3 instances, it outputs 4 classes, including the true one. In this way, the usual inequalities (*NCC-MAR D* $>$ *NCC-MAR I*, *NCC-nonMAR D* $>$ *NCC-nonMAR I*, Δ*NCC2-nonMAR* $>$ *NCC-nonMAR I*) hold also on the eucalyptus data set.

In our example, set-based observations constitute an effective way to insert knowledge about the MP into the classifier without assuming MAR; they appear hence as a promising approach for credal classifiers. It is useful to observe that coarsened observations have received considerable attention in the traditional literature of statistical inference from incomplete observations. A work of paramount importance on this topic is the popular coarse-data model of Heitjan and Rubin (1991), which generalizes the missing-data model of Rubin (1976). This appears to enforce the potential of coarsening for classification in general, and to make it worth of further investigation. From a practical viewpoint, the use of set-based observations would require some changes in the data archiving formats, which are usually designed to store precise observations only.

5. Conclusions

There are usually different amounts of information in the data that a classifier can exploit to classify different units. This depends in part of the fact that a learning set can be more informative about some units than some others, and in part on the type and amount of missing values in the units to classify. As a consequence, a classifier's predictions may be more uncertain on some units than on some others. But if we impose that the classifier is inferred using a single prior density and that it considers all the missing data as ignorable, we lose much of the capability to distinguish that some units are harder to classify than others: we cannot see anymore that some of the strong conclusions that we obtain are determined by our strong assumptions rather than the supposedly strong information in the data, and are hence actually fragile. Nor can help much in this respect the experimental evaluations of the classifier as they are, somewhat necessarily, designed to yield its average performance. And yet, we do care about distinguishing the hardness of different units in many real-world applications: in the case of medical diagnosis, for instance, we do not want a classifier to be good only *on average*; we want it to make reliable predictions on every single patient.

In this paper we have tried to pursue such a goal just by weakening the assumptions that are traditionally made by classifiers. We have modified the naive Bayes classifier so as to model prior ignorance in an objective-minded way by using a set of prior densities, and by giving the opportunity to treat some of the missing values as originated by a missingness process that we do not know (and the others by a MAR one). The resulting model, called naive credal classifier 2, departs from the more traditional classifiers in a number of ways, the more substantial one being the fact that NCC2

makes set-valued classifications in general: it issues a determinate classification (i.e., a singleton) only when it deems that there it has enough information to do so.

Extensive empirical evaluations have shown that NCC2 has high accuracy when it issues determinate classifications, and that when it is more cautious, it is very often justified: NBC is clearly shown to considerably decrease its classification accuracy on the instances classified in a set-valued way by NCC2, as well as its ability to compute predictive posterior probabilities. This indicates that set-valued classifications do indeed isolate the instances of the test set that are hard to classify. And they actually do more than just isolating them: they are in fact still informative, as unlikely classes are dropped anyway, and invite the domain expert (for instance, a doctor that has to issue the ultimate diagnosis) to avoid over-confident statements. We have also pointed out that in some extreme cases of missingness processes, the empirical evaluations made for naive Bayes can be completely biased, so that not even its average predictive accuracy can be evaluated reliably. In these cases, NCC2 was instead still reliably evaluated.

Dealing effectively with the hard instances is not an easy task in general, and we have shown that NBC fails on this a number of times by yielding unreliable classifications due to its inherent optimism. The way NCC2 sometimes fails on this is different, as it leads to an excess of caution; this is less critical although still not desirable. In these cases, the classifier is too pessimistic and the information used to build it up should be strengthened in order to obtain stronger conclusions. The more obvious way to do so with NCC2, and that—provided that one stays within the realm of tenable assumptions—is also what we recommend, is to minimize the number of missing values that are declared to be subject to an unknown missingness process. A less obvious but still very important way to do so is to simplify the model as much as possible, for example by doing feature selection. In fact we have conjectured that the same factors that can lead NBC to overfitting (e.g., high number of features or feature variables) can well result in excessive caution of NCC2. When also this is not enough, another approach may prove to be very helpful: using coarsened rather than missing values in data sets. Coarsened, or set-based, observations carry more information than missing ones in general because they naturally exclude some values as possible replacement for the missing information. Such extra information might be often available in real applications, and it seems to have the potential to lead to strong enough conclusions in a number of cases. It is also a kind of information that is typically provided by a domain expert rather than by the modeler (as opposed to MAR, for example) and as such it is a good candidate to be tenable.

More generally speaking, we regard the problem of providing a classifier with knowledge about the missingness process, which usually cannot be obtained from data, as a serious and an important topic of research. This seems to have much to do with identifying general and tenable assumptions about missingness processes of which classifiers could take advantage.

NCC2 can be regarded as the outcome of having made a first step in this direction. Doing such a step has led to a classifier that offers us a range of new opportunities to deal robustly with predictions in case of small or incomplete data sets: in particular, it allows us to check whether in very weak states of prior information we can draw strong enough conclusions for our aims, thus avoiding us to worry about doing mistaken assumptions; it lets us work incrementally towards stronger assumptions and conclusions; it enables us to uncover situations that otherwise could not be, not even experimentally. And it makes all of this possible by means of fast and exact computations.

Acknowledgments

We are grateful to all the reviewers for their constructive criticism, which has led us to expand and deepen the empirical analysis eventually making the paper gain insight, as well as to improve the paper at the level of presentation. Marco Zaffalon is also grateful to Peter Walley because of the joint work that led in 2001 to an inferential method for the naive credal classifier that was the starting point of the extension presented here. 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.

Appendix A. A Polynomial-time Procedure for Incompleteness in the Instance to Classify

Although the procedure sketched in Section 3.4 solves Problem (8), a substantially more efficient procedure can be designed to solve the same problem. However, to avoid adding further complexity to the presentation of NCC2 carried out in the main body of the paper, we present it here, in Figure 6. The correctness of the procedure is stated by the following theorem, whose proof is in Appendix B.

Theorem 7 *The optimum of Problem* (8) *is obtained by the procedure in Figure 6.*

1. Build the set $\mathcal{U} := \{x_e = \frac{\overline{n}(a_{Me'},c''_M)\underline{n}(a_{Me''},c'_M)-\underline{n}(a_{Me'},c'_M)\overline{n}(a_{Me''},c''_M)}{s[\underline{n}(a_{Me'},c'_M)-\underline{n}(a_{Me''},c'_M)]} : e = 1,\dots,k', a_{Me'}, a_{Me''} \in \mathcal{A}_e, \underline{n}(a_{Me'},c'_M) \neq \underline{n}(a_{Me''},c'_M), x_e \in (0,1)\}$, where the lower and upper counts are defined as in Theorem 6.
2. Use the points in $\mathcal{U}$ to define a partition $\mathcal{J}$ of $(0,1)$.
3. For each interval $\mathcal{I} \in \mathcal{J}$, determine an associated tuple $(a^{\mathcal{I}}_{M1},\dots,a^{\mathcal{I}}_{Mk'}) \in \mathcal{A}_1 \times \cdots \times \mathcal{A}_{k'}$ by selecting, for each $e = 1,\dots,k'$ an element among those yielded by $\operatorname{argmin}_{a^{\mathcal{I}}_{Me} \in \mathcal{A}_e} \frac{\underline{n}(a^{\mathcal{I}}_{Me},c'_M)}{\overline{n}(a^{\mathcal{I}}_{Me},c''_M)+x^{\mathcal{I}}}$, where $x^{\mathcal{I}}$ denotes the middle point of the interval $\mathcal{I}$.
4. For each of the intervals in $\mathcal{J}$, minimize the function defined by the tuple associated with the interval, by applying of Theorem 6 with suitable changes to adapt it to $\mathcal{J}$.
5. Take the minimum of the values provided by the previous step.

Figure 6: The solution procedure for Problem (8).

The intuitive idea behind the procedure is presented in the following. Let us assume that among the k attribute variables there are k' ($1 \leq k' \leq k$) that are not observed in the instance to classify. Assume, without loss of generality, that they are indexed from 1 to k'. The objective function of the problem can be re-written as (let $x := st(c''_M)$):

$$\inf_{0<x<s}\{[\frac{n(c''_M)+x}{n(c'_M)+s-x)}]^{k-1}\prod_{e=1}^{k'}[\min_{a_{Me}\in\mathcal{A}_e}\frac{\underline{n}(a_{Me},c'_M)}{\overline{n}(a_{Me},c''_M)+x}]\cdot$$
$$\cdot\prod_{j=k'+1}^{k}\frac{\underline{n}(a_{Mj},c'_M)}{\overline{n}(a_{Mj},c''_M)+x}\prod_{l=1}^{r'}[\frac{n_l(c''_M)+x}{n_l(c'_M)+s-x}\cdot\frac{n(\hat{a}_{Ml},c'_M)}{n(\hat{a}_{Ml},c''_M)+x}]\}, \quad (9)$$

where the functions $\underline{n}(a_{Me}, c'_M)/[\overline{n}(a_{Me}, c''_M) + x]$ can be minimized, given x, separately over each a_{Me}, as shown by the minima over $a_{M1}, \cdots, a_{Mk'}$.
For variable A_e, the functions of this type obtained for each different value $a_{Me} \in \mathcal{A}_e$ are compared over the interval $[0, s]$; when the value a_{Me}, which minimizes them, changes, there is a so-called *partition point*. Now, let us consider the set $\mathcal{U}$, which collects the partition points of all the variables $A_1, \ldots, A_{k'}$: the tuple $\{a_1, \cdots a_{k'}\}$ which minimizes the first factor of Problem (9) changes over the interval $[0, s]$ at every point belonging to $\mathcal{U}$. Since the function is convex on each sub-interval, we can solve a minimization problem on each sub-interval; the global minimum is finally found by comparing the local minima obtained over each sub-interval.

Let us analyze briefly the computational complexity of the procedure. The procedure is based on the determination of the partition $\mathcal{J}$, whose size is at most $1 + \sum_{e=1}^{k'} (|\mathcal{A}_e|^2 - |\mathcal{A}_e|)/2$, which is also the number of minimizations in the worst case. Each minimization is done according to Theorem 6. This is based on the computation of the logarithmic derivatives of the function of interest. Each evaluation of the logarithmic derivative is a task linear in $k + r'$, where we recall that r' is the number of non-missing MAR variables in the instance to classify. If we regard the complexity of Newton-Raphson's method as a constant, which is reasonable as it is an extremely fast algorithm, we obtain that the procedure works in time $O([1 + \sum_{e=1}^{k'} (|\mathcal{A}_e|^2 - |\mathcal{A}_e|)/2](k + r'))$. To obtain a simpler expression, we can focus on an upper bound: $O(\overline{\mathcal{A}}^2 (k + r')^2)$, where $\overline{\mathcal{A}} := \operatorname{argmax}_{e=1}^{k'} |\mathcal{A}_e|$. The complexity is then roughly quadratic in the number of attribute variables and quadratic also in the worst-case number of attributes per variable.

Appendix B. Proofs

Proof [Lemma 1] The likelihood is easily re-written as $\prod_{i=1}^{N} \vartheta_{(d_i, \hat{x}_i \in \hat{o}_i)}$ thanks to the IID assumption, where $\vartheta_{(d_i, \hat{x}_i \in \hat{o}_i)} := \sum_{\hat{x}_i \in \hat{o}_i} \vartheta_{(d_i, \hat{x}_i)}$. By Assumption (2) we have that

$$
\begin{aligned}
p(d, \hat{x} \in \hat{o} | \vartheta) &= \prod_{i=1}^{N} [\sum_{\hat{x}_i \in \hat{o}_i} \vartheta_{(d_i, \hat{x}_i)}] \\
&= \prod_{i=1}^{N} (\sum_{\hat{a}_{i1} \in \hat{o}_{i1}} \cdots \sum_{\hat{a}_{ir} \in \hat{o}_{ir}} \vartheta_{c_i} \prod_{j=1}^{k} \vartheta_{a_{ij}|c_i} \prod_{l=1}^{r} \vartheta_{\hat{a}_{il}|c_i}) \\
&= \prod_{i=1}^{N} \{\vartheta_{c_i} [\prod_{j=1}^{k} \vartheta_{a_{ij}|c_i}] [\prod_{l=1}^{r} \sum_{\hat{a}_{il} \in \hat{o}_{il}} \vartheta_{\hat{a}_{il}|c_i}]\}.
\end{aligned}
$$

Let us focus on $\prod_{i=1}^{N} \vartheta_{c_i}$. By grouping the same chances over the N units, that expression can be re-written as $\prod_{c \in \mathcal{C}} \vartheta_c^{n(c)}$, where $n(c)$ denotes the number of occurrences of c in d. In the same way, we re-write $\prod_{i=1}^{N} \vartheta_{a_{ij}|c}$ for all j and a given c as $\prod_{a_j \in \mathcal{A}_j} \vartheta_{a_j|c}^{n(a_j, c)}$, where $n(a_j, c)$ denotes the number of joint occurrences (a_j, c) in d. Now consider $\sum_{\hat{a}_{il} \in \hat{o}_{il}} \vartheta_{\hat{a}_{il}|c}$, for a given c. Whenever $\hat{o}_{il}$ coincides with $\hat{\mathcal{A}}_l$, $\sum_{\hat{a}_{il} \in \hat{o}_{il}} \vartheta_{\hat{a}_{il}|c} = 1$; and in all the other cases the sum degenerates to a single term, as the observation of the variable value has been precise. Indeed, as clarified in Section 2, according to the definition of missing data we either observe a value exactly, or we do not observe it at all.

This means that we are allowed to drop the missing values from the learning set. In other words, $\prod_{i=1}^{N} (\sum_{\hat{a}_{il} \in \hat{o}_{il}} \vartheta_{\hat{a}_{il}|c})$ becomes, for all l, $\prod_{\hat{a}_l \in \hat{\mathcal{A}}_l} \vartheta_{\hat{a}_l|c}^{n(\hat{a}_l, c)}$, where $n(\hat{a}_l, c)$ denotes the number of joint occurrences $(\hat{a}_l, c)$ in the learning set after dropping the units with missing values of $\hat{A}_l$. ■

We skip the proof of Lemma 2 as it follows a pattern analogous (and easier) to that of the previous proof.

Proof [Theorem 3] Lemmas 1 and 2, together with the chosen prior in (4), allow us to re-write $p(c_M|d^-,\hat{x}^+ \in \hat{o}^+)$ in Equation (3), that is, $p(c_M|d^-,\hat{x}^+ \in \hat{o}^+,s,t)$, as follows:

$$
\begin{aligned}
&p(c_M|d^-,\hat{x}^+ \in \hat{o}^+,s,t) \propto \\
&\int_{\Theta_C} \vartheta_{c_M} \prod_{c\in\mathcal{C}} \vartheta_c^{n(c)+st(c)-1} d\vartheta_C \\
&\cdot \prod_{j=1}^{k} \int_{\Theta_{A_j|c_M}} \vartheta_{a_{Mj}|c_M} \prod_{a_j\in\mathcal{A}_j} \vartheta_{a_j|c_M}^{n(a_j,c_M)+st(a_j,c_M)-1} d\vartheta_{A_j|c_M} \\
&\cdot \prod_{\substack{l=1 \\ \hat{o}_{Ml}\neq\hat{\mathcal{A}}_l}}^{r} \int_{\Theta_{\hat{A}_l|c_M}} \vartheta_{\hat{a}_{Ml}|c_M} \prod_{\hat{a}_l\in\hat{\mathcal{A}}_l} \vartheta_{\hat{a}_l|c_M}^{n(\hat{a}_l,c_M)+st(\hat{a}_l,c_M)-1} d\vartheta_{\hat{A}_l|c_M}.
\end{aligned}
\tag{10}
$$

Expression (10) is obtained by the following steps: (i) substituting in (3) the expressions for the prior, the likelihood and the probability of the next class obtained in Section 3.1; (ii) observing that the Dirichlet densities involved in the integration are independent, so that they can be integrated separately; (iii) dropping the integrals related to classes different from c_M, as each of them equals 1 given that it is missing the term of which to take the expectation; and (iv), for similar reasons, dropping the integrals related to missing attribute values in the unit to classify. Observe that in (10) we have used the following notation: we have denoted by $\vartheta_C \in \Theta_C$ the vector of chances ϑ_c, $c \in \mathcal{C}$. Similarly, $\vartheta_{A_j|c_M} \in \Theta_{A_j|c_M}$ (for all $j = 1,\ldots,k$) resp. $\vartheta_{\hat{A}_l|c_M} \in \Theta_{\hat{A}_l|c_M}$ (for all $l = 1,\ldots,r$) denote the vector of chances $\vartheta_{a_j|c_M}$, $a_j \in \mathcal{A}_j$, resp. $\vartheta_{\hat{a}_l|c_M}$, $\hat{a}_l \in \hat{\mathcal{A}}_l$.

Each integral in (10) represents the expectation of a chance with respect to a Dirichlet density. It is well known that such a calculation can be solved exactly (Kotz et al., 2000, Chap. 49), leading to the statement of the theorem. ■

Proof [Lemma 4] Using (5) we first re-write the function to optimize as follows:

$$
\begin{aligned}
&\Big[\frac{n(c''_M)+st(c''_M)}{n(c'_M)+st(c'_M)}\Big]^{k-1} \prod_{j=1}^{k} \frac{n(a_{Mj},c'_M)+st(a_{Mj},c'_M)}{n(a_{Mj},c''_M)+st(a_{Mj},c''_M)} \cdot \\
&\cdot \prod_{l=1}^{r'} \Big[\frac{n_l(c''_M)+st(c''_M)}{n_l(c'_M)+st(c'_M)} \cdot \frac{n(\hat{a}_{Ml},c'_M)+st(\hat{a}_{Ml},c'_M)}{n(\hat{a}_{Ml},c''_M)+st(\hat{a}_{Ml},c''_M)}\Big].
\end{aligned}
$$

Then we simplify the optimization problem according to the following considerations.

- The objective function is only concerned with the r' non-missing MAR attribute variables.
- For each $j \in \{1,\ldots,k\}$ and $c \in \mathcal{C}$, the objective function contains only one term $t(a_j,c)$. We can choose its value freely in the interval $[0,t(c)]$, as the constraints of type $\sum_{a_j\in\mathcal{A}_j} t(a_j,c) = t(c)$ are always satisfied by assigning the value $[t(c)-t(a_j,c)]$ to any term $t(a_{j'},c)$ ($j' \in \{1,\ldots,r\}$, $j' \neq j$) that does not appear in the objective function, and zero to the others. It follows that constraints of type $\sum_{a_j\in\mathcal{A}_j} t(a_j,c) = t(c)$ can be re-written as $t(a_j,c) \leq t(c)$ for all $j \in \{1,\ldots,k\}$, $c \in \mathcal{C}$. Analogous considerations hold with constraints of type $\sum_{\hat{a}_l\in\hat{\mathcal{A}}_l} t(\hat{a}_l,c) = t(c)$.

- The optimum of the problem is obtained when all the terms $t(a_{Mj}, c'_M)$ and $t(\hat{a}_{Ml}, c'_M)$ go to zero, and all the terms $t(a_{Mj}, c''_M)$ and $t(\hat{a}_{Ml}, c''_M)$ go to $t(c''_M)$. This follows from the preceding observation and because the objective function is made of non-negative terms. Therefore, in the objective function we can replace the mentioned terms with the corresponding values at the optimum.

- Constraint $\sum_{c\in\mathcal{C}} t(c) = 1$ can be replaced by $t(c_{M'}) + t(c_{M''}) = 1$, because the objective function is only concerned with those two classes, as because such a constraint naturally holds at a global optimum. Suppose the last statement is false, that is, that $t(c_{M'}) + t(c_{M''}) < 1$ at a global optimum point. Then we might keep $t(c_{M''})$ fixed and increase $t(c_{M'})$ up to $1 - t(c_{M''})$ (making the $t(\cdot)$ terms of the remaining classes go to zero), so decreasing the optimum value.

These lead to the statement of the lemma once we also remove the variable $t(c'_M)$ from consideration. ■

Proof [Theorem 5] To demonstrate Theorem 5, it is necessary first to introduce two lemmas, whose proofs are given later.

Lemma 8 *Consider Problem* (6)*, assuming that the following two conditions hold: (i) there is no* j *such that* $n(a_{Mj}, c'_M) = 0$ *nor* l *such that* $n(\hat{a}_{Ml}, c'_M) = 0$*; and (ii)* $k + r' > 0$*. Then the objective function* $h(t(c_{M''}))$ *is (strictly) convex over* $(0,1)$*.*

Lemma 9 *Consider Problem* (6)*, and the same assumption used in Lemma 8. Then the logarithmic derivative of the objective function in* $t(c''_M) = 0$ *is* $-\infty$ *if and only if any of the following cases hold:* $n(c''_M) = 0$*; there is* $j \in \{1,\ldots,k\}$ *s.t.* $n(a_{Mj}, c''_M) = 0$*; there is* $l \in \{1,\ldots,r'\}$ *s.t.* $n(\hat{a}_{Ml}, c''_M) = 0$ *and* $n_l(c''_M) \neq 0$*.*

Having introduced Lemmas 8 and 9, we can now prove Theorem 5. The first two steps of the procedure deal with special cases. The first step is justified since in the stated conditions the objective function attains the value zero, and this is a global minimum because the function is non-negative. The second step considers a degenerate problem of classification, in which there are no attribute variables; in this case the minimum is at $t(c_{M''}) = 1$, as the objective function is strictly decreasing, as it can be shown by differentiating.

Otherwise, the function is convex and the local minimum is also the global minimum. The minimum is found by the following procedure:

1. If:

   - $n(c''_M) = 0$ or
   - there is $j \in \{1,\ldots,k\}$ s.t. $n(a_{Mj}, c''_M) = 0$ or
   - there is $l \in \{1,\ldots,r'\}$ s.t. $n(\hat{a}_{Ml}, c''_M) = 0$ and $n_l(c''_M) \neq 0$,

   then let $(\ln h(t(c''_M)))'|_{t(c''_M)=0} := -\infty$, else compute $(\ln h(t(c''_M)))'|_{t(c''_M)=0}$.

2. Compute $(\ln h(t(c''_M)))'|_{t(c''_M)=1}$.

3. If $(\ln h(t(c''_M)))'|_{t(c''_M)=0} \geq 0$, let $\inf_{0<t(c''_M)<1} h(t(c''_M)) := h(0)$. Stop.

4. If $(\ln h(t(c''_M)))'|_{t(c''_M)=1} \leq 0$, let $\inf_{0<t(c''_M)<1} h(t(c''_M)) := h(1)$). Stop.

5. If $(\ln h(t(c''_M)))'|_{t(c''_M)=0} < 0$ and $(\ln h(t(c''_M)))'|_{t(c''_M)=1} > 0$, approximate the minimum numerically by Newton-Raphson's algorithm as in p. 366 of Press et al. (1993).

The procedure is basically a test based on the values of the logarithmic derivative at the extremes of the interval $[0,1]$. If the function were bounded, the last three points would obviously identify the minimum. The preceding points allow the test to be extended to the case of $h(0)$ being unbounded.[18] This happens according to the conditions stated in Lemma 9. We account for these cases by introducing the value $-\infty$ for the derivative in the solution procedure, and treating it as one of the possible values.

As far as Newton-Raphson's method is concerned, it can be applied as the first and second derivatives are available. Note that if the Newton-Raphson is combined with bracketing as in the cited implementation, then its convergence is guaranteed. ■

Proof [Lemma 8] Consider the following definitions to simplify the notation: $\alpha_j := n(a_{Mj}, c'_M)$, $\beta_j := n(a_{Mj}, c''_M)$, $\tilde{\alpha} := n(c'_M)$, $\tilde{\beta} := n(c''_M)$, $\gamma_l := n(\hat{a}_{Ml}, c'_M)$, $\delta_l := n(\hat{a}_{Ml}, c''_M)$, $\tilde{\gamma}_l := n_l(c'_M)$, $\tilde{\delta}_l := n_l(c''_M)$, $x := st(c''_M)$. Re-write Problem (6) accordingly as

$$\begin{aligned}&\inf_{0<x<s} \left(\frac{\tilde{\beta}+x}{\tilde{\alpha}+s-x}\right)^{k-1} \prod_{j=1}^{k} \frac{\alpha_j}{\beta_j+x} \prod_{l=1}^{r'} \left(\frac{\tilde{\delta}_l+x}{\tilde{\gamma}_l+s-x} \cdot \frac{\gamma_l}{\delta_l+x}\right) \\ &= \inf_{0<x<s} h(x).\end{aligned}$$

The objective function is positive on the domain of definition, so we can compute the logarithmic derivative of $h(\cdot)$:

$$\begin{aligned}&\frac{d \ln h(x)}{dx} = \\ &= \frac{k-1}{\tilde{\beta}+x} + \frac{k-1}{\tilde{\alpha}+s-x} - \sum_{j=1}^{k} \frac{1}{\beta_j+x} + \sum_{l=1}^{r'} \frac{1}{\tilde{\delta}_l+x} + \\ &- \sum_{l=1}^{r'} \frac{1}{\delta_l+x} + \sum_{l=1}^{r'} \frac{1}{\tilde{\gamma}_l+s-x}.\end{aligned} \tag{11}$$

Another differentiation leads to the following expression:

$$\begin{aligned}&\frac{d^2 \ln h(x)}{dx^2} = \\ &= -\frac{k-1}{(\tilde{\beta}+x)^2} + \frac{k-1}{(\tilde{\alpha}+s-x)^2} + \sum_{j=1}^{k} \frac{1}{(\beta_j+x)^2} + \\ &- \sum_{l=1}^{r'} \frac{1}{(\tilde{\delta}_l+x)^2} + \sum_{l=1}^{r'} \frac{1}{(\delta_l+x)^2} + \sum_{l=1}^{r'} \frac{1}{(\tilde{\gamma}_l+s-x)^2}.\end{aligned} \tag{12}$$

18. Note that the function is always defined in $t(c_{M''}) = 1$ as the cases that arise when either $n(c'_M) = 0$ or when there is l such that $n_l(c'_M) = 0$, are ruled out by the initial assumption prescribing that there is neither j such that $n(a_{Mj}, c'_M) = 0$, nor l such that $n(\hat{a}_{Ml}, c'_M) = 0$.

There are three possible cases.

1. $k = 0$ and $r' > 0$. In Expression (12), the sum on j disappears, and the other terms are positive except for two of them: $-\frac{1}{(\tilde{\alpha}+s-x)^2}$ and $-\sum_{l=1}^{r'} \frac{1}{(\tilde{\delta}_l+x)^2}$.
   Consider the first one. By definition, $\tilde{\gamma}_l \leq \tilde{\alpha}$ for each l, so that $\frac{1}{(\tilde{\alpha}+s-x)^2} \leq \frac{1}{(\tilde{\gamma}_l+s-x)^2}$, whence $-\frac{1}{(\tilde{\alpha}+s-x)^2} + \sum_{l=1}^{r'} \frac{1}{(\tilde{\gamma}_l+s-x)^2} \geq 0$.
   Now consider the second negative term in Expression (12). By definition, it holds that $\delta_l \leq \tilde{\delta}_l$ for each l, whence $\sum_{l=1}^{r'} \frac{1}{(\tilde{\delta}_l+x)^2} \leq \sum_{l=1}^{r'} \frac{1}{(\delta_l+x)^2}$, or, in other words, $-\sum_{l=1}^{r'} \frac{1}{(\tilde{\delta}_l+x)^2} + \sum_{l=1}^{r'} \frac{1}{(\delta_l+x)^2} \geq 0$.
   This shows that the negative terms in Expression (12) together with the considered positive ones produce non-negative results. Since there are positive terms left in Expression (12), the overall expression is positive. In other words, the function $\ln h(\cdot)$ is (strictly) convex, and so is $h(\cdot)$ (by applying the exponential function).

2. $k > 0$ and $r' = 0$. In this case, there is only one negative term in Expression (12): $-\frac{k-1}{(\tilde{\beta}+x)^2}$. By definition, $\beta_j \leq \tilde{\beta}$ for each j, so that $-\frac{k-1}{(\tilde{\beta}+x)^2} + \sum_{j=1}^{k} \frac{1}{(\beta_j+x)^2} \geq 0$. Since there are positive terms left in Expression (12), the function $h(\cdot)$ is (strictly) convex.

3. $k > 0$ and $r' > 0$. In this case there are two negative terms in Expression (12) (the others being positive): $-\frac{k-1}{(\tilde{\beta}+x)^2}$ and $-\sum_{l=1}^{r'} \frac{1}{(\tilde{\delta}_l+x)^2}$. We have already shown in the two preceding cases that $-\sum_{l=1}^{r'} \frac{1}{(\tilde{\delta}_l+x)^2} + \sum_{l=1}^{r'} \frac{1}{(\delta_l+x)^2} \geq 0$ and that $-\frac{k-1}{(\tilde{\beta}+x)^2} + \sum_{j=1}^{k} \frac{1}{(\beta_j+x)^2} \geq 0$. As there are positive terms left in Expression (12), the overall expression is positive. Also in this case, the function $h(\cdot)$ is (strictly) convex.

■

Proof [Lemma 9] Consider Expression (11) for the logarithmic derivative, and the notation introduced there. Let us focus on the expression obtained from Expression (11) by dropping the terms that are bounded when x approaches zero:

$$\frac{k-1}{\tilde{\beta}+x} - \sum_{j=1}^{k} \frac{1}{\beta_j+x} + \sum_{l=1}^{r'} \frac{1}{\tilde{\delta}_l+x} - \sum_{l=1}^{r'} \frac{1}{\delta_l+x}. \tag{13}$$

Of course Expression (11) goes to $-\infty$ when x approaches zero if and only if Expression (13) does the same, so that we can concentrate on the latter.

($\Leftarrow$) Let us now show that if any of the conditions in the statement holds, Expression (13) goes to $-\infty$ with x approaching zero.

First, consider the case when $\tilde{\beta} = n(c''_M) = 0$, and suppose that $k = 0$. The first term in (13) becomes $-1/x$, the second disappears, and the last two terms sum up to zero. Indeed, when $\tilde{\beta} = 0$, $\tilde{\delta}_l = \delta_l = 0$. The expression becomes $-1/x$, and the implication is verified. In the case when $r' = 0$, it must be $k > 0$ and hence the first two terms in (13) yield $-1/x$ as $\beta_j = 0$ for all j when $\tilde{\beta} = 0$; and the last two terms disappear. Again, the expression becomes $-1/x$, and the implication is verified. The case when both $k > 0$ and $r' > 0$ is verified on the basis of the previous two cases.

Second, consider the case when there is $j \in \{1,\ldots,k\}$ s.t. $\beta_j = n(a_{Mj}, c''_M) = 0$. If also $\tilde{\beta} = 0$, we would fall in the previous case and the implication would be true, so we can assume that $\tilde{\beta} > 0$. In the case when $r' = 0$, the implication would follow immediately. Consider $r' > 0$. The only possibility for Expression (13) not to go to $-\infty$ with x going to zero, is that there is l s.t. $\tilde{\delta}_l = 0$. In this way the term $-1/(\beta_j + x) = -1/x$ would sum up to zero together with the term $1/(\tilde{\delta}_l + x) = 1/x$. But this is not possible: in fact, $\tilde{\delta}_l = 0$ implies $\delta_l = 0$, so that $1/(\tilde{\delta}_l + x) = 1/x$ cancels out together with $-1/(\delta_l + x) = -1/x$. The implication is true also in this case.

Finally, consider the case when there is $l \in \{1,\ldots,r'\}$ s.t. $\delta_l = n(\hat{a}_{Ml}, c''_M) = 0$ and $\tilde{\delta}_l = n_l(c''_M) \neq 0$. Observe that the term $-1/(\delta_l + x) = -1/x$ cannot cancel out because of other terms: in fact, with the l-th terms, we have that $\tilde{\delta}_l > 0$; if there is another l' s.t. $\tilde{\delta}_l = 0$, also $\delta_l = 0$ and the related two terms sum up to zero. The only remaining possibility to cancel $-1/(\delta_l + x) = -1/x$ out is that $\tilde{\beta} = 0$. But in this case we know that the implication must be true. It follows that also in this case the implication is verified.

($\Rightarrow$) Let us now consider the reverse implication. If Expression (13) goes to $-\infty$ with x going to zero, at least one of the terms in such an expression must do the same. By considering the terms one by one, the implication is proved in a trivial way. ■

Proof [Theorem 6]

Problem

$$\min_{d \in o} \inf_{p(\theta) \in \mathcal{P}(\theta)} p(c'_M | d^-, \hat{x}^+ \in \hat{o}^+) / p(c''_M | d^-, \hat{x}^+ \in \hat{o}^+),$$

where the set of prior densities is defined according to Expression (4), the constraints are those described in Section 3.1.1, and the probabilities in the function to optimize are defined as in (5), is equivalent to the following:

$$\min_{d \in o} \inf_{0 < t(c''_M) < 1} \{ [\frac{n(c''_M) + st(c''_M)}{n(c'_M) + s - st(c''_M)}]^{k-1} \prod_{j=1}^{k} \frac{n(a_{Mj}, c'_M)}{n(a_{Mj}, c''_M) + st(c''_M)} \cdot$$
$$\cdot \prod_{l=1}^{r'} [\frac{n_l(c''_M) + st(c''_M)}{n_l(c'_M) + s - st(c''_M)} \cdot \frac{n(\hat{a}_{Ml}, c'_M)}{n(\hat{a}_{Ml}, c''_M) + st(c''_M)}] \}.$$

The problem can be then re-written as follows:

$$\inf_{0 < t(c''_M) < 1} \min_{d \in o} \{ [\frac{n(c''_M) + st(c''_M)}{n(c'_M) + s - st(c''_M)}]^{k-1} \prod_{j=1}^{k} \frac{n(a_{Mj}, c'_M)}{n(a_{Mj}, c''_M) + st(c''_M)} \cdot$$
$$\cdot \prod_{l=1}^{r'} [\frac{n_l(c''_M) + st(c''_M)}{n_l(c'_M) + s - st(c''_M)} \cdot \frac{n(\hat{a}_{Ml}, c'_M)}{n(\hat{a}_{Ml}, c''_M) + st(c''_M)}] \}. \quad (14)$$

This is obtained by inverting the order of the optimizations and using Expression (6). Note that the inner minimization only affects the product over j in the objective function. Moreover, the counts in such a product attain the same value at the optimum for any choice of $t(c''_M)$, leading to the following product: $\prod_{j=1}^{k} \underline{n}(a_{Mj}, c'_M) / [\overline{n}(a_{Mj}, c''_M) + st(c''_M)]$. Note that the pairwise counts are optimized separately as they do not affect each other. These arguments enable us to reduce

Problem (14) as:

$$\inf_{0<t(c''_M)<1}\Big\{\left[\frac{n(c''_M)+st(c''_M)}{n(c'_M)+s-st(c''_M)}\right]^{k-1}\prod_{j=1}^{k}\frac{\underline{n}(a_{Mj},c'_M)}{\overline{n}(a_{Mj},c''_M)+st(c''_M)}\cdot \tag{15}$$
$$\cdot\prod_{l=1}^{r'}\left[\frac{n_l(c''_M)+st(c''_M)}{n_l(c'_M)+s-st(c''_M)}\cdot\frac{n(\hat{a}_{Ml},c'_M)}{n(\hat{a}_{Ml},c''_M)+st(c''_M)}\right]\Big\},$$

with $\underline{n}(a_{Mj},c'_M) := \min_{d\in o} n(a_{Mj},c'_M)$ and $\overline{n}(a_{Mj},c'_M) := \max_{d\in o} n(a_{Mj},c'_M)$. At this point, Problem (15) is an instance of Problem (6) obtained for a specific choice of a complete data set d in o. ■

Proof [Theorem 7] Let us focus the attention on the term $\min_{a_{Me}\in\mathcal{A}_e}\frac{\underline{n}(a_{Me},c'_M)}{\overline{n}(a_{Me},c''_M)+x}$. As a function of x, this is the lower envelope of the set of functions $\{\frac{\underline{n}(a_{Me},c'_M)}{\overline{n}(a_{Me},c''_M)+x} : a_{Me}\in\mathcal{A}_e\} =: \mathcal{F}_e$. Consider a value x such that the lower envelope coincides with two different functions in $\mathcal{F}_e$ before and after x. This implies that the two functions must cross at x, because the function that was over the other before x must be under it after x. In other words, the set of points where any two different functions in $\mathcal{F}_e$ cross (let us call them *partition points*), contains the points where the function matched by the lower envelope changes.

We can identify the partition points in the following way. Consider any two different functions in $\mathcal{F}_e$: $\underline{n}(a_{Me'},c'_M)/[\overline{n}(a_{Me'},c''_M)+x]$ and $\underline{n}(a_{Me''},c'_M)/[\overline{n}(a_{Me''},c''_M)+x]$. Assume that $\underline{n}(a_{Me'},c'_M)\neq\underline{n}(a_{Me''},c'_M)$: in the opposite case the only way for the two functions to cross would be that also $\overline{n}(a_{Me'},c''_M)=\overline{n}(a_{Me''},c''_M)$, but the two functions would be the same, which is excluded a priori. The two functions cross when $\underline{n}(a_{Me'},c'_M)/[\overline{n}(a_{Me'},c''_M)+x]=\underline{n}(a_{Me''},c'_M)/[\overline{n}(a_{Me''},c''_M)+x]$, and this happens if and only if

$$x=\frac{\overline{n}(a_{Me'},c''_M)\underline{n}(a_{Me''},c'_M)-\underline{n}(a_{Me'},c'_M)\overline{n}(a_{Me''},c''_M)}{\underline{n}(a_{Me'},c'_M)-\underline{n}(a_{Me''},c'_M)}. \tag{16}$$

In other words, there is at most one partition point for any two different functions,[19] and we can identify it easily by (16). The maximum number of partition points is also the number of distinct pairs of different functions in $\mathcal{F}_e$, which is at most $(|\mathcal{A}_e|^2-|\mathcal{A}_e|)/2$.[20]

The crucial observation here is that the lower envelope matches a single function of $\mathcal{F}_e$ in any sub-interval of $(0,s)$ that does not contain partition points; and we are always able to select an element of $\mathcal{A}_e$ that gives rise to the matched function. In other words, the partition points can be used to define a partition of $(0,s)$, in the sub-intervals of which the lower envelop matches a single function that we are able to characterize by an element of $\mathcal{A}_e$. This argument is easily extended to the entire product over e in (9): we compute the set of partition points for each term of the product, and take their union. The union defines a partition $\mathcal{J}$ of $(0,s)$, in which every sub-interval $\mathcal{I}$ is associated with a single tuple $(a^{\mathcal{I}}_{M1},\ldots,a^{\mathcal{I}}_{Mk'})\in\mathcal{A}_1\times\cdots\times\mathcal{A}_{k'}$, so that

$$\prod_{e=1}^{k'}[\min_{a_{Me}\in\mathcal{A}_e}\frac{\underline{n}(a_{Me},c'_M)}{\overline{n}(a_{Me},c''_M)+x}]=\prod_{e=1}^{k'}\frac{\underline{n}(a^{\mathcal{I}}_{Me},c'_M)}{\overline{n}(a^{\mathcal{I}}_{Me},c''_M)+x}.$$

19. Note that it could be outside $[0,s]$.

20. Note that some partition points may be such that the lower envelope does not change the matched function in $\mathcal{F}_e$, and so they could be discarded. As an example, assume that the value x identified by (16) is a partition point only for the two mentioned functions, and that there is a third function below the other two at x. In this case the former two functions are not involved in the determination of the lower envelope at x.

Problem (9) then becomes the following:

$$\min_{\mathcal{I}\in\mathcal{J}}\inf_{x\in\mathcal{I}}\{[\frac{n(c''_M)+x}{n(c'_M)+s-x)}]^{k-1}\prod_{e=1}^{k'}\frac{\underline{n}(a^{\mathcal{I}}_{Me},c'_M)}{\overline{n}(a^{\mathcal{I}}_{Me},c''_M)+x}\cdot$$
$$\cdot\prod_{j=k'+1}^{k}\frac{\underline{n}(a_{Mj},c'_M)}{\overline{n}(a_{Mj},c''_M)+x}\cdot$$
$$\cdot\prod_{l=1}^{r'}[\frac{n_l(c''_M)+x}{n_l(c'_M)+s-x}\cdot\frac{n(\hat{a}_{Ml},c'_M)}{n(\hat{a}_{Ml},c''_M)+x}]\},$$

where the inner optimization can be solved by the procedure given in Section (3.2), applying it to the interval $\mathcal{I}$ rather than $(0,s)$.

Now it is easy to show that the procedure in Fig. 6 solves Problem (8). The first step of the procedure builds the set of partition points for the functions in $\mathcal{F}_e$, and the next one defines the partition of $(0,s)$. Since the function is convex on each sub-interval, the remaining steps solve a minimization problem on each sub-interval; finally, the global minimum is selected. ■

Appendix C. Experimental Results Data Set by Data Set

Detailed results by data set are shown in Tables 6 and 7, which refer respectively to the MAR and non-MAR setting.

Data set	*NBC*	*NCC2-MAR*				*NCC2-nonMAR*				*Subsets of instances*				
	Acc. (%)	Det. (%)	Single-Acc. (%)	SetAcc. (%)	Indet.OutSize	Det. (%)	Single-Acc. (%)	SetAcc. (%)	Indet.OutSize	NCC2-MAR D (%)	NCC2-MAR I (%)	NCC2-nonMAR D (%)	NCC2-nonMAR I (%)	ΔNCC2-nonMAR (%)
ecoli (8 cl.)	85.0	84.2	88.5	92.2	3.8	54.4	95.0	96.6	5.2	88.5	66.6 (6.4)	95.0	72.9 (3.4)	74.7 (4.3)
glass (6 cl.)	67.5	52.1	80.4	79.5	2.4	22.4	81.9	94.8	3.7	80.4	53.6 (3.9)	81.9	63.3 (2.6)	61.6 (4.9)
haberman (2 cl.)	72.0	91.8	74.2	100.0	2.0	66.7	80.8	100.0	2.0	74.2	47 (10.9)	80.8	54.1 (4.8)	56.5 (6.2)
kr-kp (2 cl.)	86.8	98.4	87.4	100.0	2.0	5.4	100.0	100.0	2.0	87.4	48 (10.2)	100.0	86.1 (0.4)	86.7 (0.4)
letter (26 cl.)	72.4	91.6	77.0	57.7	2.7	13.3	99.0	97.9	18.7	77.0	22.0 (1.2)	99.0	68.3 (0.2)	68.4 (0.2)
monks1 (2 cl.)	70.0	87.4	72.7	100.0	2.0	39.7	82.5	100.0	2.0	72.7	51.2 (5.8)	82.5	61.6 (3.0)	64.3 (3.6)
monks2 (2 cl.)	62.2	86.6	63.4	100.0	2.0	15.3	81.2	100.0	2.0	63.4	54.6 (6.8)	81.2	58.9 (2.2)	59.7 (2.3)
monks3 (2 cl.)	95.3	99.2	95.5	100.0	2.0	67.3	96.7	100.0	2.0	95.6	64 (25)	96.6	91.7 (3.2)	92.5 (3.1)
nursery (5 cl.)	87.0	97.5	87.4	78.9	2.0	49.0	98.4	99.6	2.5	87.4	66.1 (5.3)	98.4	76.0 (1.0)	76.0 (0.9)
optdigits (10 cl.)	91.2	95.2	93.5	86.9	2.7	14.8	99.9	99.9	8.0	93.5	46.9 (3.6)	99.9	89.7 (0.2)	89.8 (0.2)
pendigits (10 cl.)	87.2	96.3	89.5	81.8	2.5	29.2	97.1	99.1	7.2	89.5	26.7 (2.7)	97.1	83.1 (0.3)	83.2 (0.3)
segment (7 cl.)	91.0	89.3	95.6	96.7	3.7	31.7	98.8	99.9	6.1	95.6	52.5 (3.9)	98.8	87.4 (0.7)	88.7 (0.7)
sonar (2 cl.)	84.4	90.9	86.9	100.0	2.0	41.8	97.9	100.0	2.0	86.9	59 (14.3)	97.9	74.6 (3.1)	77.4 (2.6)
spambase (2 cl.)	89.1	99.5	89.4	100.0	2.0	42.1	97.2	100.0	2.0	89.4	31 (13.4)	97.2	83.3 (0.5)	83.7 (0.4)
spect (2 cl.)	76.1	90.8	79.3	100.0	2.0	53.9	90.2	100.0	2.0	79.3	44 (12.6)	90.2	59.8 (4.1)	63.7 (4.3)
splice (3 cl.)	94.5	97.2	96.4	96.2	2.2	0.0	100.0	100.0	3.0	96.4	49.5 (7.6)	100.0	94.9 (0.2)	94.5 (0.2)
waveform (3 cl.)	81.3	99.0	81.6	99.9	2.0	19.7	94.5	100.0	2.3	81.6	50.3 (7.6)	94.5	78.0 (0.4)	78.2 (0.2)
yeast (10 cl.)	56.6	91.9	58.6	70.0	2.3	28.0	69.1	91.1	3.7	58.5	34.9 (6.7)	69.1	51.7 (1.7)	51.7 (1.7)

Table 6: Results data set by data set under the MAR-MP setting. For each data set, the number of classes is reported aside the name. For NccMarI, NccI, ΔNCC2-nonMAR, the standard deviation is reported into brackets; for the remaining indicators, the standard deviation is smaller than 2 percentage points in the large majority of cases.

Data set	*NBC*	*NCC2-MAR*				*NCC2-nonMAR*				*Subsets of instances*				
	Acc. (%)	Det. (%)	Single-Acc. (%)	SetAcc. (%)	Indet.OutSize	Det. (%)	Single-Acc. (%)	SetAcc. (%)	Indet.OutSize	NCC2-MAR D (%)	NCC2-MAR I (%)	NCC2-nonMAR D (%)	NCC2-nonMAR I (%)	ΔNCC2-nonMAR (%)
ecoli (8cl.)	85.2	83.2	88.9	92.6	3.7	69.6	94.9	94.1	3.8	88.9	67.2 (5.7)	94.9	62.9 (3.0)	60.4 (6.7)
glass (6 cl.)	67.5	47.5	81.7	79.7	2.4	29.6	83.5	91.1	3.0	81.7	54.8 (3.4)	83.5	60.7 (2.2)	58.7 (6.9)
haberman (2 cl.)	72.0	93.5	73.8	100.0	2.0	79.4	78.5	100.0	2.0	73.8	47 (12.2)	78.5	46.6 (5.7)	47.1 (9.6)
kr-kp (2 cl.)	88.0	98.5	88.6	100.0	2.0	52.9	96.8	100.0	2.0	88.6	49.1 (9.4)	96.8	78.1 (1.0)	79.0 (0.9)
letter (26 cl.)	72.9	91.9	77.4	58.5	2.8	35.3	96.7	91.8	11.7	77.4	22.0 (1.1)	96.7	59.9 (0.3)	60.0 (0.3)
monks1 (2 cl.)	71.0	87.5	73.7	100.0	2.0	58.8	80.1	100.0	2.0	73.7	51.8 (5.5)	80.1	57.9 (2.4)	60.6 (3.2)
monks2 (2 cl.)	62.2	86.7	63.5	100.0	2.0	40.2	72.5	100.0	2.0	63.5	53.5 (6.0)	72.5	55.6 (2.2)	56.3 (2.5)
monks3 (2 cl.)	95.5	99.8	95.6	100.0	2.0	90.5	97.2	100.0	2.0	96.3	64 (31.3)	87.6	73 (14.1)	73 (14.1)
nursery (5 cl.)	87.8	97.5	88.3	80.3	2.0	68.8	97.2	99.6	2.3	88.3	67.8 (4.5)	97.2	67.1 (1.5)	66.9 (1.2)
optdigits (10 cl.)	91.3	95.3	93.5	86.2	2.6	54.6	99.0	98.4	4.9	93.5	45.2 (3.1)	99.0	82.0 (0.4)	82.4 (0.4)
pendigits (10 cl.)	87.4	96.3	89.7	81.8	2.5	57.1	95.5	96.8	5.4	89.7	25.7 (2.5)	95.5	76.5 (0.5)	77.3 (0.5)
segment (7 cl.)	91.6	89.5	95.9	97.6	3.7	54.8	97.7	99.3	5.5	95.9	54.9 (3.0)	97.7	84.2 (0.9)	87.8 (1.0)
sonar (2 cl.)	83.8	91.8	87.1	100.0	2.0	56.5	94.4	100.0	2.0	87.1	46 (17)	94.4	69.9 (3.2)	75.2 (3.2)
spambase (2 cl.)	88.9	99.5	89.2	100.0	2.0	76.1	94.1	100.0	2.0	89.2	39 (12.8)	94.1	72.6 (0.8)	73.3 (0.8)
spect (2 cl.)	73.4	88.5	78.1	100.0	2.0	67.4	83.5	100.0	2.0	78.1	36.5 (8.8)	83.5	52.3 (4.0)	60.9 (4.7)
splice (3 cl.)	95.1	97.6	96.8	96.3	2.2	0.1	100.0	100.0	2.8	96.2	52.1 (5.5)	100.0	95.1 (0.2)	95.0 (0.2)
waveform (3 cl.)	81.4	99.1	81.6	99.8	2.0	54.1	89.6	100.0	2.1	81.6	49.2 (5.5)	89.6	71.7 (0.6)	72.0 (0.6)
yeast (10 cl.)	57.2	91.5	59.1	72.1	2.3	55.1	67.3	87.5	2.9	59.0	36.7 (6.7)	67.3	44.8 (1.8)	45.1 (1.9)

Table 7: Results data set by data set under the MAR-MP setting. For each data set, the number of classes is reported aside the name. For NccMarI, NccI, ΔNCC2-nonMAR, the standard deviation is reported into brackets; for the remaining indicators, the standard deviation is smaller than 2 percentage points in the large majority of cases.

References

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.

D. F. Heitjan and D. B. Rubin. Ignorability and coarse data. *The Annals of Statistics*, 19(4):2244–2253, 1991.

M. Jaeger. Ignorability in statistical and probabilistic inference. *Journal of Artificial Intelligence Research*, 24:889–917, 2005.

S. Kotz, N. Balakrishnan, and N. L. Johnson. *Continuous Multivariate Distributions, Volume 1: Models and Applications*. Series in Probability and Statistics. Wiley, New York, 2000.

R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. Wiley, New York, 1987.

C. F. Manski. *Partial Identification of Probability Distributions*. Springer-Verlag, New York, 2003.

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, Cambridge, 1993. 2nd edition.

M. Ramoni and P. Sebastiani. Robust Bayes classifiers. *Artificial Intelligence*, 125(1–2):209–226, 2001.

D. B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, New York, 1991.

P. Walley. Inferences from multinomial data: learning about a bag of marbles. *J. R. Statist. Soc. B*, 58(1):3–57, 1996.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann, 2005.

M. Zaffalon. Statistical inference of the naive credal classifier. In G. de Cooman, T. L. Fine, and T. Seidenfeld, editors, *ISIPTA '01: Proceedings of the Second International Symposium on Imprecise Probabilities and Their Applications*, pages 384–393, The Netherlands, 2001. Shaker.

M. Zaffalon. The naive credal classifier. *Journal of Statistical Planning and Inference*, 105(1):5–21, 2002.

M. Zaffalon. Credible classification for environmental problems. *Environmental Modelling and Software*, 20(8):1003–1012, 2005a.

M. Zaffalon. Conservative rules for predictive inference with incomplete data. In F. G. Cozman, R. Nau, and T. Seidenfeld, editors, *ISIPTA '05: Proceedings of the Fourth International Symposium on Imprecise Probabilities and Their Applications*, pages 406–415, Manno, Switzerland, 2005b. SIPTA.

M. Zaffalon, K. Wesnes, and O. Petrini. Reliable diagnoses of dementia by the naive credal classifier inferred from incomplete cognitive data. *Artificial Intelligence in Medicine*, 29(1–2):61–79, 2003.

Journal of Machine Learning Research 9 (2008) 623-626 Submitted 10/07; Revised 3/08; Published 4/08

A Library for Locally Weighted Projection Regression

Stefan Klanke S.KLANKE@ED.AC.UK
Sethu Vijayakumar SETHU.VIJAYAKUMAR@ED.AC.UK
School of Informatics
University of Edinburgh
Edinburgh, EH9 3JZ, UK

Stefan Schaal SSCHAAL@USC.EDU
Dept. of Computer Science
University of Southern California
Los Angeles, CA 90089-2520, USA

Editor: Soeren Sonnenburg

Abstract

In this paper we introduce an improved implementation of locally weighted projection regression (LWPR), a supervised learning algorithm that is capable of handling high-dimensional input data. As the key features, our code supports multi-threading, is available for multiple platforms, and provides wrappers for several programming languages.

Keywords: regression, local learning, online learning, C, C++, Matlab, Octave, Python

1. Introduction

Locally weighted projection regression (LWPR) is an algorithm that achieves nonlinear function approximation in high dimensional spaces even in the presence of redundant and irrelevant input dimensions (Vijayakumar et al., 2002). At its core, it uses locally linear models, spanned by a small number of univariate regressions in selected directions in input space. This nonparametric local learning system learns rapidly with second order learning methods based on incremental training, using statistically sound stochastic cross validation.

The implementation of LWPR we present in this work is written in low-level C, requires no additional libraries, and comes with convenient wrappers for C++, Matlab and Python. Together with documentation, tutorials and additional supporting materials, it is freely available for download from `http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr`.

2. The LWPR Algorithm

The goal of LWPR is to learn a regression function from training data that incrementally arrive as input-output tuples $(\mathbf{x}_i, y_i)$, where we assume univariate output data for now. The LWPR regression function is constructed by blending local linear models $\psi_k(\mathbf{x})$ in the form

$$f(\mathbf{x}) = \frac{1}{W(\mathbf{x})} \sum_{k=1}^{K} w_k(\mathbf{x}) \psi_k(\mathbf{x}), \quad W(\mathbf{x}) = \sum_{k=1}^{K} w_k(\mathbf{x}). \tag{1}$$

Here, $w_k(\mathbf{x})$ is a locality kernel that defines the area of validity of the local models (also termed "receptive field"), which is usually modeled by a Gaussian

$$w_k(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_k)^T\mathbf{D}_k(\mathbf{x}-\mathbf{c}_k)\right), \tag{2}$$

where $\mathbf{c}_k$ is the centre of the k^{th} linear model and $\mathbf{D}_k$ is its distance metric. During training, all updates to the local models are weighted by their activation $w_k(\mathbf{x})$, facilitating fully localised and inpedendent learning. If no existing local model yields an activation above a certain threshold (e.g., $w_{gen} = 0.1$), a new local model is created with its center $\mathbf{c}_k$ set to the input datum. Thus, the number K of local models is adapted automatically.

For learning the linear models $\psi_k(\mathbf{x})$ themselves, LWPR employs an online formulation of weighted partial least squares (PLS) regression. In particular, within each local model[1] the input data $\mathbf{x}$ is projected along selected directions $\mathbf{u}_i$, yielding "latent" variables s_i with

$$s_i = \mathbf{u}_i^T\mathbf{x}_{i-1}, \qquad \mathbf{x}_i = \mathbf{x}_{i-1} - s_i\mathbf{p}_i = \mathbf{x}_{i-1} - \mathbf{p}_i\mathbf{u}_i^T\mathbf{x}_{i-1}, \qquad \mathbf{x}_0 = \mathbf{x} - \bar{\mathbf{x}},$$

where the vectors $\mathbf{p}_i$ ensure orthogonality of the projections, and $\bar{\mathbf{x}}$ is the weighted mean of the input data (as seen through the receptive field). The output of the local model is then formed by a linear combination of the latent variables (also called PLS factor loadings)

$$\psi_k(\mathbf{x}) = \beta_0 + \sum_{i=1}^{R}\beta_i s_i. \tag{3}$$

The number R of regression directions is automatically adapted to the local dimensionality of the training data, and the parameters $\mathbf{u}_i$, $\mathbf{p}_i$, and β_i can be robustly estimated from accumulating certain statistics of the training set (for details please see Vijayakumar et al., 2005). In a similar fashion, the distance metrics $\mathbf{D}$ can be adapted using stochastic cross-validation, such that the input space is covered by wide receptive fields in regions of low curvature, and narrow receptive fields where the curvature is high. The initial distance metric assigned to a newly created receptive field is a rather critical open parameter. If available, one should use an estimate of the Hessian of the function that is to be approximated. As a valuable feature of the algorithm, LWPR can optionally yield confidence bounds for its predictions (Vijayakumar et al., 2005).

There are two possible choices with regard to handling multivariate output data: First, the local models itself could be made multivariate, in which case only one layer of receptive fields is needed. Alternately, one can learn all output dimensions independently, effectively using univariate PLS regression (see, e.g., Garthwaite, 1994) within the local models. In the present implementation we use the latter approach, which is computationally more costly for many output dimensions, but usually exhibits superior prediction performance.

LWPR is an algorithm that is particularly suited (and recommended) for regression problems with a sufficiently large number of training examples. For use in small data set scenarios, the data should be presented to the algorithm multiple times in random order.

3. Details of the Implementation

This section describes several important aspects of our implementation, all of which are related to execution speed.

1. For notational convenience we drop the index k of the local model.

3.1 Data Structures and Memory Allocation

On modern computers, the speed at which many algorithms run does not only depend on the processor, but also critically on the speed of memory access. We designed our library so that memory access is as continuous as possible, thus minimising the chance of cache misses. In our library, all variables[2] of each local model are allocated together in a contiguous piece of memory. Moreover, we allocate "workspaces" as part of the LWPR model for storing intermediate results, such that no further allocations are needed during the computations.

3.2 Multithreaded Updates and Predictions

Since the LWPR algorithm is designed to be parallelisable (the local models learn independently), and nowadays even off-the-shelf mainstream computers are equipped with multi-core processors, we constructed our LWPR library such that it is capable of running the computations in multiple threads. The library currently supports POSIX threads on Linux/Unix machines and native threads on the Windows platform. The desired number of threads has to be defined at compile time, and threading can be deactivated altogether.

In order to balance the overhead of creating threads with the expected reduction in computation time, we chose to implement two complimentary strategies for distributing the work. *Predictions* of the LWPR model are split up on a per-output-dimension level, which implies that LWPR models for one-dimensional output data will not be accelerated by using multiple threads. The more costly *update* operations, however, are distributed across multiple threads on the receptive field level, so even single-output models may benefit (see Fig. 1).

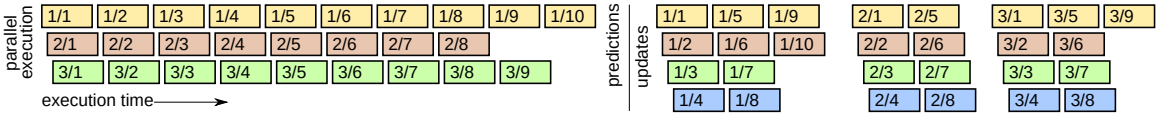


Figure 1: Illustration of our threading implementation for the case of (up to) 4 threads. The example LWPR model has 3 output dimensions with 10, 8, and 9 receptive fields, respectively. A label M/N denotes the N-th receptive field in the M-th output sub-model. For predictions, each thread handles a different output dimension. For updates, the workload of each output dimension is split up among threads, and outputs are handled one after another.

3.3 Fast Computation of Predictions and Their Gradients

For some applications of LWPR, it may be useful to compute analytic derivatives of the model, for example, to retrieve the Jacobian from a learned forward kinematics relation. The gradient of a single predicted output (1) is given by

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \frac{1}{W}\sum_k \left(\frac{\partial w_k}{\partial \mathbf{x}}\psi_k + w_k \frac{\partial \psi_k}{\partial \mathbf{x}} \right) - \frac{1}{W^2}\sum_k w_k \psi_k \sum_l \frac{\partial w_l}{\partial \mathbf{x}},$$

2. Local models require no less than 27 variables, most of them vectors or matrices, for storing all the memory terms and sufficient statistics, etc.

where $\frac{\partial w_k}{\partial \mathbf{x}} = -w_k \mathbf{D}_k(\mathbf{x}-\mathbf{c}_k)$ for the Gaussian kernel (2). The local models $\psi_k(\mathbf{x})$ are computed by PLS recursions, and therefore also the gradients of (3) have to be calculated in this rather costly way:

$$\begin{aligned}\frac{\partial \psi}{\partial \mathbf{x}} &= \sum_{i=1}^{R} \beta_i \frac{\partial s_i}{\partial \mathbf{x}_0} = \sum_{i=1}^{R} \beta_i \left(\mathbf{u}_i^T \frac{\partial \mathbf{x}_{i-1}}{\partial \mathbf{x}_0} \right)^T = \sum_{i=1}^{R} \beta_i \left(\frac{\partial \mathbf{x}_{i-1}}{\partial \mathbf{x}_{i-2}} \cdots \frac{\partial \mathbf{x}_1}{\partial \mathbf{x}_0} \right)^T \mathbf{u}_i \\ &= \beta_1 \mathbf{u}_1 + \beta_2 (\mathbf{I} - \mathbf{u}_1 \mathbf{p}_1^T)\mathbf{u}_2 + \beta_3 (\mathbf{I} - \mathbf{u}_1 \mathbf{p}_1^T)(\mathbf{I} - \mathbf{u}_2 \mathbf{p}_2^T)\mathbf{u}_3 + \dots \ .\end{aligned}$$

However, between updates (for example, during prediction-only cycles) or after training has finished, the slopes $\frac{\partial \psi}{\partial \mathbf{x}}$ of the local models do not change. Our implementation exploits this by memorising the slopes once a gradient is calculated, and directly using these slopes for predictions without running through the PLS recursions.

3.4 Matlab Interface

Our library started its life as a Matlab-only implementation, and therefore the Matlab struct describing an LWPR model is practically identical to the data structure used within the C library. When calling the C functions from Matlab via MEX-wrappers, however, these data structures normally have to be converted back and forth, which is time-consuming. Therefore, we added a special storage mechanism to our MEX-wrappers, which allows us to transfer Matlab data to and from C-managed memory. Then, updates and predictions of an LWPR model can be computed by calling the "normal" MEX-functions, but passing a certain reference identifier instead of the complete Matlab struct. As an illustration of the performance gain, we trained an LWPR model on a 2D toy data set. For accomplishing 10,000 updates, the Matlab-only implementation took roughly 100 seconds, using the MEX wrappers alone took 11.3s, but with our storage mechanism the task is finished after only 0.8s. The Matlab implementation—including the MEX-wrappers and the storage scheme—is also compatible with recent versions of Octave.[3]

Acknowledgments

This work has been carried out with support from the EU FP6 SENSOPAC project to SK and SV, funded by the European Commission.

References

P. H. Garthwaite. An interpretation of partial least squares. *Journal of the American Statistical Association*, 89(425):122–127, 1994.

S. Vijayakumar, A. D'Souza, T. Shibata, J. Conradt, and S. Schaal. Statistical learning for humanoid robots. *Autonomous Robots*, 12(1):55–69, 2002.

S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17:2602–2634, 2005.

3. Octave is a free Matlab clone available at `http://www.octave.org`. We tested our library against version 2.9.12.

Journal of Machine Learning Research 9 (2008) 627-650 Submitted 4/07; Revised 12/07; Published 4/08

Trust Region Newton Method for Large-Scale Logistic Regression

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

S. Sathiya Keerthi SELVARAK@YAHOO-INC.COM
Yahoo! Research
Santa Clara, CA 95054, USA

Editor: Alexander Smola

Abstract

Large-scale logistic regression arises in many applications such as document classification and natural language processing. In this paper, we apply a trust region Newton method to maximize the log-likelihood of the logistic regression model. The proposed method uses only approximate Newton steps in the beginning, but achieves fast convergence in the end. Experiments show that it is faster than the commonly used quasi Newton approach for logistic regression. We also extend the proposed method to large-scale L2-loss linear support vector machines (SVM).

Keywords: logistic regression, newton method, trust region, conjugate gradient, support vector machines

1. Introduction

The logistic regression model is useful for two-class classification. Given data $\mathbf{x}$ and weights $(\mathbf{w}, b)$, it assumes the following probability model

$$P(y = \pm 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(-y(\mathbf{w}^T\mathbf{x} + b))},$$

where y is the class label. If training instances are $\mathbf{x}_i, i = 1, \ldots, l$ and labels are $y_i \in \{1, -1\}$, one estimates $(\mathbf{w}, b)$ by minimizing the negative log-likelihood:

$$\min_{\mathbf{w},b} \quad \sum_{i=1}^{l} \log(1 + e^{-y_i(\mathbf{w}^T\mathbf{x}_i + b)}).$$

There are numerous applications of logistic regression. It can be extended to a multi-class classification model, which is a special case of conditional random fields, and is also called the maximum entropy model in the natural language processing community.

To have a simpler derivation without considering the bias term b, one often augments each instance with an additional dimension:

$$\mathbf{x}_i^T \leftarrow [\mathbf{x}_i^T, 1] \qquad \mathbf{w}^T \leftarrow [\mathbf{w}^T, b]. \tag{1}$$

Moreover, to obtain good generalization abilities, one adds a regularization term $\mathbf{w}^T\mathbf{w}/2$, so in this paper we consider the following form of regularized logistic regression:

$$\min_{\mathbf{w}} f(\mathbf{w}) \equiv \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\log(1+e^{-y_i\mathbf{w}^T\mathbf{x}_i}), \tag{2}$$

where $C > 0$ is a parameter decided by users so that the two terms in (2) are balanced. One can easily check that (2) is twice continuously differentiable.

There are many methods for training logistic regression models. In fact, most unconstrained optimization techniques can be considered. Those which have been used in large-scale scenarios are, for example, iterative scaling (Darroch and Ratcliff, 1972; Pietra et al., 1997; Goodman, 2002; Jin et al., 2003), nonlinear conjugate gradient, quasi Newton (in particular, limited memory BFGS) (Liu and Nocedal, 1989; Benson and Moré, 2001), and truncated Newton (Komarek and Moore, 2005). All these optimization methods are iterative procedures, which generate a sequence $\{\mathbf{w}^k\}_{k=1}^{\infty}$ converging to the optimal solution of (2). One can distinguish them according to the following two extreme situations of optimization methods:

Low cost per iteration;	$\longleftrightarrow$	High cost per iteration;
slow convergence.		fast convergence.

For instance, iterative scaling updates one component of $\mathbf{w}$ at a time, so the cost per iteration is low but the number of iterations is high. In contrast, Newton method, which is expensive at each iteration, has very fast convergence rates. Many have attempted to compare these methods for logistic regression. Minka (2003) experiments with small data sets, and Malouf (2002) has done an extensive comparison for large-scale sets. Currently, most argue that the limited memory BFGS method is the most efficient and effective (e.g., Malouf, 2002; Sutton and McCallum, 2006) and references therein). In this article, we aim at situations for which both l (number of instances) and n (number of features) are very large. In addition, the data instances $\mathbf{x}_1, \ldots, \mathbf{x}_l$ are sparse (i.e., many feature values are zero). Many recent applications from document classification and computational linguistics are of this type.

Truncated Newton methods have been an effective approach for large-scale unconstrained optimization, but their use for logistic regression has not been fully exploited. Though Komarek and Moore (2005) have considered this type of methods, their implementation does not follow rigorous optimization derivations, and hence may not be guaranteed to obtain the minimum of the negative log-likelihood. In Section 2, we discuss an efficient and robust truncated Newton method for logistic regression. This approach, called trust region Newton method, uses only approximate Newton steps in the beginning, but takes full Newton directions in the end for fast convergence.

In Sections 3 and 4, we discuss some existing optimization methods for logistic regression and conduct comparisons. As Newton method uses the exact Hessian (second derivative), it has quadratic convergence near the optimum. Results indicate that our proposed method converges much faster than quasi-Newton methods, which use only an approximate Hessian. Section 5 investigates a variant of our proposed method by using preconditioned conjugate gradients in the trust region framework. In Section 6, we extend the proposed trust region method to solve L2-loss support vector machines. Finally, Section 7 gives conclusions.

All sources used in this paper are available at

`http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

A preliminary version of this work appears in a conference paper (Lin et al., 2007).

2. Trust Region Newton Methods

In this section, we briefly discuss Newton and truncated Newton methods. For large-scale logistic regression, we then propose a trust region Newton method, which is a type of truncated Newton approach.

2.1 Newton and Truncated Newton Methods

To discuss Newton methods, we need the gradient and Hessian of $f(\mathbf{w})$:

$$\nabla f(\mathbf{w}) = \mathbf{w} + C\sum_{i=1}^{l}(\sigma(y_i\mathbf{w}^T\mathbf{x}_i)-1)y_i\mathbf{x}_i, \tag{3}$$

$$\nabla^2 f(\mathbf{w}) = I + CX^T DX, \tag{4}$$

where I is the identity matrix,

$$\sigma(y_i\mathbf{w}^T\mathbf{x}_i) = (1+e^{-y_i\mathbf{w}^T\mathbf{x}_i})^{-1}.$$

D is a diagonal matrix with

$$D_{ii} = \sigma(y_i\mathbf{w}^T\mathbf{x}_i)(1-\sigma(y_i\mathbf{w}^T\mathbf{x}_i)), \text{ and } X = \begin{bmatrix}\mathbf{x}_1^T\\ \vdots \\ \mathbf{x}_l^T\end{bmatrix}$$

is an $l \times n$ matrix. The Hessian matrix $\nabla^2 f(\mathbf{w})$ is positive definite, so (2) is strictly convex. We can further prove the following theorem.

Theorem 1 (2) *attains a unique global optimal solution.*

The proof is in Appendix A.

Since $\nabla^2 f(\mathbf{w}^k)$ is invertible, the simplest Newton method updates $\mathbf{w}$ by the following way

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \mathbf{s}^k, \tag{5}$$

where k is the iteration index and $\mathbf{s}^k$, the Newton direction, is the solution of the following linear system:

$$\nabla^2 f(\mathbf{w}^k)\mathbf{s}^k = -\nabla f(\mathbf{w}^k). \tag{6}$$

However, there are two issues in using this update rule:

1. The sequence $\{\mathbf{w}^k\}$ may not converge to an optimal solution. In fact, even the function value may not be guaranteed to decrease.

2. While we assume that the data matrix X is sparse, $X^T DX$ is much denser. The Hessian matrix is then too large to be stored. Thus, solving the linear system (6) is an issue that needs careful consideration.

Optimization researchers address the first issue by adjusting the length of the Newton direction. Two techniques are often used: line search and trust region.

For the second issue, there are two major types of methods for solving linear systems: direct methods (e.g., Gaussian elimination), and iterative methods (e.g., Jacobi and conjugate gradient). The main operation of certain iterative methods is the product between the Hessian matrix and a vector $\mathbf{s}$:

$$\begin{aligned} & \nabla^2 f(\mathbf{w})\mathbf{s} = (I + CX^T DX)\mathbf{s} \\ = \;& \mathbf{s} + C \cdot X^T (D(X\mathbf{s})). \end{aligned} \tag{7}$$

As we assume sparse X, (7) can be efficiently calculated without storing the Hessian matrix $\nabla^2 f(\mathbf{w}^k)$. Therefore, for large logistic regression, iterative methods are more suitable than direct methods, which require the whole Hessian matrix. Among iterative methods, currently conjugate gradients are the most used ones in Newton methods. The optimization procedure then has two layers of iterations: at each outer iteration an inner conjugate gradient procedure finds the Newton direction. Unfortunately, conjugate gradient methods may suffer from lengthy iterations in certain situations. To save time, one may use only an "approximate" Newton direction in the early stages of the outer iterations. Such a technique is called truncated Newton method (or inexact Newton method).

Komarek and Moore (2005) are among the first to apply truncated Newton methods for logistic regression.[1] They approximately solve (6) by conjugate gradient procedures and use (5) to update $\mathbf{w}^k$. They terminate the conjugate gradient procedure if the relative difference of log likelihoods between two consecutive conjugate gradient iterations is smaller than a threshold. However, they do not provide a convergence proof. In fact, when we tried their code, we found that $\|\nabla f(\mathbf{w}^k)\|$ may not approach zero and hence $\{\mathbf{w}^k\}$ may not converge to an optimum.

Optimization researchers have well addressed the above two issues together. They devise the procedure of outer iterations, and specify stopping conditions for the inner iterations. The overall framework guarantees the convergence to the global minimum. The truncation rule of the inner algorithm is important as one should stop after a sufficiently good direction has been found. A survey of truncated Newton methods is by Nash (2000). Some comparisons between limited memory quasi Newton and truncated Newton are by Nocedal and Nash (1991) and Zou et al. (1993).

2.2 A Trust Region Newton Method

We consider the trust region method (Lin and Moré, 1999), which is a truncated Newton method to deal with general bound-constrained optimization problems (i.e., variables are in certain intervals). We simplify the setting to unconstrained situations, so the algorithm is close to earlier work such as Bouaricha et al. (1997) and Steihaug (1983).

At each iteration of a trust region Newton method for minimizing $f(\mathbf{w})$, we have an iterate $\mathbf{w}^k$, a size Δ_k of the trust region, and a quadratic model

$$q_k(\mathbf{s}) = \nabla f(\mathbf{w}^k)^T \mathbf{s} + \frac{1}{2}\mathbf{s}^T \nabla^2 f(\mathbf{w}^k)\mathbf{s}$$

1. They minimize only the negative log likelihood without the regularization term $\mathbf{w}^T\mathbf{w}/2$.

Algorithm 1 A trust region algorithm for logistic regression

1. Given $\mathbf{w}^0$.
2. For $k = 0, 1, \ldots$ (outer iterations)
   - If $\nabla f(\mathbf{w}^k) = \mathbf{0}$, stop.
   - Find an approximate solution $\mathbf{s}^k$ of the trust region sub-problem
   $$\min_{\mathbf{s}} q_k(\mathbf{s}) \quad \text{subject to } \|\mathbf{s}\| \le \Delta_k. \tag{11}$$
   - Compute ρ_k via (8).
   - Update $\mathbf{w}^k$ to $\mathbf{w}^{k+1}$ according to (9).
   - Obtain Δ_{k+1} according to (10).

as the approximation of the value $f(\mathbf{w}^k + \mathbf{s}) - f(\mathbf{w}^k)$. Next, we find a step $\mathbf{s}^k$ to approximately minimize $q_k(\mathbf{s})$ subject to the constraint $\|\mathbf{s}\| \le \Delta_k$. We then update $\mathbf{w}^k$ and Δ_k by checking the ratio

$$\rho_k = \frac{f(\mathbf{w}^k + \mathbf{s}^k) - f(\mathbf{w}^k)}{q_k(\mathbf{s}^k)} \tag{8}$$

of the actual reduction in the function to the predicted reduction in the quadratic model. The direction is accepted if ρ_k is large enough:

$$\mathbf{w}^{k+1} = \begin{cases} \mathbf{w}^k + \mathbf{s}^k & \text{if } \rho_k > \eta_0, \\ \mathbf{w}^k & \text{if } \rho_k \le \eta_0, \end{cases} \tag{9}$$

where $\eta_0 > 0$ is a pre-specified value.

From Lin and Moré (1999), updating rules for Δ_k depend on positive constants η_1 and η_2 such that $\eta_1 < \eta_2 < 1$, while the rate at which Δ_k is updated relies on positive constants σ_1, σ_2, and σ_3 such that $\sigma_1 < \sigma_2 < 1 < \sigma_3$. The trust region bound Δ_k is updated by the rules

$$\begin{array}{lll} \Delta_{k+1} \in [\sigma_1 \min\{\|\mathbf{s}^k\|, \Delta_k\}, \sigma_2 \Delta_k] & \text{if} & \rho_k \le \eta_1, \\ \Delta_{k+1} \in [\sigma_1 \Delta_k, \sigma_3 \Delta_k] & \text{if} & \rho_k \in (\eta_1, \eta_2), \\ \Delta_{k+1} \in [\Delta_k, \sigma_3 \Delta_k] & \text{if} & \rho_k \ge \eta_2. \end{array} \tag{10}$$

Similar rules are used in most modern trust region methods. A description of our trust region algorithm is given in Algorithm 1. The main difference between our algorithm and those by Steihaug (1983) and Bouaricha et al. (1997) is on the rule (10) for updating Δ_k.

The conjugate gradient method to approximately solve the trust region sub-problem (11) is given in Algorithm 2. The main operation is the Hessian-vector product $\nabla^2 f(\mathbf{w}^k)\mathbf{d}^i$, which is implemented using the idea in Eq. (7). Note that only one Hessian-vector product is needed at each conjugate gradient iteration. Since

$$\mathbf{r}^i = -\nabla f(\mathbf{w}^k) - \nabla^2 f(\mathbf{w}^k)\bar{\mathbf{s}}^i,$$

the stopping condition (12) is the same as

$$\| -\nabla f(\mathbf{w}^k) - \nabla^2 f(\mathbf{w}^k)\bar{\mathbf{s}}^i\| \le \xi_k \|\nabla f(\mathbf{w}^k)\|,$$

Algorithm 2 Conjugate gradient procedure for approximately solving the trust region sub-problem (11)

1. Given $\xi_k < 1, \Delta_k > 0$. Let $\bar{\mathbf{s}}^0 = \mathbf{0}, \mathbf{r}^0 = -\nabla f(\mathbf{w}^k)$, and $\mathbf{d}^0 = \mathbf{r}^0$.
2. For $i = 0, 1, \ldots$ (inner iterations)
   - If
   $$\|\mathbf{r}^i\| \le \xi_k \|\nabla f(\mathbf{w}^k)\|, \tag{12}$$
   then output $\mathbf{s}^k = \bar{\mathbf{s}}^i$ and stop.
   - $\alpha_i = \|\mathbf{r}^i\|^2 / ((\mathbf{d}^i)^T \nabla^2 f(\mathbf{w}^k) \mathbf{d}^i)$.
   - $\bar{\mathbf{s}}^{i+1} = \bar{\mathbf{s}}^i + \alpha_i \mathbf{d}^i$.
   - If $\|\bar{\mathbf{s}}^{i+1}\| \ge \Delta_k$, compute τ such that
   $$\|\bar{\mathbf{s}}^i + \tau \mathbf{d}^i\| = \Delta_k, \tag{13}$$
   then output $\mathbf{s}^k = \bar{\mathbf{s}}^i + \tau \mathbf{d}^i$ and stop.
   - $\mathbf{r}^{i+1} = \mathbf{r}^i - \alpha_i \nabla^2 f(\mathbf{w}^k) \mathbf{d}^i$.
   - $\beta_i = \|\mathbf{r}^{i+1}\|^2 / \|\mathbf{r}^i\|^2$.
   - $\mathbf{d}^{i+1} = \mathbf{r}^{i+1} + \beta_i \mathbf{d}^i$.

which implies that $\bar{\mathbf{s}}^i$ is an approximate solution of the linear system (6). However, Algorithm 2 is different from standard conjugate gradient methods for linear systems as the constraint $\|\mathbf{s}\| \le \Delta$ must be taken care of. It is known that (Steihaug, 1983, Theorem 2.1) with $\bar{\mathbf{s}}^0 = \mathbf{0}$, we have

$$\|\bar{\mathbf{s}}^i\| < \|\bar{\mathbf{s}}^{i+1}\|, \forall i,$$

so in a finite number of conjugate gradient iterations, either (12) is satisfied or $\bar{\mathbf{s}}^{i+1}$ violates the trust region constraint. In the latter situation, (13) finds a point on the trust region boundary as

$$q_k(\bar{\mathbf{s}}^i + \tau \mathbf{d}^i) < q_k(\bar{\mathbf{s}}^i).$$

The whole procedure is a careful design to make sure that the approximate Newton direction is good enough and the trust region method converges.

Next, we discuss convergence properties of the trust region Newton method. Most results can be traced back to Steihaug (1983). However, here we follow Lin and Moré (1999) as our algorithmic settings are closer to it. For the sequence $\{\mathbf{w}^k\}$ to have at least one limit point,[2] since $f(\mathbf{w}^k)$ is decreasing, it suffices to show that the level set $\{\mathbf{w} \mid f(\mathbf{w}) \le f(\mathbf{w}^0)\}$ is closed and bounded. This result has been explained in the proof of Theorem 1. To have this limit point to be the minimum, Theorem 2.1 of Lin and Moré (1999) requires that $\nabla^2 f(\mathbf{w}^k)$ is uniformly bounded. We have this property as $\nabla^2 f(\mathbf{w})$ is continuous in this bounded level set.

Eq. (12) is a relative stopping condition in solving a linear system. The parameter ξ_k effectively controls the efforts associated with the inner iterations. The following theorem summarizes the convergence of Algorithm 1.

2. That is, the sequence $\{\mathbf{w}^k\}$ has at least one convergent sub-sequence.

Theorem 2 *The sequence $\{\mathbf{w}^k\}$ generated by Algorithm 1 globally converges to the unique minimum of* (2)*. If $\xi_k < 1$, then the trust region method Q-linearly converges:*

$$\lim_{k\to\infty} \frac{\|\mathbf{w}^{k+1} - \mathbf{w}^*\|}{\|\mathbf{w}^k - \mathbf{w}^*\|} < 1, \tag{14}$$

where $\mathbf{w}^*$ is the unique optimal solution of (2)*. If*

$$\xi_k \to 0 \text{ as } k \to \infty,$$

then the limit in (14) *becomes zero, so we have Q-superlinear convergence.*

We do not provide a proof here as it follows from Theorem 5.4 of Lin and Moré (1999). Since the Hessian matrix $\nabla^2 f(\mathbf{w})$ is continuously differentiable, $\nabla^2 f(\mathbf{w})$ is Lipschitz continuous around the optimal solution. Hence, as explained by Lin and Moré (1999),[3] if $\xi_k \le \kappa_0 \|\nabla f(\mathbf{w}^k)\|$ for a positive constant κ_0, then at final iterations, our algorithm has quadratic convergence:

$$\lim_{k\to\infty} \frac{\|\mathbf{w}^{k+1} - \mathbf{w}^*\|}{\|\mathbf{w}^k - \mathbf{w}^*\|^2} < 1.$$

Regarding the computational complexity, the cost per iteration is

$$\begin{aligned} & O(\text{nnz}) \text{ for 1 function and 0/1 gradient evaluations} \\ + \quad & O(\text{nnz}) \times \text{ number of conjugate gradient iterations,} \end{aligned} \tag{15}$$

where nnz is the number of nonzero elements in the sparse matrix X. Note that if $\mathbf{w}^k$ is not updated in (9), then the gradient is the same for the next iteration.

3. Related Methods and Implementation Issues

In this section, we discuss a general limited memory quasi Newton implementation (Liu and Nocedal, 1989). Many consider it to be very efficient for training logistic regression. We also discuss implementation issues of the proposed trust region Newton method.

3.1 Limited Memory Quasi Newton Method

We briefly introduce the approach by Liu and Nocedal (1989). Quasi Newton methods use certain techniques to obtain an approximate inverse Hessian H_k and can easily update it to H_{k+1}. One of the most popular updates is BFGS. The approach by Liu and Nocedal (1989) is almost the same as BFGS, but restricts the update to use only m vectors from the previous iterations. The matrix H_k is not formed explicitly and there is an efficient way to compute $H_k \nabla f(\mathbf{w}^k)$. This property is useful for large logistic regression as we cannot afford to store H_k. The procedure is sketched in Algorithm 3.

Regarding the convergence rate, Assumption 7.1 of Liu and Nocedal (1989) requires:

1. $f(\mathbf{w})$ is twice continuously differentiable.
2. The level set $\{\mathbf{w} \mid f(\mathbf{w}) \le f(\mathbf{w}^0)\}$ is bounded.

3. See the explanation given in that paper after the proof of Theorem 5.4.

Algorithm 3 Limited memory BFGS

1. Given $\mathbf{w}^0, H^0$ and a small integer m.
2. For $k = 0, 1, \ldots$
   - If $\nabla f(\mathbf{w}^k) = \mathbf{0}$, stop.
   - Using m vectors from previous iterations to calculate $H_k \nabla f(\mathbf{w}^k)$, where H_k is an approximate inverse Hessian.
   - Search α_k so that
     $$f(\mathbf{w}^k - \alpha H_k \nabla f(\mathbf{w}^k))$$
     satisfies certain sufficient decrease conditions.
   - Update H_k to H_{k+1}.

3. There are positive constants M_1 and M_2 such that
$$M_1 \|\mathbf{s}\|^2 \le \mathbf{s}^T \nabla^2 f(\mathbf{w}) \mathbf{s} \le M_2 \|\mathbf{s}\|^2, \quad \forall \mathbf{s}.$$

The function we are minimizing satisfies the first condition. The second condition follows from our proof of Theorem 1 (see Eq. 29). The third condition follows from choosing

$$M_1 = 1 \text{ and } M_2 = 1 + C\|X^T\|\|X\|.$$

Then Algorithm 3 is R-linearly convergent. That is, there is a constant $c < 1$ such that

$$f(\mathbf{w}^k) - f(\mathbf{w}^*) \le c^k (f(\mathbf{w}^0) - f(\mathbf{w}^*)), \tag{16}$$

where $\mathbf{w}^*$ is the unique optimal solution of (2). Note that (14) implies (16), so Algorithm 1 has a stronger convergence property than Algorithm 3. While truncated Newton methods find an approximate direction, they still use the exact Hessian matrix. In contrast, limited memory quasi Newton methods consider only approximate Hessian matrices, so we can expect that it has slower convergence.

The cost per iteration is

$$\begin{aligned} & O(\text{nnz}) \text{ for function/gradient evaluations in line search} \\ + \quad & O(nm) \text{ for } H_k \nabla f(\mathbf{w}^k) \text{ and updating } H_k \text{ to } H_{k+1}. \end{aligned} \tag{17}$$

As generally $nm <$ nnz, function/gradient evaluations take most computational time. Moreover, compared to (15), the cost per iteration is less than that for our trust region method. However, as we will show in experiments, LBFGS' total training time is longer due to its lengthy iterations.

In this paper, we use $m = 5$, which is the default choice in the LBFGS software (Liu and Nocedal, 1989).

3.2 Implementation Issues of Trust Region Newton Method

We give details of parameters in the proposed Algorithms 1 and 2. All settings are almost the same as the TRON software (Lin and Moré, 1999).

Problem	l	# Positive	# Negative	n	# nonzeros
a9a	32,561	7,841	24,720	123	451,592
real-sim	72,309	22,238	50,071	20,958	3,709,083
news20	19,996	9,999	9,997	1,355,191	9,097,916
yahoo-japan	176,203	15,937	160,266	832,026	23,506,415
rcv1	677,399	355,460	321,939	47,236	49,556,258
yahoo-korea	460,554	145,831	314,723	3,052,939	156,436,656

Table 1: Data statistics: l is the number of instances and n is the number of features. # nonzeros indicates the number of nonzeros among $l \times n$ values.

We set the initial $\Delta_0 = \|\nabla f(\mathbf{w}^0)\|$ and take $\eta_0 = 10^{-4}$ in (9) to update $\mathbf{w}^k$. For changing Δ_k to Δ_{k+1}, we use

$$\eta_1 = 0.25, \eta_2 = 0.75,$$
$$\sigma_1 = 0.25, \sigma_2 = 0.5, \sigma_3 = 4.0.$$

As (10) specifies only the interval in which Δ_{k+1} should lie, there are many possible choices of the update rules. We use the same rules as given by Lin and Moré (1999). In the conjugate gradient procedure, we use $\xi_k = 0.1$ in the stopping condition (12). One may wonder how the above numbers are chosen. These choices are considered appropriate following the research on trust region methods in the past several decades. It is unclear yet if they are the best for logistic regression problems, but certainly we would like to try custom settings first.

4. Experiments

In this section, we compare our approach with a quasi Newton implementation for logistic regression. After describing data sets for experiments, we conduct detailed comparisons and discuss results.

4.1 Data Sets

We consider six data sets from various sources. Table 1 lists the numbers of instances (# positive, # negative), features, and nonzero feature values. Details of data sets are described below.

a9a: This set is compiled by Platt (1998) from the UCI "adult" data set (Asuncion and Newman, 2007). It is available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/a9a`.

real-sim: This set is from the web site `http://people.cs.uchicago.edu/~vikass/datasets/lskm/svml/`. It, originally compiled by Andrew McCallum, includes Usenet articles from four discussion groups, for simulated auto racing, simulated aviation, real autos, and real aviation.

news20: This is a collection of news documents. We use the data processed by Keerthi and DeCoste (2005). They consider binary term frequencies and normalize each instance to unit length. This set is available at
`http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/news20.binary.bz2`.

	TRON		LBFGS
C	CV	Time	Time
0.25	84.69%	1	12
1	84.71%	2	24
4	84.72%	4	47
16	84.71%	7	92

(a) a9a

	TRON		LBFGS
C	CV	Time	Time
0.25	95.85%	4	17
1	96.97%	6	34
4	97.41%	10	52
16	97.51%	14	126

(b) real-sim

	TRON		LBFGS
C	CV	Time	Time
0.25	89.74%	24	78
1	93.36%	38	181
4	95.49%	59	331
16	96.30%	82	614

(c) news20

	TRON		LBFGS
C	CV	Time	Time
0.25	91.91%	28	94
1	92.50%	42	185
4	92.81%	64	326
16	92.86%	113	534

(d) yahoo-japan

	TRON		LBFGS
C	CV	Time	Time
0.25	97.18%	39	106
1	97.56%	62	427
4	97.72%	94	615
16	97.69%	118	821

(e) rcv1

	TRON		LBFGS
C	CV	Time	Time
0.25	81.34%	221	1066
1	84.03%	385	2165
4	85.75%	773	3480
16	86.40%	1888	6329

(f) yahoo-korea

Table 2: The comparison between TRON and LBFGS. Here time (in seconds) is the total training time in the CV procedure. As TRON and LBFGS minimize the same formulation and their CV accuracy values are almost the same, we present only the result of TRON. The number of CV folds is five for small problems, and is two for larger ones (yahoo-japan, rcv1, yahoo-korea). Note that the CV values do not increase using $C > 16$.

yahoo-japan: This set, obtained from Yahoo!, includes documents in hundreds of classes. We consider the class with the largest number of instances as positive and all remaining instances as negative. We use binary term frequencies and normalize each instance to unit length.

rcv1: This set (Lewis et al., 2004) is an archive of manually categorized newswire stories from Reuters Ltd. Each vector is a cosine normalization of a log transformed TF-IDF (term frequency, inverse document frequency) feature vector. The news documents are in a hierarchical structure of classes. We split the data to positive/negative by using the two branches in the first layer of the hierarchy. Data which are multi-labeled (i.e., in both branches) are not considered. The set used here can be found at
`http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/rcv1_test.binary.bz2`.

yahoo-korea: This set, from Yahoo!, includes documents in a hierarchy of classes. We consider the largest branch from the root node (i.e., the branch including the largest number of classes) as positive, and all others as negative.

Clearly, except a9a, all other sets are from document classification. We find that normalizations are usually needed so that the length of each instance is not too large. Otherwise, when the number of features is large, $\mathbf{w}^T\mathbf{x}_i$ may be huge and cause difficulties in solving optimization problems (For good performance also, past experiences show that such normalizations are usually needed). After normalization, we include the bias term using (1).

All data sets are quite balanced. It is known that unbalanced sets usually lead to shorter training time. Therefore, problems used in this article are more challenging in terms of training time.

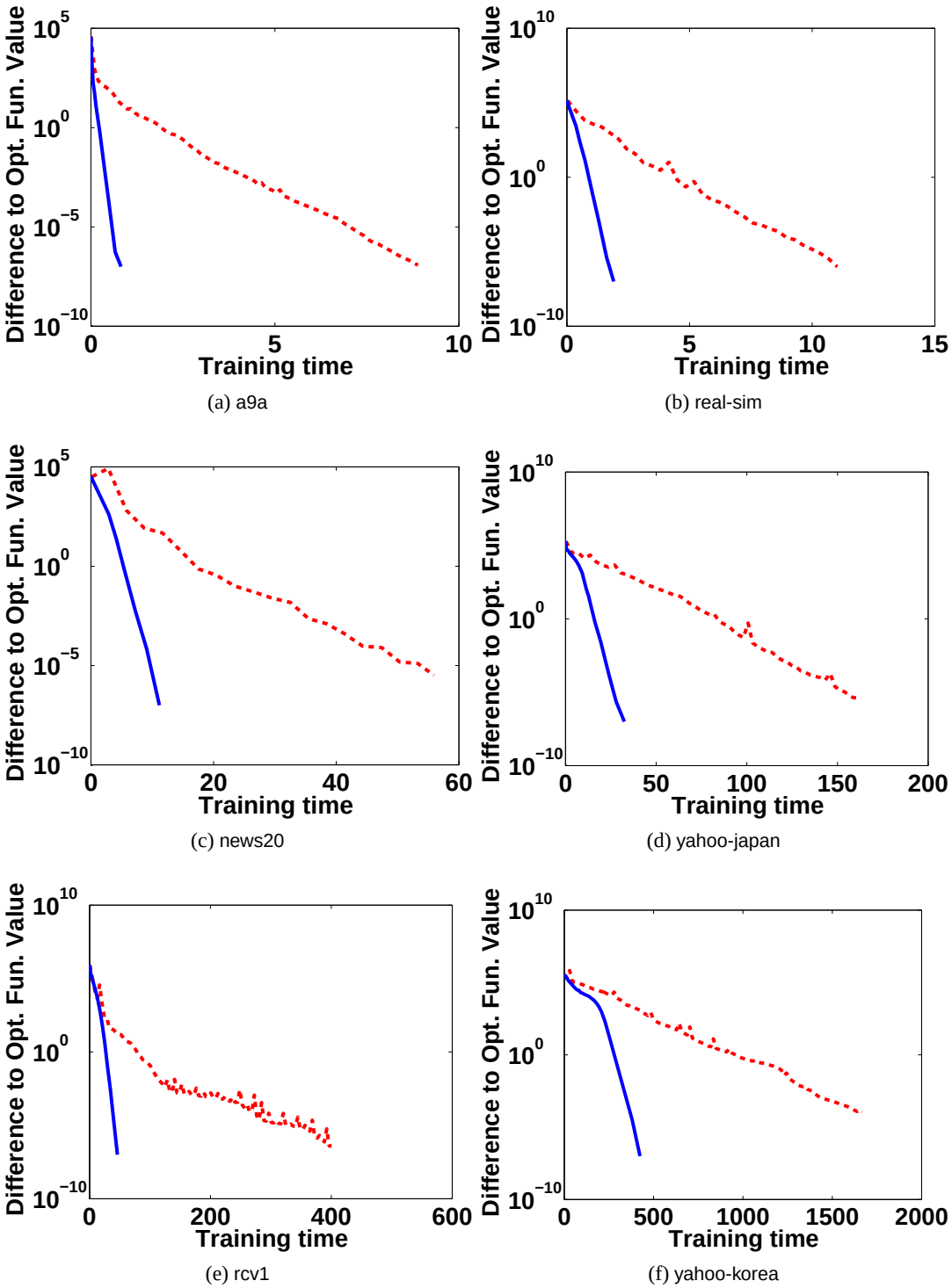


Figure 1: A comparison between TRON (blue solid line) and LBFGS (red dotted line). The y-axis shows the difference to the optimal function value. The x-axis (training time) is in seconds. We use the training set from the first training/validation split of the CV procedure and set $C = 4$.

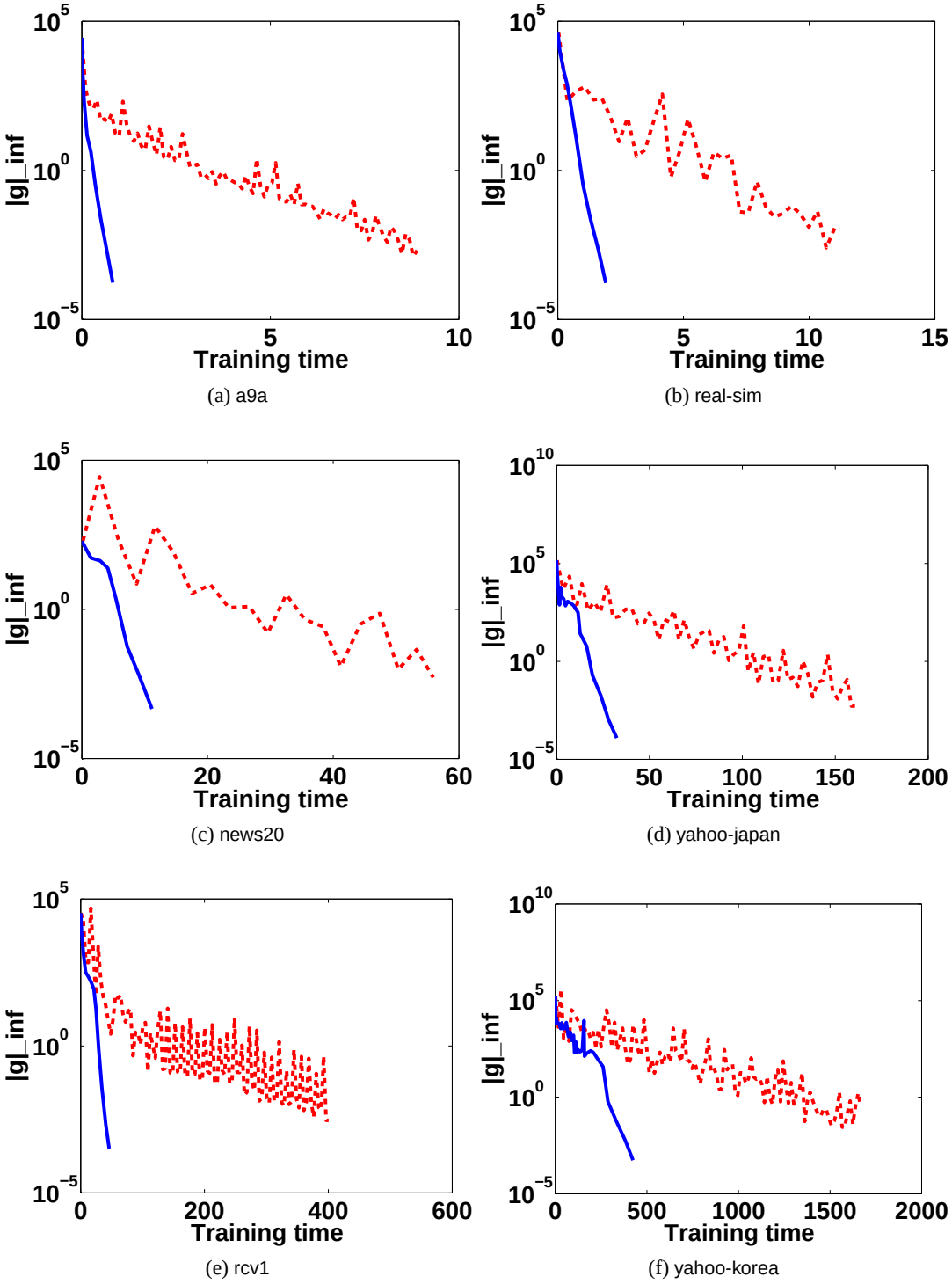


Figure 2: A comparison between TRON (blue solid line) and LBFGS (red dotted line). The y-axis shows $\|\nabla f(\mathbf{w})\|_\infty = \max_j |\nabla f(\mathbf{w})_j|$. The x-axis (training time) is in seconds. We use the training set from the first training/validation split of the CV procedure and set $C = 4$.

4.2 Comparisons

We compare two logistic regression implementations:

- TRON: the trust region Newton method discussed in Section 2.2.
- LBFGS: the limited memory quasi Newton implementation (Liu and Nocedal, 1989). See the discussion in Section 3.1. The source code is available online at

  `http://www.ece.northwestern.edu/~nocedal/lbfgs.html`.

We do not consider the code by Komarek and Moore (2005) because of two reasons. First, we have mentioned its convergence problems in Section 2.1. Second, for sparse data, it handles only problems with 0/1 feature values, but most our data have real-numbered features.

These methods are implemented in high-level languages such as C/C++ or FORTRAN. For easier experiments, we use their Matlab interfaces. Experiments are conducted on an Intel Core2 Quad (2.66GHz) computer with 8 GB RAM. All sources used for this comparison can be found at

`http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

We set the initial $\mathbf{w}^0 = \mathbf{0}$.

We conduct two types of experiments. For the first one, we simulate the practical use of logistic regression by setting a stopping condition and checking the prediction ability. Most unconstrained optimization software use gradient information to terminate the iterative procedure, so we use

$$\|\nabla f(\mathbf{w}^k)\|_\infty \leq 10^{-3} \tag{18}$$

as the stopping condition. We then report cross-validation (CV) accuracy. For the larger sets (yahoo-japan, rcv1, and yahoo-korea), we use two-fold CV. For others, five-fold CV is conducted. We do not consider other measurements such as AUC (Area Under Curve) or F-measure as all problems are rather balanced, and CV accuracy is suitable. Moreover, different values of the regularization parameter C may affect the performance as well as training time. So we try four different C values: 0.25, 1, 4, and 16. Table 2 presents the result of comparisons. We show CV accuracy and the total training time in the CV procedure.

On training time, TRON is better than LBFGS, so truncated Newton methods are effective for training logistic regression. One may wonder if any implementation-specific details cause unfair timing comparisons. Indeed we have made the experimental environments as close as possible. For the methods compared here, we store the sparse matrix X by the same compressed row format. Section 4.3 discusses that different sparse formats may lead to dissimilar computational time. For LBFGS, one main cost is on function/gradient evaluations, which are provided by users. We implement the same code in TRON and LBFGS for function/gradient evaluations. Thus, our timing comparison is quite fair.

For $C \geq 16$, the CV accuracy does not improve. One can clearly see that the training time is higher as C increases. One reason is that the second term of (4) plays a more important role and hence the Hessian matrix is more ill-conditioned. Another reason is that (18) becomes a stricter condition. In (3), the second term of $f(\mathbf{w})$ is proportional to C. Hence, practically one may use a stopping condition relative to C.

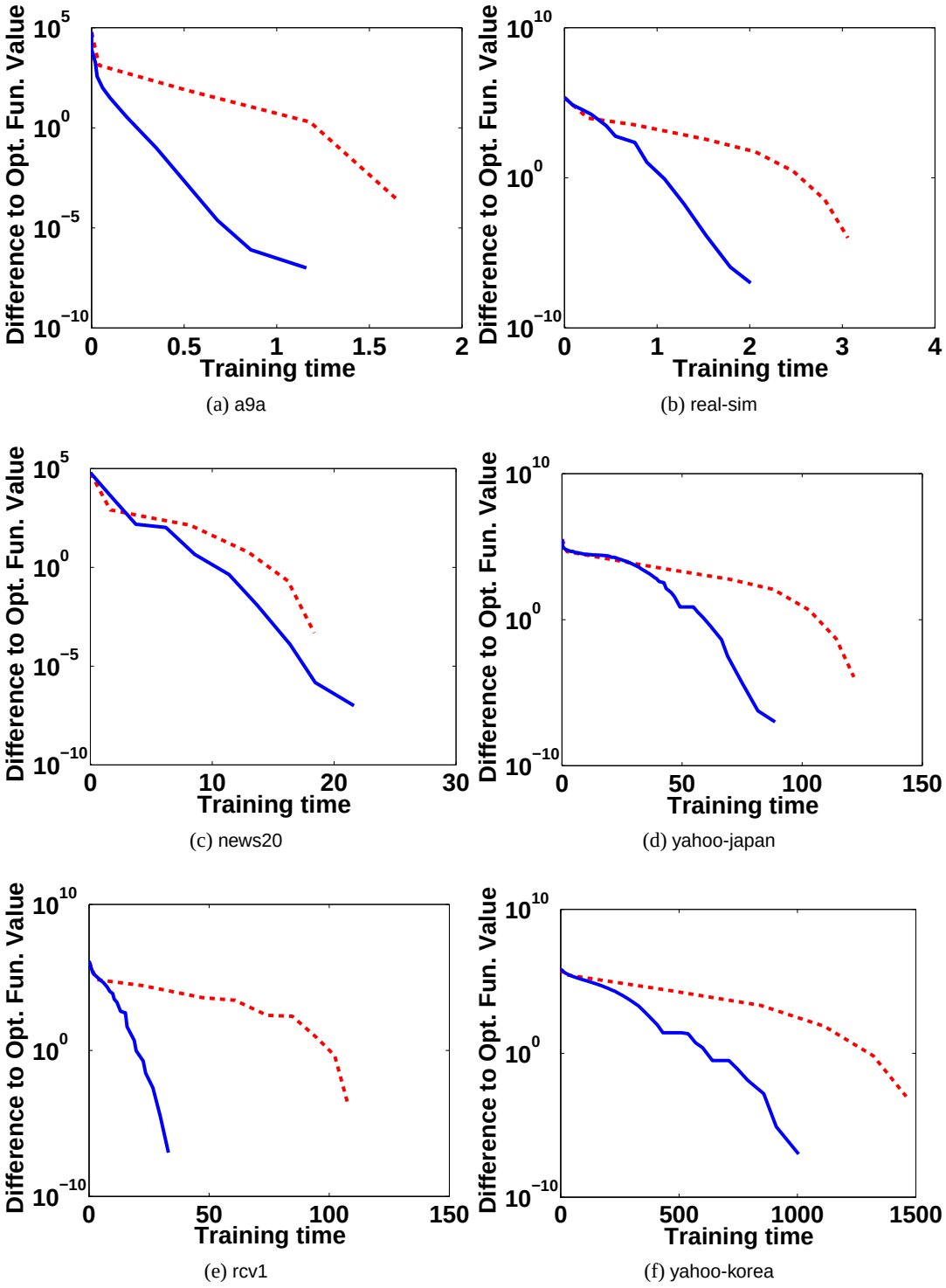


Figure 3: A comparison between TRON (blue solid line) and SVMlin (red dotted line) for L2-SVM. The *y*-axis shows the difference to the optimal function value. The *x*-axis (training time) is in seconds. We use the same training set as in Figure 1 and set $C = 4$.

For the second experiment, we check the convergence speed of both methods. Figure 1 presents the results of time versus the difference to the optimal function value. We use the training set from the first training/validation split of the CV procedure and set $C = 4$. In Figure 2, we check time against $\|\nabla f(\mathbf{w})\|_\infty$. Both figures indicate that TRON more quickly decreases the function as well as the gradient values than LBFGS. This result is consistent with the faster theoretical convergence rate of TRON.

4.3 Row and Column Formats in Storing X

A sparse matrix can be represented by many ways. Two commonly used ones are "compressed column" and "compressed row" formats (Duff et al., 1989). For example, if

$$X = \begin{bmatrix} -10 & 0 & -20 & 0 \\ 30 & 0 & 0 & 10 \end{bmatrix},$$

then its compressed column format is by three arrays:

$$\texttt{X\_val} = [-10, 30, -20, 10], \quad \texttt{X\_rowind} = [1, 2, 1, 2], \quad \texttt{X\_colptr} = [1, 3, 3, 4, 5],$$

where `rowind` means row indices and `colptr` means column pointers.[4] Alternatively, compress row format has

$$\texttt{X\_val} = [-10, -20, 30, 10], \quad \texttt{X\_colind} = [1, 3, 1, 4], \quad \texttt{X\_rowptr} = [1, 3, 5].$$

There are two important properties: First, X's column (row) format is X^T's row (column) format. Second, using the column (row) format for X leads to easy accesses of all values of one column (row). For data classification, the column (row) format thus lets us easily access any particular feature (any particular instance).

The main conjugate gradient operation (7) involves two matrix-vector products—one is with X^T, and the other is with X. In using the column format, there are ways so that for both operations, sequentially X's columns are accessed. Similarly, if using the row format, we only need to access X's rows. Thus, one may think that using the two (row and column) sparse formats does not cause many differences. Table 3 presents a comparison. Surprisingly, for some problems the difference is huge. One possible reason is the different number of nonzero entries per column and per row in X. During the matrix-vector product, as a column (or a row) is used at a time, its elements should be put in the higher level of the computer memory hierarchy. If the number of nonzeros in a column is significantly larger than those in a row, very likely a column cannot be fit into the same memory level as that for a row. We think that this situation occurs for rcv1, for which the number of instances is significantly larger than the number of features.

Of course the practical behavior depends on the computer architectures as well as how nonzero elements are distributed across rows and columns. We do not intend to fully address this issue here, but would like to point out the importance of implementation details in comparing learning algorithms. In Table 2, both methods are implemented using the row format. Without being careful on such details, very easily we may get misleading conclusions.

4. This way of using three arrays is common in FORTRAN programs, which did not support pointers. One can implement this format using pointers, where each pointer associates with values and indices of a row.

Problem	Row	Column
a9a	7	7
real-sim	14	22
news20	82	55
yahoo-japan	113	127
rcv1	118	226
yahoo-korea	1888	2060

Table 3: Total training time (in seconds) in the CV procedure by storing X in compress row and column formats. We use $C = 16$ and $\varepsilon = 0.001$.

5. Preconditioned Conjugate Gradient Methods

To reduce the number of conjugate gradient iterations, in the truncated Newton method one often uses preconditioned conjugate gradient procedures. Instead of solving the Newton linear system (6), we consider a preconditioner which approximately factorizes the Hessian matrix

$$\nabla^2 f(\mathbf{w}^k) \approx PP^T \tag{19}$$

and then solve a new linear system

$$(P^{-1}\nabla^2 f(\mathbf{w}^k)P^{-T})\hat{\mathbf{s}} = -P^{-1}\nabla f(\mathbf{w}^k),$$

where $\hat{\mathbf{s}} = P^T\mathbf{s}$. If the approximate factorization (19) is good enough, $P^{-1}\nabla^2 f(\mathbf{w}^k)P^{-T}$ is close to the identity and less conjugate gradient iterations are needed. However, as we need extra efforts to find P and the cost per conjugate iteration is higher, a smaller number of conjugate gradient iterations may not lead to shorter training time. Thus, finding suitable preconditioners is usually difficult. Popular preconditioners include, for example, diagonal matrix of the Hessian and incomplete Cholesky factorization.

Our situation differs from other unconstrained optimization applications in two aspects. First, lengthy conjugate gradient iterations often occur at final outer steps, but for machine learning applications the algorithm may stop before reaching such a stage. Thus we may not benefit from using preconditioners. Second, preconditioners are more easily obtained by assuming that the whole Hessian matrix $\nabla^2 f(\mathbf{w}^k)$ is available. As we never multiply $X^T DX$ out, $\nabla^2 f(\mathbf{w}^k)$ is not stored and the selection of preconditioners may be more restricted. In this section, we conduct experiments by using the simple diagonal preconditioner

$$P = P^T = \sqrt{\text{Diag}(\nabla^2 f(\mathbf{w}^k))}.$$

Since

$$\nabla^2 f(\mathbf{w}^k)_{ii} = 1 + C\sum_{j=1}^{l} X_{ji}^2 D_{jj},$$

one goes through all X's nonzero elements once for finding diagonal elements. The cost of obtaining the preconditioner is thus no more than that of one conjugate gradient iteration.

The trust region sub-problem needs to be adjusted. Here we follow the derivation of Lin and Moré (1999) by considering a scaled version

$$\min_{\mathbf{s}} \; q_k(\mathbf{s}) \quad \text{subject to } \|P^T\mathbf{s}\| \le \Delta_k. \tag{20}$$

Algorithm 4 Preconditioned conjugate gradient procedure for approximately solving the trust region sub-problem (21)

1. Given $\xi_k < 1, \Delta_k > 0$. Let $\hat{\mathbf{s}}^0 = \mathbf{0}, \mathbf{r}^0 = -\hat{\mathbf{g}}$, and $\mathbf{d}^0 = \mathbf{r}^0$.
2. For $i = 0, 1, \ldots$ (inner iterations)
   - If
   $$\|\mathbf{r}^i\| \le \xi_k \|\hat{\mathbf{g}}\|,$$
   then output $\mathbf{s}^k = P^{-T}\hat{\mathbf{s}}^i$ and stop.
   - $\alpha_i = \|\mathbf{r}^i\|^2 / ((\mathbf{d}^i)^T \hat{H}\mathbf{d}^i)$.
   - $\hat{\mathbf{s}}^{i+1} = \hat{\mathbf{s}}^i + \alpha_i \mathbf{d}^i$.
   - If $\|\hat{\mathbf{s}}^{i+1}\| \ge \Delta_k$, compute τ such that
   $$\|\hat{\mathbf{s}}^i + \tau \mathbf{d}^i\| = \Delta_k,$$
   then output $\mathbf{s}^k = P^{-T}(\hat{\mathbf{s}}^i + \tau \mathbf{d}^i)$ and stop.
   - $\mathbf{r}^{i+1} = \mathbf{r}^i - \alpha_i \hat{H}\mathbf{d}^i$.
   - $\beta_i = \|\mathbf{r}^{i+1}\|^2 / \|\mathbf{r}^i\|^2$.
   - $\mathbf{d}^{i+1} = \mathbf{r}^{i+1} + \beta_i \mathbf{d}^i$.

With $\hat{\mathbf{s}} = P^T \mathbf{s}$, we transform (20) to

$$\min_{\hat{\mathbf{s}}} \ \hat{q}_k(\hat{\mathbf{s}}) \quad \text{subject to } \|\hat{\mathbf{s}}\| \le \Delta_k, \tag{21}$$

where

$$\hat{q}_k(\hat{\mathbf{s}}) = \hat{\mathbf{g}}^T \hat{\mathbf{s}} + \frac{1}{2}\hat{\mathbf{s}}^T \hat{H}\hat{\mathbf{s}},$$

and

$$\hat{\mathbf{g}} = P^{-1}\nabla f(\mathbf{w}^k), \qquad \hat{H} = P^{-1}\nabla^2 f(\mathbf{w}^k)P^{-T}.$$

Eq. (21) is in the same form as (11), the sub-problem without using preconditioners, so the procedure to approximately solve (21) is almost the same as Algorithm 2. We give details in Algorithm 4. Note that in practical implementations we calculate $\hat{H}\mathbf{d}^i$ by a way similar to (7)

$$P^{-1}(P^{-T}\mathbf{d}^i + C(X^T(D(X(P^{-T}\mathbf{d}^i))))).$$

In Table 4, we present the average number of conjugate gradient iterations per fold in the CV procedure. The approach of using diagonal preconditioning reduces the number of iterations for only two problems. The number is increased for all other data sets. This experiment indicates the difficulty of doing preconditioning. Identifying effective preconditioners is thus a challenging future research issue.

Problem	CG	PCG
a9a	567	263
real-sim	104	160
news20	71	155
citeseer	113	115
yahoo-japan	278	326
rcv1	225	280
yahoo-korea	779	736

Table 4: Average number of conjugate gradient iterations per fold in the CV procedure. CG: without preconditioning. PCG: using diagonal preconditioning. We use $C = 16$ and the stopping condition $\|\nabla f(\mathbf{w}^k)\|_\infty \leq 0.001$.

6. Trust Region Method for L2-SVM

The second term in (2) can be considered as a loss function, so regularized logistic regression is related to other learning approaches such as Support Vector Machines (SVM) (Boser et al., 1992). L1-SVM solves the following optimization problem:

$$\min_{\mathbf{w}} f_1(\mathbf{w}) \equiv \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l} \max\left(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right),$$

while L2-SVM solves

$$\min_{\mathbf{w}} f_2(\mathbf{w}) \equiv \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l} \left(\max(0, 1 - y_i\mathbf{w}^T\mathbf{x}_i)\right)^2. \tag{22}$$

SVM is often used with a nonlinear kernel, where data $\mathbf{x}_i$ are mapped to a high dimensional space. However, for document classification, past experiments show that with/without nonlinear mapping gives similar performances. For the case of no nonlinear mapping, we have the possibility of directly solving bigger optimization problems. We refer to such cases as *linear* SVM, and considerable efforts have been made on its fast training (e.g., Kao et al., 2004; Keerthi and DeCoste, 2005; Joachims, 2006; Shalev-Shwartz et al., 2007; Smola et al., 2008). L1-SVM is not differentiable, so our method cannot be applied. For L2-SVM, the training objection function (22) is differentiable but not twice differentiable (Mangasarian, 2002). In this section, we extend our trust region method for L2-SVM. We then compare it with an earlier Newton method for L2-SVM (Keerthi and DeCoste, 2005).

6.1 Trust Region Method

Let $f_2(\mathbf{w})$ be the L2-SVM function. It is strictly convex, so a proof similar to Theorem 1 shows that a unique global minimum exists. From Mangasarian (2002), $f_2(\mathbf{w})$ is continuously differentiable with the gradient

$$\nabla f_2(\mathbf{w}) = (I + 2CX_{I,:}^T X_{I,:})\mathbf{w} - 2CX_{I,:}^T \mathbf{y}_I,$$

where

$$I = \{i \mid 1 - y_i\mathbf{w}^T\mathbf{x}_i > 0\} \tag{23}$$

is an index set depending on $\mathbf{w}$ and $X_{I,:}$ includes X's rows corresponding to the set I. Unfortunately, L2-SVM is not twice differentiable, so one cannot use Newton directions. However, as shown by Mangasarian (2002), this function is almost twice differentiable. The gradient $\nabla f_2(\mathbf{w})$ is Lipschitz continuous, so one can define the *generalized* Hessian matrix

$$B(\mathbf{w}) = I + 2CX^T DX,$$

where

$$D_{ii} = \begin{cases} 1 & \text{if } 1 - y_i \mathbf{w}^T \mathbf{x}_i > 0, \\ \text{any element in } [0,1] & \text{if } 1 - y_i \mathbf{w}^T \mathbf{x}_i = 0, \\ 0 & \text{if } 1 - y_i \mathbf{w}^T \mathbf{x}_i < 0. \end{cases}$$

Then the trust region method (Lin and Moré, 1999) can be applied by replacing $\nabla^2 f(\mathbf{w})$ in Section 2.2 with $B(\mathbf{w})$. In other words, we use the generalized Hessian matrix $B(\mathbf{w})$ to obtain Newton-like directions. As $B(\mathbf{w})$ is uniformly bounded:

$$1 \le \|B(\mathbf{w})\| \le 1 + 2C\|X^T\|\|X\|, \quad \forall \mathbf{w},$$

Theorem 2.1 of Lin and Moré (1999) implies the global convergence. However, we cannot apply Theorem 5.4 of Lin and Moré (1999) to have quadratic convergence. This result requires the twice continuous differentiability.

For experiments here, we set $D_{ii} = 0$ if $1 - y_i \mathbf{w}^T \mathbf{x}_i = 0$. The Hessian-vector product in the conjugate gradient procedure is then

$$B(\mathbf{w})\mathbf{s} = \mathbf{s} + 2C \cdot X_{I,:}^T (D_{I,I}(X_{I,:}\mathbf{s})). \tag{24}$$

6.2 Modified Newton Method for L2-SVM

The method by Keerthi and DeCoste (2005) is currently one of the most efficient methods to train large-scale linear L2-SVM. Its key idea is that for any given index set $I \subset \{1, \ldots, l\}$, if the optimal solution $\mathbf{w}^*$ of the following problem

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i \in I}(1 - y_i \mathbf{w}^T \mathbf{x}_i)^2 \tag{25}$$

satisfies

$$1 - y_i (\mathbf{w}^*)^T \mathbf{x}_i \begin{cases} > 0 & \text{if } i \in I, \\ \le 0 & \text{if } i \notin I, \end{cases}$$

then $\mathbf{w}^*$ is an optimal solution of the L2-SVM problem (22). Once I is fixed, (25) is a simple regularized least square problem and can be solved by the following linear system:

$$(I + 2CX_{I,:}^T X_{I,:})\mathbf{w} = 2CX_{I,:}^T \mathbf{y}_I. \tag{26}$$

One then guesses this set I by (23) and solves (26). The matrix in (26) is a generalized Hessian at $\mathbf{w}$, so (26) intends to obtain a Newton-like direction. Keerthi and DeCoste (2005) use conjugate gradient methods to solve (26), and the procedure is described in Algorithm 5. They prove that Algorithm 5 converges to the optimal solution of (22) in a finite number of iterations. This convergence result assumes that at each iteration, (26) is exactly solved. However, they use a relative stopping condition in practical implementations, so the convergence remains an issue. In contrast, the convergence of our trust region method holds when the conjugate gradient procedure only approximately minimizes the trust-region sub-problem.

Algorithm 5 Modified Newton Method for L2-SVM

1. Given $\mathbf{w}^0$.
2. For $k = 0, 1, \ldots$
   - If $\nabla f(\mathbf{w}^k) = \mathbf{0}$, stop.
   - Set up (26) using
     $$I_k = \{i \mid 1 - y_i (\mathbf{w}^k)^T \mathbf{x}_i > 0\}.$$
     Solve (26) by the conjugate gradient procedure and obtain $\bar{\mathbf{w}}^k$.
   - Let $\mathbf{s}^k = \bar{\mathbf{w}}^k - \mathbf{w}^k$.
     Find
     $$\alpha_k = \arg\min_{\alpha \geq 0} f(\mathbf{w}^k + \alpha \mathbf{s}^k),$$
     and set $\mathbf{w}^{k+1} = \mathbf{w}^k + \alpha_k \mathbf{s}^k$.

6.3 Comparisons

We compare our proposed trust region implementation (TRON) in Section 6.1 with SVMlin

`http://people.cs.uchicago.edu/~vikass/svmlin.html`,

an implementation of the method by Keerthi and DeCoste (2005). To solve (26), SVMlin considers a relative stopping condition for the conjugate gradient procedure. Following their convergence result, we modify SVMlin to quite accurately solve the linear system (26): Recall in Algorithm 5 that we sequentially obtain the following items:

$$\mathbf{w}^k \rightarrow I_k \rightarrow \bar{\mathbf{w}}^k.$$

We then use

$$\|(I + 2CX^T_{I_k,:} X_{I_k,:})\bar{\mathbf{w}}^k - 2CX^T_{I_k,:} \mathbf{y}_{I_k}\|_\infty \leq 10^{-3}$$

as the stopping condition of the conjugate gradient procedure in SVMlin.

Figure 3 presents the result of time versus the difference to the optimal function value. Both approaches spend most of their time on the operation (24) in the conjugate gradient procedure. Clearly, TRON more quickly reduces the function value. SVMlin is slower because it accurately solves (26) at early iterations. Hence, many conjugate gradient iterations are wasted. In contrast, trust region methods are effective on using only approximate directions in the early stage of the procedure.

7. Discussion and Conclusions

As logistic regression is a special case of maximum entropy models and conditional random fields, it is possible to extend the proposed approach for them. The main challenge is to derive the Hessian matrix and efficiently calculate the Hessian-vector product. This topic deserves a thorough investigation in the future.

One may use a different regularized term for logistic regression. For example, the two-norm $\|\mathbf{w}\|^2/2$ could be replaced by a one-norm term $\|\mathbf{w}\|_1$. Then (2) becomes

$$\min_{\mathbf{w}} \quad \|\mathbf{w}\|_1 + C\sum_{i=1}^{l}\log(1+e^{-y_i\mathbf{w}^T\mathbf{x}_i}). \tag{27}$$

This formula has been used for some applications. See (Balakrishnan and Madigan, 2005) and Koh et al. (2007) and references therein. Unfortunately, (27) is not differentiable on $\mathbf{w}$. We can transform it to a twice-differentiable bound-constrained problem by using $\mathbf{w} \equiv \mathbf{w}^+ - \mathbf{w}^-$:

$$\begin{aligned} \min_{\mathbf{w}^+,\mathbf{w}^-} \quad & \sum_{j=1}^{n} w_j^+ + \sum_{j=1}^{n} w_j^- + C\sum_{i=1}^{l}\log(1+e^{-y_i(\mathbf{w}^+-\mathbf{w}^-)^T\mathbf{x}_i}) \\ \text{subject to} \quad & w_j^+ \geq 0, w_j^- \geq 0, \quad j=1,\ldots,n. \end{aligned} \tag{28}$$

As the truncated Newton method by Lin and Moré (1999) exactly targets at such bound-constrained problems, we can thus extend the proposed approach for (28). A comparison to investigate if our method is better than existing ones is an interesting direction for future work.

In summary, we have shown that a trust region Newton method is effective for training large-scale logistic regression problems as well as L2-SVM. The method has nice optimization properties following past developments for large-scale unconstrained optimization. It is interesting that we do not need many special settings for logistic regression; a rather direct use of modern trust region techniques already yields excellent performances. From this situation, we feel that many useful optimization techniques have not been fully exploited for machine learning applications.

Acknowledgments

The authors thank Jianxiong Dong of Yahoo! for helping to provide the data sets yahoo-japan and yahoo-korea. Part of this work was done when the first and the second authors visited Yahoo! Research. The first and the second authors were partially supported by grants from the National Science Council of Taiwan.

Appendix A. Proof of Theorem 1

Since $f(\mathbf{w})$ is strictly convex, a minimum attained is unique and global. The remaining issue is to check if a minimum exists (as strictly convex functions like e^x do not attain a minimum). It suffices to prove that the level set is bounded:

$$\{\mathbf{w} \mid f(\mathbf{w}) \leq f(\mathbf{w}^0)\}, \tag{29}$$

where $\mathbf{w}^0$ is any vector. If this property is wrong, there is a sequence $\{\mathbf{w}^k\}$ in the set (29) satisfying $\|\mathbf{w}^k\| \to \infty$. However,

$$f(\mathbf{w}^k) \geq \frac{1}{2}\|\mathbf{w}^k\|^2 \to \infty$$

contradicts the fact that $f(\mathbf{w}^k) \leq f(\mathbf{w}^0), \forall k$.

References

Arthur Asuncion and David J. Newman. UCI machine learning repository, 2007. URL http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html.

Suhrid Balakrishnan and David Madigan. Algorithms for sparse linear classifiers in the massive data setting. 2005. URL http://www.stat.rutgers.edu/~madigan/PAPERS/sm.pdf.

Steven Benson and Jorge J. Moré. A limited memory variable metric method for bound constrained minimization. Preprint MCS-P909-0901, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 2001.

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.

Ali Bouaricha, Jorge J. Moré, and Zhijun Wu. Newton's method for large-scale optimization. Preprint MCS-P635-0197, Argonne National Laboratory, Argonne, Illinois, 1997.

John N. Darroch and Douglas Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

Iain S. Duff, Roger G. Grimes, and John G. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15:1–14, 1989.

Joshua Goodman. Sequential conditional generalized iterative scaling. In *ACL*, pages 9–16, 2002.

Rong Jin, Rong Yan, Jian Zhang, and Alex G. Hauptmann. A faster iterative scaling algorithm for conditional exponential model. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, 2003.

Thorsten Joachims. Training linear SVMs in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2006.

Wei-Chun Kao, Kai-Min Chung, Chia-Liang Sun, and Chih-Jen Lin. Decomposition methods for linear support vector machines. *Neural Computation*, 16(8):1689–1704, 2004. URL http://www.csie.ntu.edu.tw/~cjlin/papers/linear.pdf.

S. Sathiya Keerthi and Dennis DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

Kwangmoo Koh, Seung-Jean Kim, and Stephen Boyd. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007. URL http://www.stanford.edu/~boyd/l1_logistic_reg.html.

Paul Komarek and Andrew W. Moore. Making logistic regression a core data mining tool: A practical investigation of accuracy, speed, and simplicity. Technical Report TR-05-27, Robotics Institute, Carnegie Mellon University, 2005.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

Chih-Jen Lin and Jorge J. Moré. Newton's method for large-scale bound constrained problems. *SIAM Journal on Optimization*, 9:1100–1127, 1999.

Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust region Newton method for large-scale logistic regression. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007. Software available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th conference on Natural language learning*, pages 1–7. Association for Computational Linguistics, 2002.

Olvi L. Mangasarian. A finite Newton method for classification. *Optimization Methods and Software*, 17(5):913–929, 2002.

Thomas P. Minka. A comparison of numerical optimizers for logistic regression, 2003. URL `http://research.microsoft.com/~minka/papers/logreg/`.

Stephen G. Nash. A survey of truncated-Newton methods. *Journal of Computational and Applied Mathematics*, 124(1-2):45–59, 2000.

Jorge Nocedal and Stephen G. Nash. A numerical study of the limited memory BFGS method and the truncated-newton method for large scale optimization. *SIAM Journal on Optimization*, 1(3): 358–372, 1991.

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 C. Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007.

Alex J. Smola, S V N Vishwanathan, and Quoc Le. Bundle methods for machine learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

Trond Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20:626–637, 1983.

Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006.

Xiaolei Zou, I. Michael Navon, M. Berger, P. K. H. Phua, Tamar Schlick, and F.X. Le Dimet. Numerical experience with limited-memory quasi-Newton and truncated Newton methods. *SIAM Journal on Optimization*, 3(3):582–608, 1993.

Journal of Machine Learning Research 9 (2008) 651-682 Submitted 1/06; Revised 10/07; Published 4/08

Graphical Models for Structured Classification, with an Application to Interpreting Images of Protein Subcellular Location Patterns

Shann-Ching Chen SHANNCC@ANDREW.CMU.EDU
Department of Biomedical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Geoffrey J. Gordon GGORDON@CS.CMU.EDU
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Robert F. Murphy MURPHY@CMU.EDU
Departments of Biological Sciences, Biomedical Engineering, and Machine Learning
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Editor: Nir Friedman

Abstract

In structured classification problems, there is a direct conflict between expressive models and efficient inference: while graphical models such as Markov random fields or factor graphs can represent arbitrary dependences among instance labels, the cost of inference via belief propagation in these models grows rapidly as the graph structure becomes more complicated. One important source of complexity in belief propagation is the need to marginalize large factors to compute messages. This operation takes time exponential in the number of variables in the factor, and can limit the expressiveness of the models we can use. In this paper, we study a new class of potential functions, which we call decomposable k-way potentials, and provide efficient algorithms for computing messages from these potentials during belief propagation. We believe these new potentials provide a good balance between expressive power and efficient inference in practical structured classification problems. We discuss three instances of decomposable potentials: the associative Markov network potential, the nested junction tree, and a new type of potential which we call the voting potential. We use these potentials to classify images of protein subcellular location patterns in groups of cells. Classifying subcellular location patterns can help us answer many important questions in computational biology, including questions about how various treatments affect the synthesis and behavior of proteins and networks of proteins within a cell. Our new representation and algorithm lead to substantial improvements in both inference speed and classification accuracy.

Keywords: factor graphs, approximate inference algorithms, structured classification, protein subcellular location patterns, location proteomics

1. Introduction

In standard supervised classification problems, the label of each test instance is independent of the labels of all other instances. In some problems, however, we may receive multiple test instances at a time, along with side information about dependences among the labels of these instances. For

example, if each instance is a handwritten character, the side information might be that the string of characters forms a common English word; or, if each instance is a microscope image of a cell with a certain protein tagged, the side information might be that several cells share the same tagged protein. To solve such a *structured classification problem* in practice, we need both an expressive way to represent our beliefs about the structure, as well as an efficient probabilistic inference algorithm for classifying new groups of instances.

Unfortunately, the goal of having an expressive language is in direct conflict with the goal of having an efficient inference algorithm: while Markov random fields or factor graphs can represent arbitrary dependences among instances, inference rapidly becomes intractable as the graph structure becomes more complicated (see, e.g., Koller and Friedman, 2007). Simple graphs such as pairwise links arranged in chains or trees lead to efficient inference, but these structures may not allow us to express our beliefs accurately or completely. On the other hand, if we try to couple large groups of labels (either directly, by specifying a factor that links to a large number of labels, or indirectly, by using a graph with large loops), the cost of inference grows exponentially.

To speed up inference, we can move to approximate algorithms such as loopy belief propagation (see, e.g., Koller and Friedman, 2007). Loopy belief propagation handles large loops efficiently, but it does nothing to speed up the task of working with single large factors. In fact, in practical problems, the operation of marginalizing a large factor can easily become the main bottleneck for inference, preventing us from using more-expressive models.

Therefore, in this paper, we study a new class of potential functions, which we call decomposable k-way potentials. Computing messages for these potentials is much more efficient than for general potentials, even though the new potentials can express distributions that cannot be represented by groups of smaller potentials. Accordingly, we believe these new potentials provide a better balance between expressive power and efficient inference than was previously available.

We discuss three instances of decomposable potentials: the associative Markov network potential, the nested junction tree, and a new type of potential which we call the voting potential. We use these potentials to classify images of protein subcellular location patterns in groups of cells. Classifying protein subcellular location patterns is important as a step in solving many practical computational biology problems, particularly in the area of systems biology: for example, it can help in designing high-throughput screening systems for drug discovery, or in conducting experiments to determine the effect of various treatments on the synthesis and behavior of proteins and networks of proteins within a cell. Our new representation and algorithm lead to substantial improvements in both inference speed and classification accuracy.

Preliminary versions of portions of this work have been presented previously (Chen and Murphy, 2006; Chen et al., 2006a,b). These papers describe applications of decomposable potentials and the corresponding fast inference algorithms for segmenting and classifying images of protein subcellular location patterns. But, none of these papers describe the idea of decomposable potentials of Section 4 or the nested inference algorithm of Section 5 in full generality. They also do not cover some of the instances of decomposable potentials described in Section 6; and, the experiments of Sections 8–9 have not been reported previously.

2. Factor Graphs

The factor graph representation of a probability distribution (Kschischang et al., 2001) describes the relationships among a set of variables x_i using local factors or potentials φ_j. Each factor depends

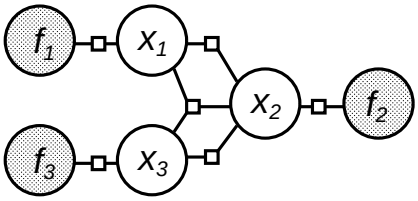


Figure 1: A probability distribution represented as a factor graph. Small squares denote potential functions; for example, this factor graph contains a potential which connects the variables x_1, x_2, and x_3, and another which connects x_1 and f_1.

on only a subset of the variables, and the overall probability distribution is the product of the local factors, together with a normalizing constant Z:

$$P(x) = \frac{1}{Z} \prod_{\text{factors } j} \varphi_j(x_{V(j)}).$$

Here $V(j)$ is the set of variables that are arguments to factor j; for example, if φ_j depends on x_1, x_3, and x_4, then $V(j) = \{1,3,4\}$ and $x_{V(j)} = (x_1, x_3, x_4)$.

Each variable x_i or factor φ_j corresponds to a node in the factor graph. Fig. 1 shows an example: the large nodes represent variables, with shaded circles for observed variables and open circles for unobserved ones. The small square nodes represent factors, and there is an edge between a variable x_i and a factor φ_j if and only if φ_j depends on x_i, that is, when $i \in V(j)$. (By convention the graph only shows factors with two or more arguments. Factors with just a single argument are not explicitly represented, but are implicitly allowed to be present at each variable node.)

The inference task in a factor graph is to combine the evidence from all of the factors to compute properties of the distribution over x represented by the graph. Naively, we can do inference by enumerating all possible values of x, multiplying together all of the factors, and summing to compute the normalizing constant. Unfortunately, the total number of terms in the sum is exponential in the number of random variables in the graph. So, usually, a better way to perform inference is via a message-passing algorithm called belief propagation (BP). Here we briefly review the basics of BP in factor graphs; for more details, see Kschischang et al. (2001).

The basic BP algorithm works on a factor graph which is tree-shaped (i.e., has no cycles). It sends messages from every variable to each of its neighboring factors, and from every factor to each of its neighboring variables. Messages to or from a variable x_i will be vectors whose length is equal to the number of values that x_i can take on.

For a variable x_i with neighboring factors $\varphi_1, \varphi_2, \ldots, \varphi_k$, suppose that x has received messages $m_{j \to i}(x_i)$ from its neighbors φ_j for $j \in \{1 \ldots (k-1)\}$. (To simplify notation, we have numbered x_i's neighbors consecutively starting from 1. This is not a loss of generality since we can always temporarily permute our factor indices to make it so.) Suppose also that x_i has local evidence represented by the one-argument factor $\varphi_i^{\text{loc}}(x_i)$. Then we can compute x_i's message to φ_k as

$$m_{i \to k}(x_i) = \varphi_i^{\text{loc}}(x_i) \prod_{j=1}^{k-1} m_{j \to i}(x_i). \tag{1}$$

That is, we take the componentwise product of all of the messages from factors $1 \ldots k-1$, multiply in the local evidence, and send the result to φ_k. The normalization of the message is arbitrary, so for convenience or numerical precision we may multiply each component of the message by an arbitrary constant.

Similarly, suppose that a factor φ_j has neighbors $x_1, x_2, \ldots, x_k$ (again numbered consecutively without loss of generality) and has received messages $m_{i \to j}(x_i)$ for $i \in \{1 \ldots (k-1)\}$. Then we can compute φ_j's message to x_k as

$$m_{j \to k}(x_k) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_{k-1}} \varphi_j(x_1, \ldots, x_k) \prod_{i=1}^{k-1} m_{i \to j}(x_i). \tag{2}$$

Unlike Equation 1, in Equation 2 we must marginalize out variables other than x_k by summing over their possible values.

BP works by picking an arbitrary node of the graph as root and then making two passes over the tree: first it passes messages inward from the leaves to the root, and then outward from the root to the leaves. When the message passing finishes, the posterior marginal probability of any random variable x_i is just the componentwise product of its local potential and all of its incoming messages:

$$P(x_i) = \varphi^{\text{loc}}(x_i) \prod_{\{j | i \in V(j)\}} m_{j \to i}(x_i). \tag{3}$$

2.1 Working with General Graphs

The above discussion assumed that our factor graph was tree-shaped. For graphs with loops, we have two alternatives: first, we can collapse groups of variable nodes together into combined nodes, which can turn our graph into a tree and allow us to run BP as above. Second, we can run an approximate inference algorithm that doesn't require a tree-shaped graph. We can also combine these alternatives if desired, grouping variable nodes to reduce the number of loops in the graph so that our approximate inference becomes more accurate.

When we combine a set of variable nodes, the new node represents all possible settings of all of the original nodes. E.g., if we collapse a variable x_1 that has settings T, F with a variable x_2 that has settings A, B, C, then the combined variable x_{12} has settings TA, TB, TC, FA, FB, FC. When we collapse a set of variables, we need to alter the neighboring factor nodes: any factor adjacent to any of the original nodes becomes a neighbor of the new combined node, and its list of arguments is extended if necessary to include all variables in the collapsed set.

The advantage of collapsing variable nodes is that it can allow us to simplify the structure of our graph: if in the collapsed graph there are two factor nodes with the same set of arguments, then we can combine them by multiplying their potentials elementwise. For example, Fig. 2 shows the result of collapsing x_1 and x_2 in the factor graph of Fig. 1. The potentials φ_{23} and φ_{123} from the original graph have the same set of neighbors in the new graph, and so can be combined into one factor node. Similarly, the local potentials φ_1^{loc} and φ_2^{loc} can be combined with the factor φ_{12} to form a new local potential at the collapsed node x_{12}. Notice that the new factor graph is tree-shaped, even though the original one had loops.

When removing loops from a factor graph, we may wish to include each original variable in more than one combined node. We are free to do so as long as we adjust our potentials to enforce the constraint that all copies of the variable must agree on its value. That is, when any two copies

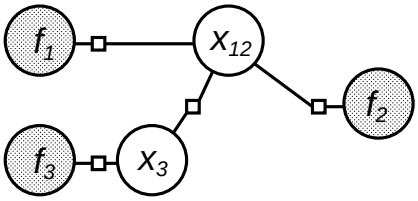


Figure 2: Another factor graph that can represent the same probability distribution as the graph of Fig. 1. The new graph was derived by collapsing together the nodes for variables x_1 and x_2 from Fig. 1.

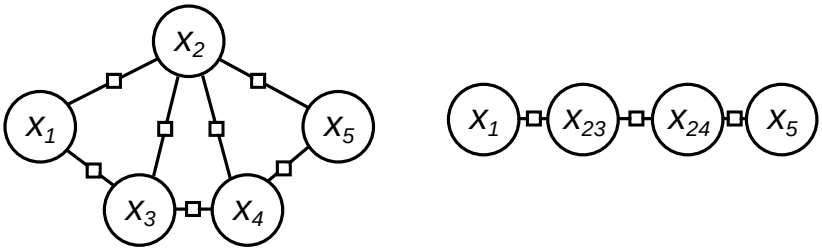


Figure 3: A factor graph with multiple loops (left) and a tree derived from the factor graph by collapsing pairs of nodes (right).

disagree, at least one potential must be zero. The cost of storing or working with such a hard-constraint potential depends only on the number of *distinct* variables in its argument list.

For example, in the graph of Fig. 3, we could form a tree by collapsing x_2, x_3, and x_4 into a single node. The resulting graph would have variables x_1, x_{234}, and x_5, and factors φ_{1234} and φ_{2345}. But, we can form a tree with smaller factors if we group x_2 with x_3 and also separately with x_4: the resulting graph has variables x_1, x_{23}, x_{24}, and x_5, and factors φ_{123}, φ_{234}, and φ_{245}. The potential φ_{234} encodes the constraint that the settings of x_{23} and x_{24} must agree on the assignment to x_2.

A factor graph that has been reduced to a tree is equivalent to a *junction tree*. A junction tree is a connected acyclic graph whose nodes are labeled with sets of variables in a way that satisfies the *running intersection property*: if a variable is present at two nodes A and B, it is also present at all nodes along the (unique) path connecting A and B in the tree. The factor nodes in a tree-shaped factor graph correspond to nodes of the junction tree, with labels equal to their argument sets. The collapsed variable nodes correspond to edges of the junction tree. The groups of original variables at each collapsed variable node (such as $\{x_2, x_4\}$ in the figure) are called *separators*, since conditioning on all of the variables in a separator is sufficient to separate the factor graph into two or more disconnected pieces. The running intersection property ensures that we can define potentials that constrain all copies of a variable to agree.

If we collapse our factor graph all the way to a tree, we can do inference with the exact BP algorithm from above. If we leave some loops, a similar BP algorithm can still work: we can initialize all messages to be uniform, then start at an arbitrary node and use the same formulas as before

for message calculation (Equations 1–2). However, we may have to update each message several times before the marginals converge. The version of the algorithm that updates messages repeatedly until convergence is called loopy BP, or LBP. Inference with LBP is approximate because it can double-count evidence: messages to a node i from two nodes j and k can both contain information from a common neighbor l of j and k. Several researchers have empirically demonstrated that when LBP converges, the posterior marginal probabilities from Equation 3 often approximate the true marginals well (Murphy et al., 1999; McEliece et al., 1998; Zhang and Chang, 2004). If LBP oscillates between some steady states and does not converge, we can stop the process after some number of iterations; in this case, the approximate posteriors will usually be inaccurate. Although oscillations can be avoided by using "momentum" (Murphy et al., 1999), which replaces the messages that were sent at time t with a weighted average of the messages at times t and $t-1$, in some cases the approximate posteriors are still inaccurate (Murphy et al., 1999). The convergence of LBP depends on the exact graph structure and on the type and strength of the factors involved (Pearl, 1988; Heskes, 2004). Recently, researchers have developed sufficient conditions for the convergence of LBP (Weiss, 2000; Tatikonda and Jordan, 2002; Ihler et al., 2005), and a measurement of message errors has been proposed (Ihler et al., 2005).

For either exact or loopy BP, the runtime for each pass over the factor graph is exponential in the number of distinct original variables included in the largest factor. So, inference can become prohibitively expensive if our factors are too large, either because they were too large in the original graph or because we merged too many variables.

3. Factor Graphs and Structured Classification

To use belief propagation to solve structured classification problems, we need two things: a local classifier for individual instances, and a factor graph which encodes our prior beliefs about likely arrangements of instance labels. The local classifier tells us the likelihood of individual test examples under each possible class assignment, while the factor graph tells us how to trade off evidence at one example against evidence at another. We can learn local classifiers in a number of ways; for the cell image classification experiments below, we use standard support vector machines, together with a post-processing step that allows us to interpret the SVM outputs as probabilities.

There are also a number of ways to construct factor graphs that encode our beliefs about likely label vectors. In our experiments below, we construct a factor graph in two steps: first we use domain-specific heuristics to identify pairs of examples whose labels are likely to be the same, and use these pairs to build a *similarity graph* with an edge between each such pair of examples. Then, we use this similarity graph to decide what potentials to add to our factor graph. (In the protein subcellular location pattern classification problem, the similarity graph edges come from either physical proximity or similarity in appearance.) Given the similarity graph, we compare factor graphs built from several different types of potentials; the following sections introduce these potentials and discuss their advantages and disadvantages.

3.1 The Potts Potential

The simplest potential function is the Potts potential. The Potts potential is a pairwise (i.e., two-argument) factor which encourages two nodes x_i and x_j to have the same label:

$$\varphi(x_i, x_j) = \begin{cases} \omega & x_i = x_j \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

Here $\omega > 1$ is an arbitrary parameter which expresses how strongly we believe that x_i and x_j have the same label. If we use one Potts potential for each edge in the similarity graph, the overall probability of a vector of labels x is

$$P(x) = \frac{1}{Z} \prod_{\text{nodes } i} P(x_i) \prod_{\text{edges } i,j} \varphi(x_i, x_j) \tag{5}$$

where we have written Z for a normalizing constant and $P(x_i)$ for the probability which the base classifier assigns to label x_i for node i. Equations 4–5 form what is known as a *Potts model*.

Unfortunately, the Potts model does not perfectly capture our desired intuition about inference from labels of neighboring cells. To see why, consider a two-class prediction problem where node x_i has k_A neighbors of class A and k_B neighbors of class B, and suppose that classes A and B have equal prior probability for x_i given the classifier output. In this situation the ratio of posterior probabilities for classes A and B will be $\omega^{k_A - k_B}$. So for example, if ω is 2, and if x_i has 1 neighbor of class A and 3 of class B, then the ratio of probabilities will be $2^{1-3} = 1/4$. So, class A will be 1/4 as likely as class B, and $P(x_i \text{ has label } A) = 0.2$.

However, if x_i has 7 neighbors of class A and 9 neighbors of class B, the posterior probability of class A will still be 0.2, even though our intuition tells us that the probability should be much closer to 0.5 in this case. The same will hold whenever there are 2 more neighbors of class B than class A, even if the counts are 107 and 109. Worse, as class B's majority gets larger, the ratio of probabilities will approach 0 exponentially fast. So, a sufficiently strong vote from x_i's neighbors will quickly overwhelm any evidence at x_i itself—an undesirable situation.

The source of this problem is that, in Equation 5, the evidence from separate potentials has to combine multiplicatively. So, as long as the evidence from different neighbors acts through separate potentials, we will see an exponential dependence between the number of neighbors of a given class and the probability of that class. We can reduce the severity of this problem by choosing ω to be only a little bit larger than 1. But, to fix the problem we need to move to potential functions that depend on $k > 2$ nodes. Our experimental results, below, will show that potentials that combine evidence additively can perform better than the Potts potential over a wider range of inference problems.

3.2 The Voting Potential

To capture the intuition that we should be less certain about nodes whose neighbors split relatively evenly, we propose a new potential which we call the *voting potential*. In the voting potential, a node's classification is influenced by the *proportion* of classes among its neighbors, rather than the difference in class counts as in the Potts model.

The voting potential has one distinguished argument, called the *center*; the remaining arguments are called *voters*. In the factor graphs for our cell classification experiments below, we use one voting potential per cell j; the center for the jth potential is cell j itself, and the voters are the cells that are adjacent to j in the similarity graph. We will write $N(j)$ for the set of similarity-graph neighbors of cell j, so that the jth potential depends on variables $V(j) = \{j\} \cup N(j)$.

x_1	x_2	x_3	φ
0	0	0	3/4
0	0	1	1/2
0	1	0	1/2
0	1	1	1/4
1	0	0	1/4
1	0	1	1/2
1	1	0	1/2
1	1	1	3/4

Table 1: An example of the voting potential with parameter $\lambda = 2$ for $n = 2$ classes. The center node x_1 has two neighbors, x_2 and x_3.

We define the voting potential as follows:

$$\varphi_j(x_{V(j)}) = \frac{\lambda/n + \sum_{i \in N(j)} I(x_i, x_j)}{|N(j)| + \lambda}. \tag{6}$$

Here n is the number of classes, λ is a smoothing parameter, and I is an indicator function:

$$I(x_i, x_j) = \begin{cases} 1 & \text{if } x_i = x_j \\ 0 & \text{otherwise.} \end{cases}$$

(The normalization constant in the denominator of Equation 6 is irrelevant to inference, and is included only for ease of interpretation.) An example of the voting potential is given in Table 1.

The voting potential function combines the evidence from all of node x_j's neighbors into a summary vote which then influences x's classification. The parameter λ controls how much weight we put on a vote from a small number of neighboring cells. We can interpret it as the size of an additional set of fictitious neighbors whose votes are distributed uniformly; this trick limits the influence of a vote from a small number of neighbors, and is called Laplace smoothing. For example, looking at the first and fifth rows of the table, we can see that when both of x_1's neighbors are 0, x_1 is 3 times as likely to be 0 as 1, since there are $2+1$ votes for $x_1 = 0$ and $0+1$ votes for $x_1 = 1$.

The behavior of the voting potential contrasts with that of the Potts potential described in Section 3.1, in which each neighbor separately influences the center node without reference to the other neighbors. In line with our intuition, we will see below that networks which use the voting potential can yield more accurate results than the Potts model for structured classification problems.

3.3 The AMN Potential

The associative Markov network (AMN) potential (Taskar et al., 2004) is defined to be

$$\varphi(x_1 \ldots x_k) = 1 + \sum_{y=1}^{n} (\omega_y - 1) I(x_1 = x_2 = \ldots = x_k = y) \tag{7}$$

for parameters $\omega_y > 1$, where $I(\text{predicate})$ is defined to be 1 if the predicate is true and 0 if it is false. So, the AMN potential is constant unless all of the variables $x_1 \ldots x_k$ are assigned to the same class y, in which case it is equal to ω_y.

The AMN potential reduces to the Potts potential when $k = 2$ and $\omega_y = \omega$ for all y. So, in this case it inherits the same problems that the Potts potential has. For any k, the AMN potential has no direct effect on a label vector's likelihood unless *all* of the k argument variables agree on the label y. (It affects all vectors' likelihoods indirectly through the normalizing constant.) As k gets larger, there are comparatively fewer label vectors where all k labels agree, and so the AMN potential may not have much influence on the overall posterior distribution over label vectors unless the cell-level classifiers already were very close to agreement. And in fact, our experiments below demonstrate that factor graphs based on the AMN potential perform best when the neighborhood size k is relatively small (but larger than 2).

4. Decomposable Potentials

While k-way factors can lead to more accurate inference, they can also slow down belief propagation. For a general k-way factor, it takes time exponential in k even to look at all of the entries. So, we cannot expect to find inference algorithms for general k-way potentials that take less than exponential time.

For specific k-way potentials, though, we can hope to take advantage of special structure to design a fast inference algorithm. In particular, for many interesting potential functions, we can write down an algorithm which efficiently performs sums of the form required for message computation:

$$\sum_{x_1} \sum_{x_2} \cdots \sum_{x_{k-1}} \varphi_j^*(x_1, \ldots, x_k), \tag{8}$$

$$\varphi_j^*(x_1, \ldots, x_k) = m_1(x_1) m_2(x_2) \ldots m_k(x_{k-1}) \varphi_j(x_1, \ldots, x_k). \tag{9}$$

Here $m_i(x_i)$ is the message to factor j from variable x_i. (If we removed loops in our factor graph by collapsing groups of variables, then Equation 9 may look instead like

$$\varphi_j^*(x_1, \ldots, x_k) = m_{12}(x_1, x_2) m_{245}(x_2, x_4, x_5) \ldots m_{k-1}(x_{k-1}) \varphi_j(x_1, \ldots, x_k).$$

That is, the argument sets of the messages may overlap with one another. The derivations below apply equally well to either expression for φ_j^*.)

For example, as we will see below, we can compute Equations 8–9 quickly if φ_j is a sum of terms $\sum_l \psi_{jl}$ where each term ψ_{jl} depends only on a small subset of its arguments $x_1 \ldots x_k$. Or, we can compute Equations 8–9 quickly if φ_j is constant except at a small number of input vectors $(x_1, \ldots, x_k)$. In the first case we will say that φ_j is a sum of *low-arity* terms ψ_{jl}, and in the second case we will say that φ_j is *sparse*.

More generally, suppose that φ_j^* in Equation 8 can be written as a sum of products of low-arity functions: writing ψ_{jl} for a generic term in the sum and ξ_{jlm} for a generic factor of ψ_{jl},

$$\varphi_j^*(x_1, \ldots, x_k) = \sum_{l=1}^{L_j} \psi_{jl}(x_1, \ldots, x_k) = \sum_{l=1}^{L_j} \prod_{m=1}^{M_{jl}} \xi_{jlm}(x_{V(j,l,m)}) \tag{10}$$

where the set of indices $V(j,l,m) \subseteq \{1 \ldots k\}$ tells us which variables ξ_{jlm} depends on. Also suppose that, for each term ψ_{jl} in the sum, the sets $V(j,l,m)$ can be arranged into a junction tree. That is, suppose that we can build a cycle-free graph on M_{jl} nodes, with one node labeled with $V(j,l,m)$ for each m, which satisfies the running intersection property.

If φ_j^* satisfies the above properties, and if L_j, M_{jl}, and $|V(j,l,m)|$ are small for all j, l, and m, then we will say that φ_j^* is *decomposable*. And, we will say that φ_j is a *decomposable potential* (in the context of this message computation). Decomposable potentials include the special cases mentioned above, namely sparse potentials and potentials that are sums of low-arity terms, as well as a wide variety of other examples which we will describe in more detail below. We will see below that we can evaluate Equations 8–9 quickly for a decomposable potential by using an algorithm very similar to belief propagation.

5. Belief Propagation with Decomposable Potentials

When we are running BP or loopy BP on a factor graph with decomposable potentials, we can accelerate the computation of the belief messages that would otherwise be slow to compute. There are two types of messages that we need for BP or loopy BP, shown in Equations 1 and 2.

Messages from a variable (or a separator, which we treat as a single large variable) to a factor are fast to compute in any case: we can calculate them from Equation 1 by looping over the incoming edges at node x_i and, for each edge, looping over the n possible classes we can assign to x_i. So, we do not need to accelerate the computation of these messages.

Messages from a factor to a variable (Equation 2) are slow to compute naïvely if the factor connects to many other variables. In this section we will show that we can calculate these messages efficiently for decomposable potentials of the form shown in Equation 10. So, this section shows that we can implement BP and loopy BP efficiently for decomposable potentials. The derivation of this section works with general decomposable potentials; below, in Sections 6.1 and 6.2, we will work out the formulas for specific cases including the voting potential and the associative Markov network potential.

The algorithm that we will use to compute the messages is essentially the same as the overall belief propagation algorithm. So, when we perform inference on a factor graph with decomposable potentials, we will be running two nested copies of belief propagation: the inner copy will act on a single decomposable potential, and will compute the messages which the outer copy needs to send from that potential.

Substituting Equations 8–10 into Equation 2 and rearranging terms tells us that the desired message is

$$\begin{aligned} m_{j\to k}(x_k) &= \sum_{x_1}\sum_{x_2}\cdots\sum_{x_{k-1}} \varphi_j(x_1,\ldots,x_k)\prod_{i=1}^{k-1} m_{i\to j}(x_i) \\ &= \sum_{x_1}\sum_{x_2}\cdots\sum_{x_{k-1}} \left(\sum_{l=1}^{L_j}\prod_{m=1}^{M_{jl}} \xi_{jlm}(x_{V(j,l,m)})\right) \\ &= \sum_{l=1}^{L_j}\sum_{x_1}\sum_{x_2}\cdots\sum_{x_{k-1}}\prod_{m=1}^{M_{jl}} \xi_{jlm}(x_{V(j,l,m)}). \end{aligned} \tag{11}$$

Now let us fix l temporarily, leaving a term of the form

$$\sum_{x_1}\sum_{x_2}\cdots\sum_{x_{k-1}}\prod_{m=1}^{M_{jl}} \xi_{jlm}(x_{V(j,l,m)}). \tag{12}$$

We have assumed that the sets $V(j,l,m)$ for $m \in \{1,\ldots,M_{jl}\}$ can be arranged into a junction tree. Pick a node of this junction tree whose label contains x_k, and call this node the root. Refer to each node by its index m, and write $\mathrm{par}(m)$ for the parent of node m.

Now pick an arbitrary leaf of the junction tree. Without loss of generality, say that this leaf has index $m = 1$. We can partition the variables in the set $V(j,l,1)$ into two subsets,

$$C(1) = V(j,l,1) \cap V(j,l,\mathrm{par}(1)) \quad \text{and} \quad D(1) = V(j,l,1) \setminus V(j,l,\mathrm{par}(1)).$$

$C(1)$ contains the variables that node 1 has in Common with its parent, while $D(1)$ contains the variables from node 1 that are Distinct from its parent's variables. The variables in $D(1)$ can only appear in $V(j,l,1)$: if any of them appeared in $V(j,l,m)$ for $m \neq 1$ it would violate the running intersection property, since any path from 1 to m has to pass through $\mathrm{par}(1)$.

Without loss of generality, suppose that the variables in $D(1)$ are numbered consecutively from 1, say $D(1) = \{1,2\}$. That means that the factor ξ_{jl1} depends on x_1 and x_2, but no other factor ξ_{jlm} for $m \neq 1$ depends on x_1 or x_2. So, by the distributive law, we can rearrange Equation 12 as follows:

$$\sum_{x_3} \cdots \sum_{x_{k-1}} \prod_{m=2}^{M_{jl}} \xi_{jlm}(x_{V(j,l,m)}) \cdot \left(\sum_{x_1} \sum_{x_2} \xi_{jl1}(x_{V(j,l,1)}) \right) \tag{13}$$

To get Equation 13 we have moved the x_1 and x_2 summations inward as far as possible.

We can think of the expression in parentheses in Equation 13 as a message which travels from node 1 to node $\mathrm{par}(1)$ of the junction tree: it depends only on the variables in the set $C(1)$, and it summarizes everything that node $\mathrm{par}(1)$ needs to know about node 1 in order to compute the term of Equation 12. We will write $\mu_{1 \to \mathrm{par}(1)}$ for this message, that is,

$$\mu_{1 \to \mathrm{par}(1)}(x_{C(1)}) = \sum_{x_1} \sum_{x_2} \xi_{jl1}(x_{V(j,l,1)}). \tag{14}$$

We can continue computing messages in this fashion from leaf nodes of the junction tree to their parents. After several steps our expression will look something like this:

$$\sum_{x_7} \cdots \sum_{x_{k-1}} \prod_{m=4}^{M_{jl}} \xi_{jlm}(x_{V(j,l,m)}) \prod_{m=1}^{3} \mu_{m \to \mathrm{par}(m)}(x_{C(m)}). \tag{15}$$

Here we have eliminated the variables x_1 through x_6 and computed the messages from nodes $m = 1,2,3$ to their parents. At this point suppose that node $m = 4$ is an internal node of the junction tree, and that it has as its only child node $m = 1$. Because we have already computed the message from node 1 to node 4, we can now compute the message from 4 to $\mathrm{par}(4)$. Suppose that $D(4) = \{7\}$; then we can rearrange Equation 15 as follows.

$$\begin{aligned} &\sum_{x_8} \cdots \sum_{x_{k-1}} \prod_{m=5}^{M_{jl}} \xi_{jlm}(x_{V(j,l,m)}) \times \\ &\qquad \prod_{m=2}^{3} \mu_{m \to \mathrm{par}(m)}(x_{C(m)}) \left(\sum_{x_7} \mu_{1 \to 4}(x_{C(1)}) \xi_{jl4}(x_{V(j,l,4)}) \right). \end{aligned} \tag{16}$$

Here we have used the distributive law to move the x_7 summation inward as far as possible. As above, none of the functions ξ_{jlm} for $m > 4$ can depend on x_7: if they did, then by the running

intersection property, 7 would have to be an element of $V(\text{par}(4))$ and couldn't be in $D(4)$. The message $\mu_{1\to 4}$ could depend on x_7, since 7 could be in $C(1)$. So, to be safe, we have left $\mu_{1\to 4}$ inside the x_7 summation. On the other hand, the messages $\mu_{2\to\text{par}(2)}$ and $\mu_{3\to\text{par}(3)}$ cannot depend on x_7: if one of them did, 7 would have to be in $V(j,l,\text{par}(2))$ or $V(j,l,\text{par}(3))$, which would again violate the running intersection property.

In a natural generalization of Equation 14, we will write $\mu_{4\to\text{par}(4)}(x_{C(4)})$ for the expression in parentheses in Equation 16. It should be clear at this point that we can compute a message from each node in the junction tree to its parent, so long as we work from the leaves upward; the message from a node m to its parent $\text{par}(m)$ will depend on the messages from m's children to m. Once we process all of the children of the root r, we will be left with an expression that contains summations only over variables in $V(j,l,r)$. (In fact, it will contain summations over exactly the variables in $V(j,l,r)\setminus\{k\}$.) We can perform these summations to find the desired term (Equation 12), which is a function only of x_k. We can then repeat the process for each l to get all of the terms in Equation 11.

The above algorithm computes the message from a factor φ_j to a single neighboring variable x_k. If we want the messages from φ_j to all of its variables we can run the above algorithm multiple times; however, the multiple runs will redundantly recompute many of the messages $\mu_{s\to t}$. Instead, as is usual for the belief propagation algorithm, we can combine all of the runs into a single computation which passes one message in each direction over each edge of the junction tree.

If we have merged groups of variables in constructing our outer junction tree, then there are two modifications needed for the above analysis. First, if the argument sets of the incoming messages at a given factor node overlap, we may need to build different inner junction trees to compute different outgoing messages. So, we may not be able to share computation as described in the previous paragraph. This loss of sharing may increase our runtime by a small factor. Second, and more importantly, if the desired outer message depends on several original variables, then there may be no node of our inner junction tree that contains all the needed variables. In this case we may condition on the possible values of some of the needed variables, and use several runs of inner message passing, each of which computes a slice of the desired outer message. In general, there may be several ways to decompose a potential, and several possible choices of which variables to condition on for a given decomposition. Each of these setups may lead to a different runtime for message computation. See Section 6.3 below and Kjærulff (1998) for more details.

6. Instances of Decomposable Potentials

Decomposable potentials are common and useful. In this section, we discuss the details and derivations of message passing with the voting potential, the associative Markov network potential, and the nested junction tree.

6.1 Decomposing the Voting Potential

The general BP-style algorithm of Section 5 is more complicated than we need when we are computing messages for the voting potential: since Equation 6 is a sum of low-arity functions rather than a sum of products of low-arity functions, the computation for each term in the sum is particularly simple. So, in this section we will derive efficient expressions for the necessary messages.

There are two types of messages we need to think about: those from a factor φ_j to the node x_k that φ_j is centered on, and those from a factor φ_j to some non-centered variable node x_i with $i\neq k$. To simplify notation, assume that $V(j)=\{1,\ldots,k\}$; also assume that we have normalized the

messages $m_{i\to j}(x_i)$ so that $\sum_{x_i} m_{i\to j}(x_i) = 1$. With these assumptions, the message from a factor φ_j to its center variable node x_k can be computed as follows:

$$\begin{aligned}
&(\lambda + |N(j)|)\, m_{j\to k}(x_k) \\
&= \sum_{x_1}\sum_{x_2}\cdots\sum_{x_{k-1}} \left(\lambda/n + \sum_{i=1}^{k-1} I(x_i, x_k)\right) \prod_{i'=1}^{k-1} m_{i'\to j}(x_{i'}) \\
&= \frac{\lambda}{n}\sum_{x_1}\sum_{x_2}\cdots\sum_{x_{k-1}}\prod_{i'=1}^{k-1} m_{i'\to j}(x_{i'}) + \\
&\quad \sum_{x_1}\sum_{x_2}\cdots\sum_{x_{k-1}}\sum_{i=1}^{k-1} I(x_i, x_k)\prod_{i'=1}^{k-1} m_{i'\to j}(x_{i'}) \\
&= \frac{\lambda}{n} + \sum_{i=1}^{k-1}\sum_{x_1}\sum_{x_2}\cdots\sum_{x_{k-1}} I(x_i, x_k)\prod_{i'=1}^{k-1} m_{i'\to j}(x_{i'}) \\
&= \frac{\lambda}{n} + \sum_{i=1}^{k-1}\sum_{x_1}\sum_{x_2}\cdots\sum_{x_{k-1}} I(x_i, x_k) m_{i\to j}(x_i) \prod_{i'=1, i'\neq i}^{k-1} m_{i'\to j}(x_{i'}) \\
&= \frac{\lambda}{n} + \sum_{i=1}^{k-1}\sum_{x_i} I(x_i, x_k) m_{i\to j}(x_i) \\
&= \frac{\lambda}{n} + \sum_{i=1}^{k-1} m_{i\to j}(x_k).
\end{aligned}$$

The first equation above is the definition of the desired message. The second equation distributes multiplication over addition. The third equation uses the fact that all terms in the product $\prod_{i'=1}^{k-1} m_{i'\to j}(x_{i'})$ are independent, along with our assumption $\sum_{x_{i'}} m_{i'\to j}(x_{i'}) = 1$, to compute the first summation. The fourth equation factors $m_{i\to j}$ out of the product. The fifth equation uses again the facts that all terms in the product are independent and $\sum_{x_{i'}} m_{i'\to j}(x_{i'}) = 1$. The last line uses the fact that $I(x_i, x_k)$ is nonzero iff $x_i = x_k$.

The message from a factor j to a non-centered variable can be computed similarly. Under the same assumptions as above, we will calculate the message $m_{j\to 1}$:

$$\begin{aligned}
&(\lambda + |N(j)|)\, m_{j\to 1}(x_1) \\
&= \sum_{x_2}\cdots\sum_{x_k} \left(\lambda/n + \sum_{i=1}^{k-1} I(x_i, x_k)\right) \prod_{i'=2}^{k} m_{i'\to j}(x_{i'}) \\
&= \frac{\lambda}{n} + \sum_{x_2}\cdots\sum_{x_k}\sum_{i=1}^{k-1} I(x_i, x_k)\prod_{i'=2}^{k} m_{i'\to j}(x_{i'}) \\
&= \frac{\lambda}{n} + \sum_{x_2}\cdots\sum_{x_k} I(x_1, x_k)\prod_{i'=2}^{k} m_{i'\to j}(x_{i'}) + \sum_{x_2}\cdots\sum_{x_k}\sum_{i=2}^{k-1} I(x_i, x_k)\prod_{i'=2}^{k} m_{i'\to j}(x_{i'}) \\
&= \frac{\lambda}{n} + \sum_{x_k} I(x_1, x_k) m_{k\to j}(x_k) + \sum_{x_2}\cdots\sum_{x_k}\sum_{i=2}^{k-1} I(x_i, x_k)\prod_{i'=2}^{k} m_{i'\to j}(x_{i'}) \\
&= \frac{\lambda}{n} + m_{k\to j}(x_1) + \sum_{x_2}\cdots\sum_{x_k}\sum_{i=2}^{k-1} I(x_i, x_k)\prod_{i'=2}^{k} m_{i'\to j}(x_{i'})
\end{aligned}$$

$$
\begin{aligned}
&= \frac{\lambda}{n} + m_{k\to j}(x_1) + \sum_{i=2}^{k-1} \sum_{x_2} \ldots \sum_{x_k} I(x_i, x_k) m_{i\to j}(x_i) m_{k\to j}(x_k) \prod_{i'=2, i'\neq i}^{k-1} m_{i'\to j}(x_{i'}) \\
&= \frac{\lambda}{n} + m_{k\to j}(x_1) + \sum_{i=2}^{k-1} \sum_{x_i} \sum_{x_k} I(x_i, x_k) m_{i\to j}(x_i) m_{k\to j}(x_k) \\
&= \frac{\lambda}{n} + m_{k\to j}(x_1) + \sum_{i=2}^{k-1} \sum_{x_k} m_{i\to j}(x_k) m_{k\to j}(x_k).
\end{aligned}
$$

The first equality above is the definition of the desired message. The second pulls the term λ/n out of the sums, using the facts that the terms in the product $\prod_{i'=2}^{k} m_{i'\to j}(x_{i'})$ are independent and $\sum_{x_{i'}} m_{i'\to j}(x_{i'}) = 1$. The third equality splits the sum over i into two pieces, one for $i = 1$ and the other for $i \geq 2$. The fourth equality uses the independence of the terms in the left-hand product to simplify away the summations over x_2 through x_{k-1}. The fifth uses the fact that $I(x_1, x_k) = 0$ when $x_1 \neq x_k$. The sixth equality pulls the terms $m_{i\to j}$ and $m_{k\to j}$ out of the remaining product. The seventh uses the independence of terms in the product to simplify away the summations over variables other than x_i and x_k. And the last equality uses the fact that $I(x_i, x_k) = 0$ when $x_i \neq x_k$.

Despite the fact that the variables $x_1, \ldots, x_k$ have exponentially many possible assignments, the above derivations show that we can compute the messages $m_{j\to k}$ and $m_{j\to 1}$ exactly and almost instantaneously. The message $m_{j\to k}$ is particularly easy to interpret: it is the (Laplace smoothed) average of the messages from $x_1, \ldots, x_{k-1}$.

6.2 Decomposing the AMN Potential

It is even easier to derive an efficient expression for the messages from an AMN potential than it was for the messages from a voting potential. As before, let us assume that $V(j) = \{1, \ldots, k\}$, that we desire the message $m_{j\to k}(x_k)$, and that we have normalized the messages $m_{i\to j}(x_i)$ to sum to 1 over x_i for each $i = 1, \ldots, k-1$. Since the AMN potential is symmetric in its arguments, there is only one type of message to calculate.

We can write Equation 7 in the form of Equation 10 by noting that

$$I(x_1 = x_2 = \ldots = x_k = y) = I(x_1 = y) I(x_2 = y) \ldots I(x_k = y)$$

and that each function $I(x_i = y)$ depends on only one variable x_i. Given this representation, the desired message is:

$$
\begin{aligned}
m_{j\to k}(x_k) &= \sum_{x_1} \sum_{x_2} \ldots \sum_{x_{k-1}} \left(1 + \sum_y (\omega_y - 1) \prod_{i=1}^{k} I(x_i, y) \right) \prod_{i'=1}^{k-1} m_{i'\to j}(x_{i'}) \\
&= 1 + \sum_{x_1} \sum_{x_2} \ldots \sum_{x_{k-1}} \sum_y (\omega_y - 1) \prod_{i=1}^{k} I(x_i, y) \prod_{i'=1}^{k-1} m_{i'\to j}(x_{i'}) \\
&= 1 + \sum_y (\omega_y - 1) I(x_k, y) \sum_{x_1} \sum_{x_2} \ldots \sum_{x_{k-1}} \prod_{i=1}^{k-1} I(x_i, y) m_{i\to j}(x_i) \\
&= 1 + \sum_y (\omega_y - 1) I(x_k, y) \prod_{i=1}^{k-1} m_{i\to j}(y) \\
&= 1 + (\omega_{x_k} - 1) \prod_{i=1}^{k-1} m_{i\to j}(x_k).
\end{aligned}
$$

The first equality above is the definition of the desired message. The second pulls the constant 1 outside of the sums and uses the independence of terms in the product. The third rearranges the order of the sums and products. The fourth uses the fact that the product is zero unless $x_i = y$ for all $i \in \{1,\ldots,k-1\}$. The fifth uses the fact that $I(x_k,y)$ is 0 unless $x_k = y$.

6.3 The Nested Junction Tree

The nested junction tree method (Kjærulff, 1998) can speed up message propagation in the junction trees that arise when we remove loops from a factor graph. In particular, it helps compute messages from the large factors that arise when we merge groups of variables. It works by noticing that each of these messages is computed from the product of many smaller factors, which can sometimes be arranged into a nontrivial "inner" junction tree.

Unlike the previous two examples (the voting and AMN potentials), the nested junction tree method does not attempt to look inside the factors of the original factor graph. Instead, it keeps track of the variable merges during junction tree construction, and sometimes is able to "undo" some of these merges temporarily to provide computational savings.

For example, the factor graph of Fig. 4 can be collapsed into a junction tree by merging x_2 and x_3 as shown. The time and space costs of belief propagation on this junction tree are dominated by its largest potential, φ_{2345}. Consider the message from φ_{2345} to x_{23}:

$$m_{2345\to 23}(x_2,x_3) = \sum_{x_4}\sum_{x_5} \varphi_{2345}(x_2,x_3,x_4,x_5) m_{4\to 2345}(x_4) m_{5\to 2345}(x_5).$$

Standard belief propagation will first compute

$$\begin{aligned} \varphi^*_{2345}(x_2,x_3,x_4,x_5) &= \varphi_{2345}(x_2,x_3,x_4,x_5) m_{4\to 2345}(x_4) m_{5\to 2345}(x_5) \\ &= \varphi_{245}(x_2,x_4,x_5)\varphi_{345}(x_3,x_4,x_5) m_{4\to 2345}(x_4) m_{5\to 2345}(x_5) \end{aligned}$$

for all settings of (x_2,x_3,x_4,x_5), and then marginalize φ^*_{2345} to get $m_{2345\to 23}$.

If there are (for example) ten possible settings of each original variable, the space required for computing $m_{2345\to 23}$ with standard belief propagation is 10100 locations: 10^4 for storing φ^*_{2345}, and 10^2 for storing $m_{2345\to 23}$. And, the time cost is 39900 flops: for each of the 10^4 settings of (x_2,x_3,x_4,x_5) we must multiply together one element each from the tables φ_{245}, φ_{345}, $m_{4\to 2345}$, and $m_{5\to 2345}$, at a cost of 3 flops per iteration. Then, for each of the 10^2 elements of $m_{2345\to 23}$, we must sum 10^2 elements of φ^*_{2345} (using 10^2-1 flops), leading to a total cost of $3\times 10000 + 99\times 100$.

However, if we examine the message computations in more detail, it turns out that we can take advantage of the structure of φ^*_{2345} to save time and space. Since φ^*_{2345} was formed by multiplying together four smaller tables, and since the argument sets of these smaller tables form a junction tree, φ^*_{2345} is decomposable. (The inner junction tree is shown at the far right of Fig. 4.) Using this fact, we can find $m_{2345\to 23}$ by passing messages through the inner junction tree several times. In more detail, suppose we pick the factor φ_{345} as the root of the inner junction tree. This factor contains one of the message variables (x_3) but not the other one (x_2). So, we must condition on each value of x_2 in turn. For each value of x_2, we first compute the intermediate message

$$\varphi^{x_2}(x_4,x_5) = \varphi_{245}(x_2,x_4,x_5) m_{4\to 2345}(x_4) m_{5\to 2345}(x_5).$$

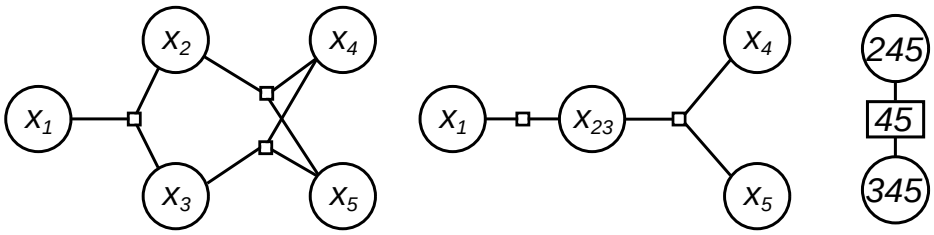


Figure 4: A factor graph for which the nested junction tree method can speed up belief propagation. Left: original factor graph. Middle: factor graph after merging x_2 and x_3. Right: inner junction tree for computing the message $m_{2345\to 23}$.

This message is associated with the separator $\{x_4, x_5\}$ in the inner junction tree. We can then incorporate this inner message into the factor $\{x_3, x_4, x_5\}$, and marginalize to get a slice of our desired (outer) message:

$$m_{2345\to 23}(x_2, x_3) = \sum_{x_4}\sum_{x_5} \varphi_{345}(x_3, x_4, x_5)\varphi^{x_2}(x_4, x_5).$$

Each pass through the inner junction tree keeps x_2 fixed, computing a slice $m_{2345\to 23}(x_2, \cdot)$ of the outer message. (By $m_{2345\to 23}(x_2, \cdot)$, we mean a table of the values of $m_{2345\to 23}$ for a fixed value of the first argument and all possible values of the second argument.)

The nested belief propagation algorithm for computing $m_{2345\to 23}$ saves us both space and time. For time, the cost to compute an element of one of the φ^{x_2} tables is 2 multiplications; there are 10^2 elements of each table, and 10 tables, so the total cost of this step is $2 \times 10 \times 100 = 2000$ flops. To compute $\varphi_{345}(x_3, x_4, x_5)\varphi^{x_2}(x_4, x_5)$ for fixed x_2 takes 1000 flops (one per table element), and to marginalize out (x_4, x_5) takes 99 flops for each of the 10 resulting elements, for a total of $1000 + 10 \times 99 = 1990$ per slice. Since there are 10 slices, we need 19900 flops for all of them. The grand total is therefore $19900 + 2000 = 21900$, a savings of approximately 45% compared to standard belief propagation.

The space savings are even greater: we can reuse a single array of size 10^2 for all of the φ^{x_2} tables, and a single array of size 10^3 for all of the products $\varphi_{345}(x_3, x_4, x_5)\varphi^{x_2}(x_4, x_5)$. Adding in the cost of storing the final message, we have a space cost of $100 + 1000 + 100 = 1200$ locations, a savings of about 88%.

An important limitation of the nested junction tree method is the following lemma, which provides a lower bound on inference time based on the number of variables in the largest node label (or clique):

Lemma 1 *In a junction tree T with binary variables, let C be the largest clique, and say that we wish to compute the posterior marginal for some variable x_q. So long as C is minimal, the nested junction tree method cannot reduce the time for this inference task to less than $\Omega(2^{|C|})$. (The time for non-binary variables is at least as large.)*

Proof Each edge (i, j) in clique C arises either because there is a potential that directly links variables x_i and x_j (this case includes so-called "moral edges"), or because we triangulated a chordless cycle that contains x_i and x_j (in which case some other clique sends C a message that links x_i with x_j). The nested junction tree method achieves its speedup by avoiding consideration of some of

these edges when computing outgoing messages from C: for a message M from C to some other clique D, we can ignore the corresponding incoming message N from D to C. Without N, we may be able to ignore as many as $\binom{|M|}{2}$ of the edges of C while calculating M, since each edge that connects two variables in M may be supported only by N.

Because we can ignore some of the edges of C, we can avoid building a single large table of size $2^{|C|}$, and instead run a nested copy of the message passing algorithm to find M. This inner message passing algorithm works on an inner junction tree T' built from the remaining edges of C. To compute M, we pick some clique C' of T' as root; then, for each possible setting of the variables in $M \setminus C'$, we pass messages from leaves to root in T' to find a slice of M. This slice corresponds to the fixed setting of $M \setminus C'$, and covers all possible settings of the variables in $M \cap C'$.

To calculate the total cost of these inner runs of message passing, we need to look at the structure of T'. In particular, we need to figure out the size of the largest clique of T'. For this purpose, we can divide the variables of T' into three sets: those in $M \setminus C'$ (which we hold fixed during each inner iteration), those in $M \cap C'$ (which form the resulting slice of M), and those in $C \setminus M$ (the rest of C). The variables in $M \cap C'$ are fully connected to one another, since they are all members of C'. And, the variables in $C \setminus M$ are fully connected to one another, since none of them are covered by the incoming message N. But, these two sets of variables are also fully connected to each another: no edge between $C \setminus M$ and $M \cap C'$ can be covered by N, since each such edge has one vertex in M and one outside of M. So, $C \setminus M$ and $M \cap C'$ together form a clique of T'.

Recapping, we have $2^{|M \setminus C'|}$ inner runs of message passing, each of which works with a junction tree containing a clique of size $|C \setminus M| + |M \cap C'|$. We will now prove by induction that inference takes time at least $k2^{|C|}$, where k is an implementation-dependent constant.

For the inductive step, our runtime for calculating M is at least

$$2^{|M \setminus C'|} \times k2^{|C \setminus M| + |M \cap C'|}$$

for $2^{|M \setminus C'|}$ runs of message passing, each of which costs $k2^{|C \setminus M| + |M \cap C'|}$ by the inductive hypothesis. Since $|M \setminus C'| + |C \setminus M| + |M \cap C'| = |C|$, our runtime is therefore at least $k2^{|C|}$, as claimed.

There are several possible base cases. The most obvious is when $|C| = 1$; in this case we can choose $k > 0$ so that the runtime is at least $2k$. The induction can also bottom out if the nested junction tree method becomes inapplicable at any step. There are two ways it can become inapplicable: first, if $M \setminus C' = \emptyset$, then the argument above means that the nested junction tree has only a single clique of size $|C|$, and the nested junction tree method therefore offers no speedup. In this case we need to build a table of size $2^{|C|}$ for inference. So, we can again choose $k > 0$ so that our runtime is at least $k2^{|C|}$.

Second, the nested junction tree method is inapplicable if clique C has no outgoing messages. C has no outgoing messages if and only if we select it as the root for message passing. In this case, C gets incoming messages from all of its neighbors, and the nested junction tree method again offers no speedup: to perform any inference task on C's fully-connected graph, we must again build a table of size $2^{|C|}$ and take time at least $k2^{|C|}$ for some $k > 0$.

So, by choosing $k > 0$ small enough to satisfy all of the above base cases, we have completed the induction. The only remaining detail is the question of non-binary variables. But, it should be obvious from the proof above that any non-binary variables can only increase the cost of inference. ■

Lemma 1 means that we cannot expect the nested junction tree method on its own to make it practical

to work with trees with very large cliques. This conclusion is borne out by the experimental results of Kjærulff (1998), Tables 1 and 2: the time savings shown there are never greater than about 60%. In contrast, the examples of the previous two sections show that decomposable potentials in general can lead to far greater savings: in these examples, the message calculation time goes from exponential to polynomial in the size of the clique, and Fig. 10 below shows that this change can easily result in speedups of multiple orders of magnitude.

7. Prior Updating

By calculating messages as described in Section 6.1, we can run loopy belief propagation on a factor graph that includes voting potentials. However, we might expect the messages from a factor φ_j to a non-centered variable x_i (where $i \neq c_j$) to be fairly weak: the overall vote of all of x_{c_j}'s neighbors will not be influenced very much by x_i's single vote, so there will not be a strong penalty if x_i votes the wrong way.

This observation suggests an even simpler algorithm for inference: we can run loopy BP but ignore all of the messages from factors to non-centered variables. (Ignoring a message means considering it to be uniform.) We will call this algorithm Prior Updating, or PU, since it works by using the current classifications of a node's neighbors to update the prior for the node's own classification. Our group proposed a version of Prior Updating for inference on graphs (Chen and Murphy, 2006) before the work described in the current paper, which explores its relationship to potential functions in loopy belief propagation.

We can expect that PU will be noticeably faster than loopy BP, since there will usually be many more non-centered variables than there are centered ones in each factor. We might also hope that PU could be more accurate than loopy BP, since it is less prone to double-count evidence: we have broken any loops which would allow a message that x_{c_j} sends to factor φ_j to return back and influence the classification of x_{c_j}. (As it turns out, we will see below that in our experiments PU is often slightly worse and sometimes slightly better than loopy BP on factor graphs with voting potentials.) We will examine the speed and classification performance of loopy BP, PU, and other inference methods in Section 9. PU can be seen as an approximation of LBP on factor graphs with voting potentials, and like LBP, is not guaranteed to converge. However, in our experiments, we never observed problems with convergence for PU or for LBP with the voting potential.

8. Experimental Materials and Methods

2D HeLa Image Set We applied our methods to the problem of classifying subcellular patterns in an image of many cells, a problem that we have formalized previously (Chen and Murphy, 2006). The starting point is a set of fluorescence microscope images of HeLa cells created by introducing antibodies and molecular probes against proteins in major subcellular organelles (Boland and Murphy, 2001). The data set contains 862 single-cell images from ten classes, with each class having between 73 and 98 images. The true class of each image is known with certainty since the fluorescent probe present in each slide is known.

Classifying protein subcellular patterns is important since it allows us to identify where a protein is located in cell organelles, which is required for it to carry out its specific function (Boland and Murphy, 2001). This knowledge is critical to understanding how that protein works in a cell, and to describing cell behaviors under different conditions.

In the past, the most common way to determine protein location patterns has been by human interpretation of fluorescence microscope images. In recent years, however, automated systems have been developed for consistent and objective interpretation of such images; the current paper's techniques are designed to help the accuracy of these automated systems (Chen et al., 2006c; Glory and Murphy, 2007). Automated systems, when available, can be preferable to human image annotation, since they can help avoid human biases and errors. They also make it possible to analyze images from high-throughput microscopy, which would otherwise be too numerous for humans to handle.

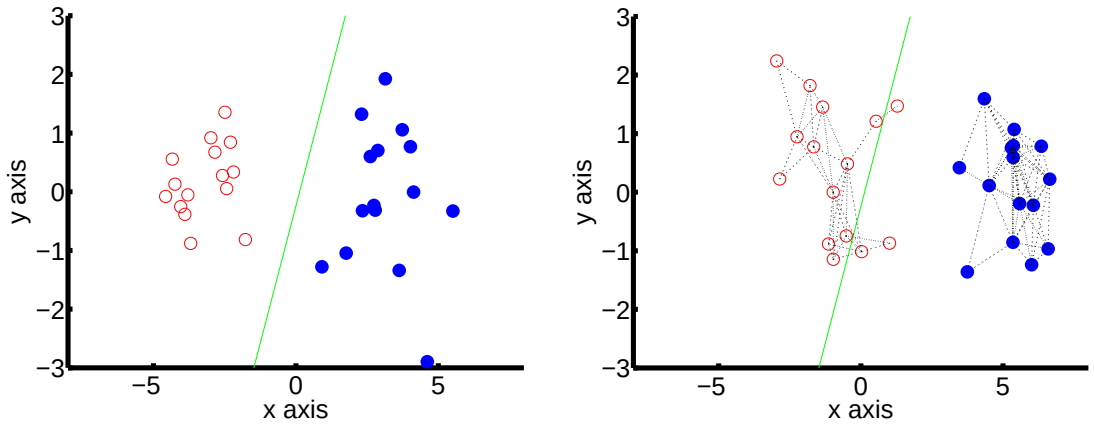
Subcellular Location Features Several sets of informative features have been developed to describe protein subcellular patterns (Boland and Murphy, 2001; Chen et al., 2006c; Glory and Murphy, 2007). These features, termed Subcellular Location Features (SLFs), are of several types, including Zernike moment features, Haralick texture features, morphological features and wavelet features. The details for different versions of SLFs are reviewed elsewhere (Huang and Murphy, 2004). The best single-cell classification results obtained to date with a single feature set for the 2D HeLa data set were with feature set SLF16 (Huang and Murphy, 2004), which we have therefore used in the work described here. In this feature set, each cell is represented by a feature vector f of length $d = 47$.

Support Vector Machine To determine the evidence for each individual cell we used Support Vector Machine classifiers (Cortes and Vapnik, 1995), as implemented in the LIBSVM library version 2.82 (Chang and Lin, 2001). To handle problems with $k > 2$ classes, we learned $k(k-1)/2$ binary SVM classifiers, one for each pair of classes. We then derived probability estimates by applying sigmoid functions to the decision values of these SVMs, in an improved implementation (Lin et al., 2003) of the Platt Scaling method (Platt, 2000).

Simulating Multi-Cell Images We are interested in the simultaneous classification of all of the cells in a multi-cell microscope image. Unfortunately, it is difficult to collect multi-cell images for which we know the ground truth classification of each cell: if we prepare a slide from a mixture of two or more types of cells, we do not have direct control over which type appears where. For this reason, we simulated the multi-cell problem by creating synthetic multi-cell images using multiple real single-cell HeLa images.[1] To generate a structured classification problem, we selected two of the ten classes at random; then we selected N_1 images from the first class and N_2 from the second, with $N_1 + N_2 = 12$. We then treated these images as if they were the output of a segmentation algorithm that was run on a multi-cell image.

Constructing the Similarity Graph As an intermediate step in the construction of our factor graph, we built a *similarity graph*: the nodes of this graph correspond to cells, and an edge between two cells means that we believe that they are likely to share the same label. The simplest approach to building the similarity graph is to include all possible edges. With a fully-connected similarity graph, all cells in the test image are considered equally similar to one another; such a graph expresses a prior belief that images containing a few groups of same-class cells are more likely than images containing cells of many different classes. Our experiments below show that even this modest amount of prior information can improve the accuracy of our classifier compared to the no-edges (independent classification) case; for example, in Figs. 7–9, the performance of LBVP is higher

1. We are in the process of analyzing true multi-cell images with known ground truth, which we collect by tagging one of the cell types with a fluorescent marker at a wavelength different from the one used to label the target protein.



(a) Training Stage

(b) Testing Stage

Figure 5: Classification of multiple examples using proximity in feature space. (a) At the training stage, a linear classifier separates two classes in a 2D feature space. (b) At the testing stage, we have added feature bias, causing 3 examples of one class to be misclassified. But, these examples can be classified correctly by constructing a similarity graph and running belief propagation on the corresponding factor graph.

when 100% of edges are present than it is when 0% of edges are present. But of course, if more-specific information is available, we will get better performance by using it to construct a more informative similarity graph with an intermediate number of edges.

One possible source of additional information is physical proximity between cells. Physical proximity is informative if we believe that nearby cells are likely to share an ancestor. However, we wanted to avoid having to simulate cell positions in our synthetic images, so in the current work we did not include edges based on physical proximity. In previous work (Chen and Murphy, 2006), we did evaluate graphs built using physical proximity, and they improved classification accuracy when applicable. So long as edges tend to connect cells that share the same class, the exact source of edges does not matter for our inference algorithms; so, the experimental results below should apply equally well to graphs that contain edges from physical proximity.

Instead, for our experiments below, we built the similarity graph according to feature-space proximity: we added edges between cells whose feature vectors were close to one another according to z-scored Euclidean distance. Using feature-space proximity in this way makes sense because minor experimental variations can perturb the features of a whole group of test cells in similar ways.

In more detail, minor differences in how cells were prepared (such as variations in plating time, exposure time, concentration of reagents, ambient temperature, etc.) can cause a noticeable bias in the computed features of a test group of cells compared to our training set. We do not generally have enough cells of enough different classes in a test group to estimate this bias accurately before classifying the cells. However, if there is a margin in feature space between classes, then building a similarity graph based on feature-space proximity is likely to connect each cell mostly to other cells

of the same class. So, as long as the feature bias is not so large that it causes most cells of a given class to be misclassified, we can hope to "rescue" cells that have been moved just across the class boundary, since they will be connected to other cells of the same class that are correctly classified. Fig. 5 justifies this intuition with a simple synthetic example.

In addition to the above reasons, feature-space proximity among test images could be informative if the training set wasn't large enough or the classifier wasn't flexible enough to learn the location classes well. In our experiments, we believe that the influence of this last effect is minimal.

To produce a variety of graphs with different edge densities, we introduced a parameter d_{cutoff}: we connected two nodes whenever their z-scored Euclidean distance in feature space was less than d_{cutoff}.[2] Large values of d_{cutoff} correspond to graphs with many edges, while small values correspond to graphs with few edges. We varied d_{cutoff} to produce graphs with edge densities ranging from 0% to 100% of the possible edges.

Constructing the Factor Graphs From each similarity graph we built several different factor graphs using different kinds of potentials. Each different factor graph corresponds to a different way to turn our qualitative similarity judgements into a precise probability distribution over label vectors.

The simplest factor graph was the Potts model. In the Potts model, we used one Potts potential for each edge in the similarity graph; each potential had parameter $\omega = 1.7$. The next type of factor graph was an associative Markov network. In this model we used one AMN potential for each node; this potential covered the node and all of its similarity-graph neighbors, and had $\omega_y = 2.9$ for all y. The last type of factor graph used the voting potential. In this graph there was one voting potential centered on each node; this potential covered the node and all of its similarity-graph neighbors, and had parameter $\lambda = 1.7$. For all three types of potential, we determined the above parameter values ahead of time by a coarse search.[3]

Synthetic Graphs Since the size of the 2D HeLa data set is limited, we performed additional experiments on automatically-generated inference problems. These experiments investigated the sensitivity of our method to the accuracy of the base classifier. We generated a synthetic graph by picking two of the ten classes at random and two numbers N_1 and N_2. We generated N_1 cells of the first class and N_2 cells of the second. For each cell we selected feature vectors of length $d = 2$ from a standard normal distribution; for one of the groups of cells we then displaced the feature vectors by a distance s. We chose $s = 3$ as a value which yielded a reasonable degree of overlap between classes. Finally, as above, we connected pairs of nodes whose feature-space distances were less than d_{cutoff}.

Synthetic Evidence To generate the evidence for a node in one of our synthetic graphs, we picked random scores for each class. The score for the true class was generated from a normal distribution

2. A reviewer of a previous version of this paper suggested that all edges should be present in the graph, and that we should adjust the weight of each edge based on the distance between the nodes. This is a reasonable suggestion; however, some of the potential functions we are evaluating do not have an obvious way to take edge weights into account. So, for the sake of an easier comparison, in this paper we work only with 0-1 weights.
3. As we examined the parameter space of the AMN potential, we discovered that results appear to be very sensitive to the strength parameter ω and the edge density of the graph. So, while we fixed a single compromise parameter for the AMN potentials in our experiments (so that our AMN results are comparable to our results for other potentials), we strongly recommend parameter learning for practical use of the AMN potential, for example, by the method described in (Taskar et al., 2004).

with mean $\mu > 0$ and unit variance, while the scores for other classes were generated from normal distributions with mean 0 and unit variance. Each score was then transformed by a sigmoid to produce the evidence for its corresponding class. Large values of μ result in a highly-accurate simulated base classifier, while small values yield a classifier that performs only a little better than chance. We show results for a range of values of μ that result in base accuracies from 50% to 90%.

9. Experimental Results

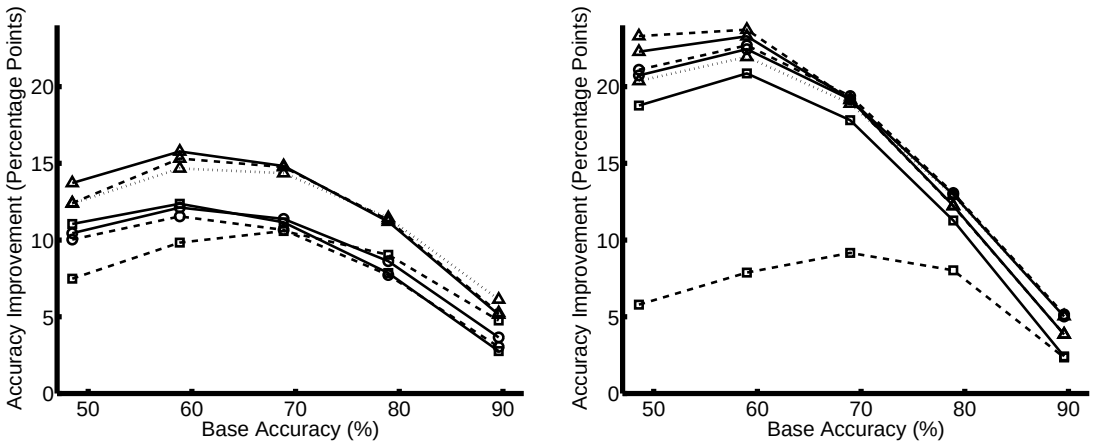
We conducted experiments to determine the effect of various potential functions and inference algorithms on overall classification accuracy in structured classification problems. In particular, we designed our experiments to compare the two-way Potts potential with the k-way AMN and voting potentials, and to compare our approximate inference algorithms to their exact counterparts. Our results support the following conclusions:

- We can achieve better classification accuracy by moving from the Potts model, with its two-way potentials, to models that contain k-way potentials for $k > 2$.
- Of the k-way potentials that we tested, the voting potential is the best for a range of problem types.
- For small networks where exact inference is feasible, our approximate inference algorithms yield results similar to exact inference at a fraction of the computational cost.
- For larger networks where exact inference becomes intractable, our approximate inference algorithms are still feasible, and structured classification with approximate inference lets us take advantage of the similarity graph to improve classification accuracy compared to the base (unstructured) classifier.

9.1 Synthetic Graphs

Our first experiment used small, synthetic graphs to compare the Potts, AMN, and voting potentials, and to compare exact and approximate inference. (Since we wanted a large number of test samples, and since we wanted to vary the accuracy of the base classifier over a wide range, it would not have been practical to conduct this experiment with the real HeLa data.) In this experiment, for each trial, we randomly generated a synthetic similarity graph with 10 nodes split between 2 classes (out of a list of 4 total classes), as described above. We used 50% graph edge density for the Potts and voting potentials and 40% graph edge density for the AMN potential, since these densities were approximately optimal according to our preliminary experiments. We also generated simulated classifier likelihoods at each node using values of μ that corresponded to base classifier accuracies ranging from 50% to 90%.

For each generated classification problem, we built three different factor graphs according to the methods described above: a Potts model, a model that used AMN potentials, and a model that used voting potentials. Finally, we computed posterior marginal class probabilities using seven different inference algorithms: exact inference on the Potts model (EIPP), loopy belief propagation on the Potts model (LBP), exact inference on the model with AMN potentials (EIAMN), loopy belief propagation on the model with AMN potentials (LBAMN), exact inference on the model with voting potentials (EIVP), loopy belief propagation on the model with voting potentials (LBVP), and



(a) (N_1,N_2)=(5,5) (b) (N_1,N_2)=(2,8)

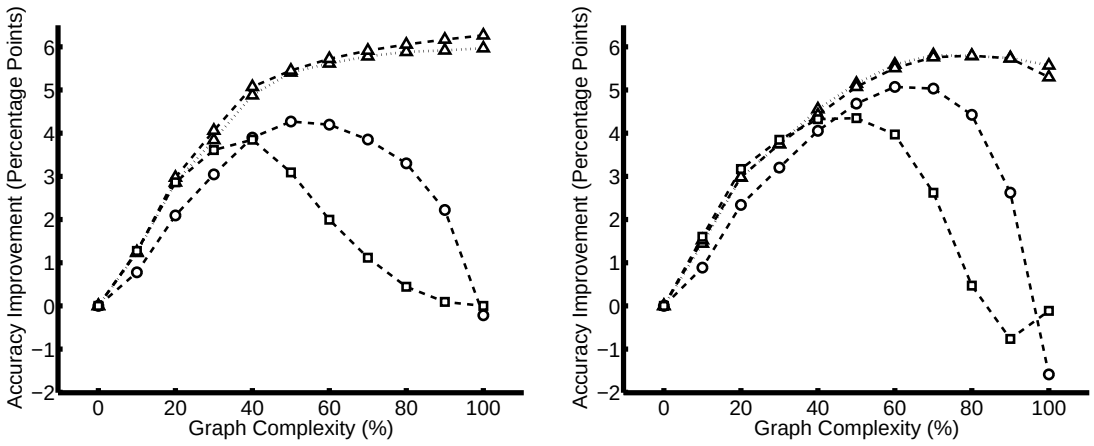
Figure 6: Accuracy Improvement vs. Base Accuracy with different inference methods on graphs with synthetic evidence: exact inference on the model with voting potentials (EIVP, $\triangle-$), loopy belief propagation on the model with voting potentials (LBVP, $\triangle--$), prior updating on the model with voting potentials (PU, $\triangle\cdots$), exact inference on the model with AMN potentials (EIAMN, $\square-$), loopy belief propagation on the model with AMN potentials (LBAMN, $\square--$), exact inference on the Potts model (EIPP, $\bigcirc-$) and loopy belief propagation on the Potts model (LBP, $\bigcirc--$). (a) Graphs with equal numbers of nodes from two classes, $(N_1, N_2) = (5,5)$. (b) Graphs with unequal numbers of nodes from two classes, $(N_1, N_2) = (2,8)$. 95% confidence bars (not shown) are smaller than the plot symbols.

prior updating on the model with voting potentials (PU). We evaluated each method by averaging its classification accuracy over all nodes in the graph, and then over 10,000 different graphs; the results are shown in Fig. 6.

As Fig. 6 illustrates, the methods using the voting potential (EIVP, LBVP, and PU) significantly outperformed the other potentials when the graph has an equal number of nodes from each class. For the unequal case, the voting potential had comparable performance to the Potts potential, and outperformed the AMN potential. In addition, the approximate methods (PU, LBVP, and LBP) did not differ substantially from their corresponding exact algorithms (EIVP, EIVP, and EIPP respectively), with the exception that LBAMN is substantially worse than EIAMN, and PU appears to be slightly worse than EIVP at lower base accuracies, and slightly better at higher base accuracies.

9.2 HeLa Data

Encouraged by the above results, we next considered classification accuracy for the real HeLa cell images. The factor graphs in this case are too large for exact inference, and therefore we cannot compare approximate and exact inference algorithms. Instead, we sought in our second experi-



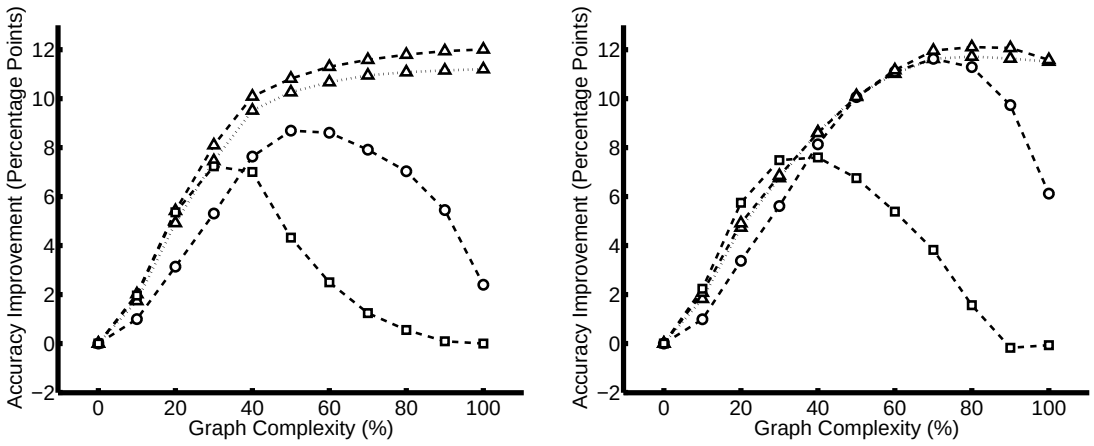
(a) (N_1,N_2)=(6,6) (b) (N_1,N_2)=(2,10)

Figure 7: Accuracy Improvement vs. Graph Complexity with different approximate inference methods on graphs with evidence from real data and base accuracy = 91.6%: loopy belief propagation on the model with voting potentials (LBVP, $\triangle--$), prior updating on the model with voting potentials (PU, $\triangle\cdots$), loopy belief propagation on the model with AMN potentials (LBAMN, $\square--$) and loopy belief propagation on the Potts model (LBP, $\bigcirc--$). (a) Graphs with equal numbers of nodes from two classes, $(N_1,N_2)=(6,6)$. (b) Graphs with unequal numbers of nodes from two classes, $(N_1,N_2)=(2,10)$. Confidence bars are not shown, since paired tests are more powerful for our experimental setup; see text for statistical comparisons.

ment to determine whether our approximate inference algorithms are able to improve classification accuracy substantially compared to our baseline SVM classifier.

For each trial in this experiment, we built a graph containing 12 cells in two classes. (We did not tell the classifier which two classes we used, so there were still 10 possible labels for each cell.) We assigned evidence to each cell using the SVM classifier described above; the base accuracy of this classifier was 91.6%. In order to evalute the inference performance on different base accuracies, we also selected the best 4 and 6 (out of 47) features to achieve the lower base accuracies of 70.6% and 83.1%, respectively. To test the algorithms' performance on a variety of problems, we adjusted d_{cutoff} to achieve levels of connectivity ranging from 0% to 100% of the possible edges; 0% graph complexity corresponds to a completely disconnected graph, in which each node's class is assigned using just the base classifier, while 100% means that the graph is fully connected.

We evaluated each algorithm's accuracy by 6-fold cross-validation: we trained the SVM using 5/6 of the images, used the other 1/6 of the images to construct testing networks, and recorded the improvement in classification accuracy compared to the base classifier in each testing network. We repeated this procedure 6 times so that every image appeared in the test partition once. Finally, we averaged the overall accuracy over 10 different splits into folds. The results in Fig. 7, Fig. 8, and Fig. 9 demonstrate that PU and LBVP can robustly achieve a good improvement in classification



(a) (N_1,N_2)=(6,6) (b) (N_1,N_2)=(2,10)

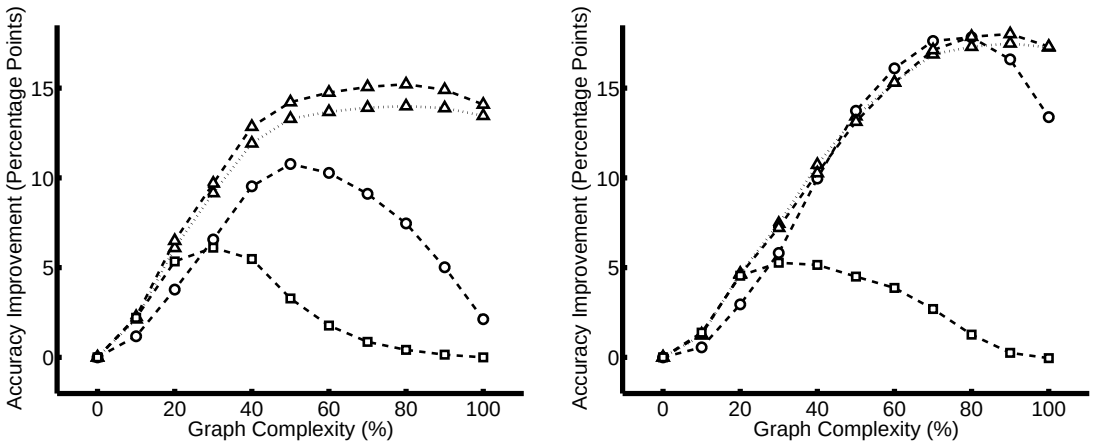
Figure 8: Accuracy Improvement vs. Graph Complexity with different approximate inference methods on graphs with evidence from real data and base accuracy = 83.1%: loopy belief propagation with voting potentials (LBVP, $\triangle --$), prior updating with voting potentials (PU, $\triangle \cdots$), loopy belief propagation with AMN potentials (LBAMN, $\square --$) and loopy belief propagation on the Potts model (LBP, $\bigcirc --$). (a) Graphs with equal numbers of nodes from two classes, $(N_1, N_2) = (6, 6)$. (b) Graphs with unequal numbers of nodes from two classes, $(N_1, N_2) = (2, 10)$. Confidence bars are not shown, since paired tests are more powerful for our experimental setup; see text for statistical comparisons.

accuracy on graphs with equal numbers of nodes from two classes, as compared to methods based on other potentials. (One-tailed paired t-test: $p = 0.0033, p = 0.0018, p = 0.0001$ for graphs with 91.6%, 83.1%, and 70.6% base accuracies, respectively. Tests were conducted at graph complexity 50%; p values are for comparison to closest competitor.)

On graphs with unequal class sizes, voting and Potts potentials achieved statistically comparable results when the graph complexity was tuned optimally. But, the performance of the voting-potential-based methods was more robust: their accuracy remained high for a wider range of graph complexities. (For example, one-tailed paired t-test for LBVP versus LBP: $p = 0.0036$ for graphs with 91.6% base accuracies at graph complexity 80%; $p = 0.0008$ for graphs with 83.1% base accuracies at graph complexity 90%; $p \leq 0.0001$ for graphs with 70.6% base accuracies at graph complexity 100%.)

9.3 Inference Speed

Our final experiment compares the computational efficiency of the different inference methods. Each point in Fig. 10 shows the average inference time per trial on different sizes of graphs with various algorithms (note the logarithmic time scale). Each graph has $N_1 = N_2$, uses four of the ten classes from the HeLa data set, and has 50% graph complexity. In this figure, we also included the processing time for loopy belief propagation with voting potentials using naive message com-



(a) (N_1,N_2)=(6,6) (b) (N_1,N_2)=(2,10)

Figure 9: Accuracy Improvement vs. Graph Complexity with different approximate inference methods on graphs with evidence from real data and base accuracy = 70.6%: loopy belief propagation on the model with voting potentials (LBVP, $\triangle--$), prior updating on the model with voting potentials (PU, $\triangle\cdots$), loopy belief propagation on the model with AMN potentials (LBAMN, $\square--$) and loopy belief propagation on the Potts model (LBP, $\bigcirc--$). (a) Graphs with equal numbers of nodes from two classes, $(N_1, N_2) = (6, 6)$. (b) Graphs with unequal numbers of nodes from two classes, $(N_1, N_2) = (2, 10)$. Confidence bars are not shown, since paired tests are more powerful for our experimental setup; see text for statistical comparisons.

putation (LBVPNC), which computes messages using ordinary marginalization rather than our new message computation algorithms. LBVPNC is expected to be faster then EIVP when the number of neighbors of one node is only a small fraction of the total number of nodes; otherwise LBVPNC could be much slower than EIVP, as is the case in this experiment.

The exact inference methods take time exponential in the size of the graph and are impractical to run for graphs of more than 12 nodes, while the processing times of PU, LBVP and LBP are much faster, and remain well under a second for the largest graphs tested. (With more than four classes, the exact inference times would rise even more quickly, and the graphs would have to be even smaller to allow exact inference.) The processing times for approximate inference methods are a combination of two factors: the number of iterations to converge and the time for each iteration. Fig. 11 shows the number of iterations needed for each method.

10. Related Work

The problem of how to quickly compute belief messages (or other similar quantities in an inference algorithm) is an important one, and several special cases of our decomposable structure have been previously described in the literature. One recent example is an algorithm for fast inference in

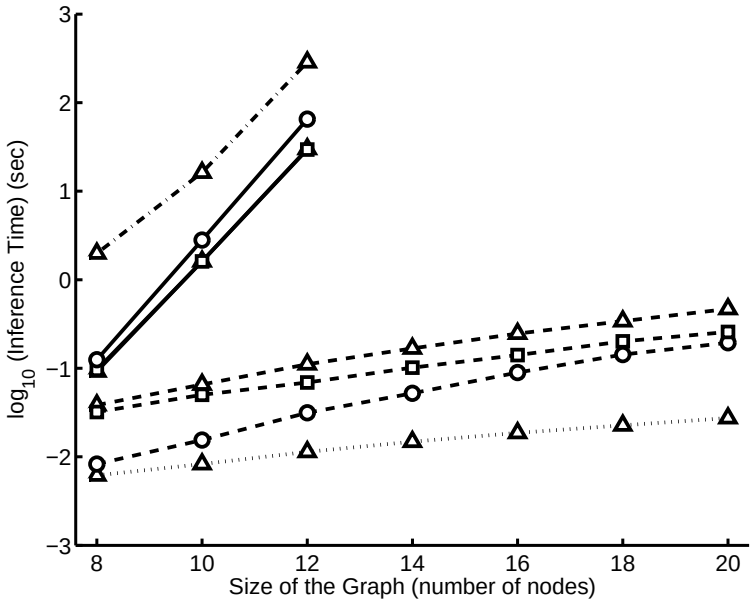


Figure 10: Dependence of inference time on graph size for different inference methods: exact inference on the model with voting potentials (EIVP, $\triangle-$), loopy belief propagation on the model with voting potentials (LBVP, $\triangle--$), loopy belief propagation on the model with voting potentials using naive message computation (LBVPNC, $\triangle\cdot-$), prior updating on the model with voting potentials (PU, $\triangle\cdots$), exact inference on the model with AMN potentials (EIAMN, $\square-$), loopy belief propagation on the model with AMN potentials (LBAMN, $\square--$), exact inference on the Potts model (EIPP, $\bigcirc-$) and loopy belief propagation on the Potts model (LBP, $\bigcirc--$). Values shown are times for one run of inference on the given graph, averaged over 12 trials.

hidden Markov models with structured transition matrices (Siddiqi and Moore, 2005). HMMs are a special case of chain-shaped factor graphs: there is one node representing the state x_t at each time step t, and there is a factor connecting the nodes x_t and x_{t+1} for each t. The potential functions for all factors are the same, and are equal to the transition matrix: $\varphi(x_t, x_{t+1}) = P(x_{t+1} \mid x_t)$.

The special structure for transition matrices proposed by Sidiqqi and Moore is called "Dense Mostly Constant," or DMC. In a DMC transition matrix for n states, each row contains k arbitrary entries in specified positions, and the other $n-k$ entries are all equal to a shared constant. The k arbitrary entries may be in different positions for each row, and the shared constant may also differ from row to row. Siddiqi and Moore show how to run the forward-backward, Viterbi, and Baum-Welsh algorithms quickly for HMMs with DMC transition matrices.

The DMC structure is a special case of our decomposable potential structure: any potential function corresponding to a DMC transition matrix can be written as a rank-one term plus a sparse term. More precisely,

$$\phi(x_t, x_{t+1}) = q(x_t) + \sum_{x,x' \in S} (w_{xx'} - q(x)) I(x_t, x) I(x_{t+1}, x').$$

Here $q(x_t)$ is the shared constant for the row corresponding to x_t, S is the set of transition matrix entries (x,x') which are allowed to differ from the shared constants, and $w_{xx'}$ is the value of the transition matrix entry at position (x,x'). The functions $q(x_t)$, $I(x_t,x)$, and $I(x_{t+1},x')$ each have arity 1; and, the set S is sparse, since it has at most $kn \ll n^2$ elements. Since DMC potentials are decomposable, our message calculation algorithm allows us to run belief propagation quickly; doing so is essentially equivalent to Sidiqqi and Moore's implementation of the forward-backward algorithm.

As we have pointed out above, the associative Markov network potential is also an example of our decomposable potential structure. The original paper on AMNs used an inference algorithm based on linear programming rather than on belief propagation (Taskar et al., 2004). However, the reason that their LP-based inference algorithm is tractable is exactly that they can take advantage of the AMN potential's special structure. There does not appear to be a simple way to extend their LP-based inference algorithm to the more general decomposable potential structure studied here, but this would be an interesting direction for future work. Another interesting direction for future work would be to extend their parameter-learning algorithm to handle more general structures like the decomposable potentials studied here.

In the context of a computer vision application, Felzenszwalb and Huttenlocher (2004) describe how to run loopy belief propagation quickly for a number of different pairwise potential functions. One that they consider is the Potts potential, and their algorithm for handling this potential takes advantage of the fact that it is a constant plus a sparse matrix, $1+(\omega-1)I$. They also consider pairwise potentials based on distance functions, which do not in general appear to be examples of our class of decomposable potential functions. On the other hand, they consider only potentials with up to two arguments.

In addition to the potentials studied in this paper, multiple examples of specific decomposable potentials exist in the literature. One of the most common forms is a potential used in directed graphical models (Bayes nets), in which the child's distribution depends on the sum of the parents' values (e.g., Frey and Kannan, 2000). Another good example is a potential used to represent probability distributions over graphs, in which a structure's probability depends on the degree of its nodes (Morris et al., 2003). All of these examples share a certain similarity to several of the special cases of decomposable potentials mentioned above, in that the speedup comes from accelerating the calculation of a single large sum within the message computation.

Another way to handle large sums is to build a sum tree: for example, the sum $x_1+x_2+x_3+x_4$ can be rewritten $[(x_1+x_2)+(x_3+x_4)]$. So, a potential with 4 arguments, $\varphi(x_1,x_2,x_3,x_4) = f(x_1+x_2+x_3+x_4)$, can be replaced by three smaller potentials and two new variables:

$$I(y_1 = x_1+x_2)\, I(y_2 = x_3+x_4)\, f(y_1+y_2).$$

Similarly, a potential that depends on a sum of n terms can be replaced by $n-1$ three-argument potentials and $n-2$ additional variables. See Liao et al. (2006) for an example of an algorithm based on this intuition. While this method works well for potentials that are functions of sums, it is not straightforward to extend it to more complicated potentials of the form we consider in this paper.

A final, and related, example of a tractable approximation to belief propagation is given by Barber (2001). Barber considers the directed version of the belief propagation algorithm, with conditional probability distribution functions of the form

$$P(x \mid \mathbf{s}) = f(\theta \cdot \mathbf{s}).$$

Here x is a node in the graph, $\mathbf{s}$ is the vector of parents of x, θ is a vector of fixed parameters, and f is a one-dimensional function with a tractable (approximate) Fourier expansion. This class of potentials is different from ours; perhaps an even larger class of potentials could be handled by combining the two techniques, arriving at a class of potentials that looked like

$$\varphi_j(x_1,\ldots,x_k) = \sum_i f_i\left(\sum_{l=1}^{L_j}\prod_{m=1}^{M_{jl}}\xi_{jlm}(x_{V(j,l,m)})\right)$$

where f_i is a basis function (e.g., a multiple of a sine wave if we are using Fourier expansions as above).

It is possible to find the most likely vector of labels for a Potts model using graph cuts when each variable x_i is binary. A related algorithm can be used for approximate inference with non-binary variables, and always finds a label vector within a factor of 2 of the most likely. (See Kleinberg and Tardos, 1999; Taskar et al., 2004, for a discussion.) It does not appear to be easily possible to extend this group of algorithms to k-way potentials such as the voting potential.

In our own previous work, we applied PU to image segmentation problems to identify cell regions (Chen et al., 2006b). In this case, the voting potential assumed that two sources of information were available: images of nuclear staining and of cell boundary locations, both of which were expected to be noisy. The nuclear staining provided an initial assignment of whether a pixel belongs to the background or to one of the cells, while the cell boundary image provided a probability estimate of whether two neighboring pixels should have the same label. The results indicated that PU provided efficient and accurate inference.

11. Conclusions and Future Work

We have examined the problem of classifying multiple dependent examples in a protein subcellular location pattern recognition task. We have compared several different graphical models and inference algorithms designed to solve this sort of structured classification problem, including one based on the novel voting potential function. The voting potential, like the previously studied Potts and AMN potentials, encodes the intuition that a variable is likely to have the same class as its neighbors. Our experiments show that the voting potential often does a better job of encoding this intuition than the Potts and AMN potentials do.

In addition to the new potential, we have presented new approximate inference algorithms: first, we have shown how to implement loopy belief propagation efficiently for networks with decomposable potentials, including the AMN and voting potentials. And second, we have suggested ignoring certain belief messages during LBP for the voting potential, resulting in an algorithm called Prior Updating. These fast algorithms enable us to use potentials like the voting potential on real data, where they would otherwise be impractical.

We believe that the voting potential function and the class of decomposable potentials will generalize to other graphs and other applications: for example, in image segmentation, it is common to use a potential which encourages nearby pixels to be part of the same object. With a potential similar to the voting potential described here, we could allow many neighboring pixels to vote on which object a particular pixel belongs to; and in fact, we have conducted initial experiments in this direction (Chen et al., 2006b). For another example, in regression and classification it is common to reject outliers to improve the robustness of the learned concept. In a given training set, multiple outliers may result from similar causes, such as being out of focus or in a different phase of the

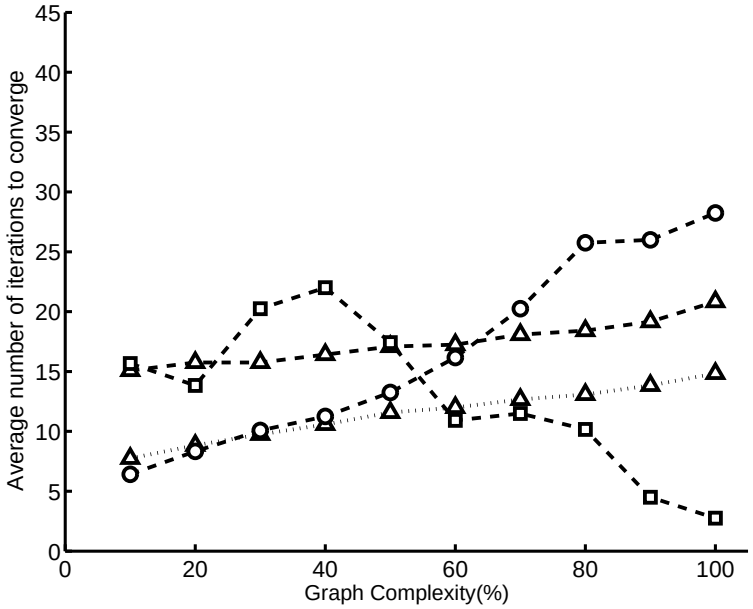


Figure 11: Dependence of number of iterations to converge on graph size for different inference methods: loopy belief propagation on the model with voting potentials (LBVP, $\triangle--$), prior updating on the model with voting potentials (PU, $\triangle\cdots$), loopy belief propagation on the model with AMN potentials (LBAMN, $\square--$) and loopy belief propagation on the Potts model (LBP, $\bigcirc--$). Values shown are numbers of iterations for one run of inference to converge on the given graph, averaged over 12 trials.

cell cycle. By using feature-space similarity to connect training examples in a graph, we can infer which types of outliers are present in the data and use this information to determine more accurately whether each point is an outlier.

In addition to the specific potentials we propose, our algorithms for calculating belief messages efficiently will also generalize to other domains. Whenever a graph contains a factor with a decomposable potential function, we can greatly reduce the time required to calculate belief messages from that factor. Our message calculation algorithm is exact; one might expect that further speed improvements would be possible if we are willing to accept approximate calculations. One promising avenue that we intend to explore is a "loopy" version of our message calculation algorithm. In such an algorithm, an inner loop of message passing would approximate the belief messages needed by the outer loop.

Acknowledgments

This work was supported in part by NSF grant EF-0331657. Facilities and infrastructure support were provided by NIH National Technology Center for Networks and Pathways grant U54 RR022241 and by NIH National Center for Biomedical Computing grant National U54 DA021519.

References

D. Barber. Tractable approximate belief propagation. In *Advanced Mean Field Methods: Theory and Practice*, pages 197–212. MIT Press, 2001.

M. V. Boland and R. F. Murphy. A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of HeLa cells. *Bioinformatics*, 17:1213–1223, 2001.

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`.

S.-C. Chen and R. F. Murphy. A graphical model approach to automated classification of protein subcellular location patterns in multi-cell images. *BMC Bioinformatics*, 7:90, 2006.

S.-C. Chen, G. J. Gordon, and R. F. Murphy. A novel approximate inference approach to automated classification of protein subcellular location patterns in multi-cell images. In *Proceedings of the 2006 IEEE International Symposium on Biomedical Imaging (ISBI)*, pages 558–561, 2006a.

S.-C. Chen, T. Zhao, G. J., Gordon, and R. F. Murphy. A novel graphical model approach to segmenting cell images. In *Proceedings of the 2006 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–8, 2006b.

X. Chen, M. Velliste, and R. F. Murphy. Automated interpretation of subcellular patterns in fluorescence microscope images for location proteomics. *Cytometry*, 69A:631–640, 2006c.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 261–268, 2004.

B. J. Frey and A. Kannan. Accumulator networks: Suitors of local probability propagation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 486–492, 2000.

E. Glory and R. F. Murphy. Automated subcellular location determination and high throughput microscopy. *Developmental Cell*, 12:7–16, 2007.

T. Heskes. On the uniqueness of loopy belief propagation fixed points. *Neural Computation*, 16: 2379–2413, 2004.

K. Huang and R. F. Murphy. Boosting accuracy of automated classification of fluorescence microscope images for location proteomics. *BMC Bioinformatics*, 5:78, 2004.

A. T. Ihler, J. W. Fisher, III, and A. S. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6:905–936, 2005.

U. Kjærulff. Inference in Bayesian networks using nested junction trees. In *Learning in Graphical Models*, pages 51–74. Kluwer Academic Press, 1998.

J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 14–23, 1999.

D. Koller and N. Friedman. *Bayesian Networks and Beyond*. 2007. Draft manuscript.

F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

L. Liao, D. Fox, and H. Kautz. Location-based activity recognition. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 18, pages 787–794, Cambridge, MA, 2006. MIT Press.

H.-T. Lin, C.-J. Lin, and R. C. Weng. *A Note on Platt's Probabilistic Outputs for Support Vector Machines*, 2003. Technical report, Department of Computer Science, National Taiwan University.

R. McEliece, D. MacKay, and J. Cheng. Turbo decoding as an instance of Pearl's belief propagation algorithm. *IEEE Journal on Selected Areas in Communications*, 16:140–152, 1998.

Q. D. Morris, B. J. Frey, and C. J. Paige. Denoising and untangling graphs using degree priors. In *Neural Information Processing Systems Conference (NIPS)*, pages 385–392, 2003.

K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference—an empirical study. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 467–475, 1999.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, 1988.

J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 2000.

S. Siddiqi and A. Moore. Fast inference and learning in large-state-space HMMs. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 800–807, 2005.

B. Taskar, V. Chatalbashev, and D. Koller. Learning associative Markov networks. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pages 102–109, 2004.

S. C. Tatikonda and M. I. Jordan. Loopy belief propagation and Gibbs measures. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 493–500, 2002.

Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12:1–41, 2000.

D.-Q. Zhang and S.-F. Chang. Learning to detect scene text using a higher-order MRF with belief propagation. *IEEE Workshop on Learning in Computer Vision and Pattern Recognition, in conjunction with CVPR (LCVPR)*, 6:101–108, 2004.

Journal of Machine Learning Research 9 (2008) 683-718 Submitted 4/07; Revised 11/07; Published 4/08

Learning Control Knowledge for Forward Search Planning

Sungwook Yoon SUNGWOOK.YOON@ASU.EDU
Computer Science and Engineering Department
Arizona State University
Tempe, AZ 85281

Alan Fern AFERN@EECS.ORST.EDU
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR 97330

Robert Givan GIVAN@PURDUE.EDU
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907

Editor: Carlos Guestrin

Abstract

A number of today's state-of-the-art planners are based on forward state-space search. The impressive performance can be attributed to progress in computing domain independent heuristics that perform well across many domains. However, it is easy to find domains where such heuristics provide poor guidance, leading to planning failure. Motivated by such failures, the focus of this paper is to investigate mechanisms for learning domain-specific knowledge to better control forward search in a given domain. While there has been a large body of work on inductive learning of control knowledge for AI planning, there is a void of work aimed at forward-state-space search. One reason for this may be that it is challenging to specify a knowledge representation for compactly representing important concepts across a wide range of domains. One of the main contributions of this work is to introduce a novel feature space for representing such control knowledge. The key idea is to define features in terms of information computed via relaxed plan extraction, which has been a major source of success for non-learning planners. This gives a new way of leveraging relaxed planning techniques in the context of learning. Using this feature space, we describe three forms of control knowledge—reactive policies (decision list rules and measures of progress) and linear heuristics—and show how to learn them and incorporate them into forward state-space search. Our empirical results show that our approaches are able to surpass state-of-the-art non-learning planners across a wide range of planning competition domains.

Keywords: planning, machine learning, knowledge representation, search

1. Introduction

In recent years, forward state-space search has become a popular approach in AI planning, leading to state-of-the-art performance in a variety of settings, including classical STRIPS planning (Hoffmann and Nebel, 2001; Vidal, 2004), optimal planning (Helmert et al., 2007), temporal-metric planning (Do and Kambhampati, 2003), nondeterministic planning (Bryce and Kambhampati, 2006), and oversubscribed planning (Benton et al., 2006), among others. Given that forward state-space search

is one of the oldest techniques available for planning, and many other search spaces and approaches have been developed, this state-of-the-art performance is somewhat surprising. One of the key reasons for the success is the development of powerful domain-independent heuristics that work well on many AI planning domains. Nevertheless, it is not hard to find domains where these heuristics do not work well, resulting in planning failure. We are motivated by these failures and in this study, we investigate machine learning techniques that find domain-specific control knowledge that can improve or speed-up forward state-space search in a non-optimal, or satisficing, planning setting.

As outlined in Section 3 there is a large body of work on learning search-control knowledge for AI planning domain. However, despite the significant effort, none of these approaches has been demonstrated to be competitive with state-of-the-art non-learning planners across a wide range of planning domains. There are at least two reasons for the performance gap between learning and non-learning planners. First, most prior work on learning control knowledge has been in the context of non-state-of-the-art planning approaches such as partial-order planning, means-ends analysis, among others. In fact, we are only aware of two recent efforts (Botea et al., 2005; Coles and Smith, 2007) that learn control knowledge for forward state-space search planners. Even these approaches have not demonstrated the ability to outperform the best non-learning planners as measured on planning competition domains. Second, it is a challenge to define a hypothesis space for representing control knowledge that is both rich enough for a wide variety of planning domains, yet compact enough to support efficient and reliable learning. Indeed, a common shortcoming of much of the prior work is that the hypothesis spaces, while adequate for the small number of domains investigated, were not rich enough for many other domains.

The primary goal of this work is to contribute toward reversing the performance gap between learning and non-learning planners. In this work, we do this by addressing each of the above two issues. First, our system is based on the framework of forward state-space search, in particular, being built upon the state-of-the-art planner FF (Hoffmann and Nebel, 2001). Second, we propose a novel hypothesis space for representing useful heuristic features of planning states. We show how to use this feature space as a basis for defining and learning several forms of control knowledge that can be incorporated into forward state-space search. The result is a learning-based planner that learns control knowledge for a planning domain from a small number of solved problems and is competitive with and often better than state-of-the-art non-learning planners across a substantial set of benchmark domains and problems.

A key novelty of our proposed feature space is that it leverages the computation of relaxed plans, which are at the core of the computation of modern forward-search heuristics (Bonet and Geffner, 2001; Hoffmann and Nebel, 2001). Relaxed plans are constructed by ignoring, to varying degrees, the delete/negative effects of actions and can be computed very efficiently. The length of these plans can then serve as an informative heuristic (typically non-admissible) for guiding state-space search. In addition to their length, relaxed plans contain much more information about a search state that is ignored by most forward-search planners.[1] Our proposed feature space gives one way of using this information by viewing the relaxed plan as a structure for defining potentially useful features of the current state. As an example of the utility of our feature space, note that the fact that relaxed planning ignores delete effects is the main reason that the length sometimes dramatically underestimates the true distance to goal, leading to poor heuristic guidance (note that relaxed-plan length can also overestimate the distance to goal). Our feature space is able to partially

1. One exception is Vidal (2004) where relaxed plans are also used to extract macro actions.

capture information about the delete effects ignored in a relaxed plan, which can be used to learn knowledge that partially compensates for the underestimation.

We use the relaxed-plan feature space to learn two forms of knowledge for controlling forward state-space planning. In each case, the knowledge is learned based on a set of training problems from a domain, each labeled by a solution. First, we consider learning knowledge in the form of linear heuristic functions. In particular, we learn heuristics that are linear combinations of relaxed-plan features, with one of those features being the relaxed-plan length. Thus, our heuristic learner can be viewed as an approach for automatically correcting deficiencies of the usual relaxed-plan length heuristic, by augmenting it with a weighted combination of additional features selected from the large space of possible relaxed-plan features.

As a second form of control knowledge, we investigate reactive policies. Learning reactive policies for planning domains has been studied by several researchers (Khardon, 1999; Martin and Geffner, 2000; Yoon et al., 2002, 2005). However, all of these studies have used the learned policies as stand-alone search-free planners that simply execute the linear sequence of actions selected by the policies. While this non-search approach is efficient and has been shown to work well in a number of domains, it often fails due to flaws in the policy that arise due to imperfect learning. Nevertheless, such policies capture substantial information about the planning domain, which we would like to exploit in a more robust way. In this work, we propose and evaluate a simple way of doing this by integrating learned policies into forward search. At each search node (i.e., state), we execute the learned policy for a fixed horizon and add all of the states encountered to the search queue. In this way, flaws in the policy can be overcome by search, while the search efficiency can be substantially improved by quickly uncovering good "deep states" that are found by the policy. We evaluate this idea using two representations for learned policies both of which make use of relaxed-plan features—decision lists of rules similar to Yoon et al. (2002) and measures of progress (Yoon et al., 2005)—both of which make use of relaxed-plan features.

In our experiments, we learned and evaluated both forms of control knowledge on benchmark problems from recent planning competitions. We learned on the first 15 problems in each domain and tested on the remaining problems. The results are very much positive. Forward state-space search with the learned control knowledge outperforms state-of-the-art planners, in most of the competition domains. We also demonstrate the utility of our relaxed-plan feature space by considering feature spaces that ignore parts of the relaxed-plan information, showing that using the relaxed-plan information leads to the best performance.

The remainder of the paper is structured as follows. In Section 2, we introduce the problem of learning domain-specific control knowledge for planning and in Section 3 we overview some of the prior work in this area. In Section 4, we describe the two general forms of control knowledge, heuristics and reactive policies, that we consider learning in this work and how we will use that knowledge to guide forward state-space search. In Section 5, we will describe a relaxed-plan feature space that will serve as our basis for representing both learned heuristics and policies. In Sections 6 and 7 we will describe specific representations for policies and heuristics, in terms of the relaxed-plan features, and give learning algorithms for these forms of control knowledge. In Section 8, we demonstrate the effectiveness of our proposal in this study through empirical results on benchmark planning domains. In Section 9, we summarize and discuss potential future extensions to this work.

2. Problem Setup

In this paper, we focus our attention on learning control knowledge for deterministic STRIPS planning domains. Below we first give a formal definition of the types of planning domains we consider and then describe the learning problem.

2.1 Planning Domains

A deterministic planning domain $\mathcal{D}$ defines a set of possible actions $\mathcal{A}$ and a set of states $\mathcal{S}$ in terms of a set of predicate symbols $\mathcal{P}$, action types Y, and objects O. Each state in $\mathcal{S}$ is a set of facts, where a fact is an application of a predicate symbol in $\mathcal{P}$ to the appropriate number of objects from O. There is an action in $\mathcal{A}$, for each way of applying the appropriate number of objects in O to an action type symbol in Y. Each action $a \in \mathcal{A}$ consists of: 1) an action name, which is an action type applied to the appropriate number of objects, 2) a set of precondition state facts $\text{Pre}(a)$, 3) two sets of state facts $\text{Add}(a)$ and $\text{Del}(a)$ representing the add and delete effects respectively. As usual, an action a is applicable to a state s iff $\text{Pre}(a) \subseteq s$, and the application of an (applicable) action a to s, denoted $a(s)$, results in the new state $a(s) = (s \setminus \text{Del}(a)) \cup \text{Add}(a)$.

Given a planning domain, a planning problem $\mathbb{P}$ from the domain is a tuple (s, A, g), where $A \subseteq \mathcal{A}$ is a set of applicable actions, $s \in \mathcal{S}$ is the initial state, and g is a set of state facts representing the goal. A solution plan for a planning problem is a sequence of actions $(a_1, \ldots, a_h)$, where the sequential application of the sequence starting in state s leads to a goal state s' where $g \subseteq s'$. Later in the paper, in Section 7, when discussing measures of progress, it will be useful to talk about reachability and deadlocks. We say that a planning problem (s, A, g) is reachable from problem (s_0, A, g) iff there is some action sequence in A^* that leads from s_0 to s. We say that a planning problem $\mathbb{P}$ is deadlock free iff all problems reachable from $\mathbb{P}$ are solvable.

2.2 Learning Control Knowledge from Solved Problems

The bi-annual International Planning Competition (IPC), has played a large role in the recent progress observed in AI planning. Typically the competition is organized around a set of planning domains, with each domain providing a sequence of planning problems, often in increasing order of difficulty. Despite the fact that the planners in these competitions experience many similar problems from the same domain, to our knowledge only one of them, Macro-FF (Botea et al., 2005), has made any attempt to learn from previous experience in a domain to improve performance on later problems.[2] Rather they solve each problem as if it were the first time the domain had been encountered. The ability to effectively transfer domain experience from one problem to the next would provide a tremendous advantage. Indeed, the potential benefit of learning domain-specific control knowledge can be seen by the impressive performance of planners such as TL Plan (Bacchus and Kabanza, 2000) and SHOP (Nau et al., 1999), where human-written control knowledge is provided for each domain. However, to date, most "learning to plan" systems have lagged behind the state-of-the-art non-learning domain-independent planners. One of the motivations and contributions of this work is to move toward reversing that trend.

The input to our learner will be a set of problems for a particular planning domain along with a solution plan to each problem. The solution plan might have been provided by a human or automat-

2. Macro-FF learned on training problems from each domain provided by the organizers before the competition, rather than learning during the actual competition itself.

ically computed using a domain-independent planner (for modestly sized problems). The goal is to analyze the training set to extract control knowledge that can be used to more effectively solve new problems from the domain. Ideally, the control knowledge allows for the solutions of large, difficult problems that could not be solved within a reasonable time limit before learning.

As a concrete example of this learning setup, in our experiments, we use the problem set from recent competition domains. We first use a domain-independent planner, in our case FF (Hoffmann and Nebel, 2001), to solve the low-numbered planning problems in the set (typically corresponding to the easier problems). The solutions are then used by our learner to induce control knowledge. The control knowledge is then used to solve the remaining, typically more difficult, problems in the set. Our objective here is to obtain fast, satisficing planning through learning. The whole process is domain independent, with knowledge transferring from easy to hard problems in a domain. Note that although learning times can be substantial, the learning cost can be amortized over all future problems encountered in the domain.

3. Prior Work

There has been a long history of work on learning-to-plan, originating at least back to the original STRIPS planner (Fikes et al., 1972), which learned triangle tables or macros that could later be exploited by the planner. For a collection and survey of work on learning in AI planning see Minton (1993) and Zimmerman and Kambhampati (2003).

A number of learning-to-plan systems have been based on the explanation-based learning (EBL) paradigm, for example, Minton et al. (1989) among many others. EBL is a deductive learning approach, in the sense that the learned knowledge is provably correct. Despite the relatively large effort invested in EBL research, the best approaches typically did not consistently lead to significant gains, and even hurt performance in many cases. A primary way that EBL can hurt performance is by learning too many, overly specific control rules, which results in the planner spending too much time simply evaluating the rules at the cost of reducing the number of search nodes considered. This problem is commonly referred to as the EBL utility problem (Minton, 1988).

Partly in response to the difficulties associated with EBL-based approaches, there have been a number of systems based on inductive learning, perhaps combined with EBL. The inductive approach involves applying statistical learning mechanisms in order to find common patterns that can distinguish between good and bad search decisions. Unlike EBL, the learned control knowledge does not have guarantees of correctness, however, the knowledge is typically more general and hence more effective in practice. Some representative examples of such systems include learning for partial-order planning (Estlin and Mooney, 1996), learning for planning as satisfiability (Huang et al., 2000), and learning for the Prodigy means-ends framework (Aler et al., 2002). While these systems typically showed better scalability than their EBL counterparts, the evaluations were typically conducted on only a small number of planning domains and/or small number of test problems. There is no empirical evidence that such systems are robust enough to compete against state-of-the-art non-learning planners across a wide range of domains.

More recently there have been several learning-to-plan systems based on the idea of learning reactive policies for planning domains (Khardon, 1999; Martin and Geffner, 2000; Yoon et al., 2002). These approaches use statistical learning techniques to learn policies, or functions, that map any state-goal pair from a given domain to an appropriate action. Given a good reactive policy for a domain, problems can be solved quickly, without search, by iterative application of the policy. Despite

its simplicity, this approach has demonstrated considerable success. However, these approaches have still not demonstrated the robustness necessary to outperform state-of-the-art non-learning planners across a wide range of domains.

Ideas from reinforcement learning have also been applied to learn control policies in AI planning domains. Relational reinforcement learning (RRL) (Dzeroski et al., 2001), used Q-learning with a relational function approximator, and demonstrated good empirical results in the Blocksworld. The Blocksworld problems they considered were complex from a traditional RL perspective due to the large state and action spaces, however, they were relatively simple from an AI planning perspective. This approach has not yet shown scalability to the large problems routinely tackled by today's planners. A related approach, used a more powerful form of reinforcement learning, known as approximate policy iteration, and demonstrated good results in a number of planning competition domains (Fern et al., 2006). Still the approach failed badly on a number of domains and overall does not yet appear to be competitive with state-of-the-art planners on a full set of competition benchmarks.

The most closely related approaches to ours are recent systems for learning in the context of forward state-space search. Macro-FF (Botea et al., 2005) and Marvin (Coles and Smith, 2007) learn macro action sequences that can then be used during forward search. Macro-FF learns macros from a set of training problems and then applies them to new problems. Rather, Marvin is an online learner in the sense that it acquires macros during search in a specific problem that are applied at later stages in the search. As evidenced in the recent planning competitions, however, neither system dominates the best non-learning planners.

Finally, we note that researchers have also investigated domain-analysis techniques, for example, Gerevini and Schubert (2000) and Fox and Long (1998), which attempt to uncover structure in the domain by analyzing the domain definition. These approaches have not yet demonstrated the ability to improve planning performance across a range of domains.

4. Control Knowledge for Forward State-Space Search

In this section, we describe the two general forms of control knowledge that we will study in this work: heuristic functions and reactive policies. For each, we describe how we will incorporate them into forward state-space search in order to improve planning performance. Later in the paper, in Sections 6 and 7, we will describe specific representations for heuristics and policies and give algorithms for learning them from training data.

4.1 Heuristic Functions

The first and most traditional forms of control knowledge we consider are heuristic functions. A heuristic function $H(s,A,g)$ is simply a function of a state s, action set A, and goal g that estimates the cost of achieving the goal from s using actions in A. If a heuristic is accurate enough, then greedy application of the heuristic will find the goal without search. However, when a heuristic is less accurate, it must be used in the context of a search procedure such as best-first search, where the accuracy of the heuristic impacts the search efficiency. In our experiments, we will use best-first search, which has often been demonstrated to be an effective, though sub-optimal, search strategy in forward state-space planning. Note that by best-first search, here we mean a search that is guided by only the heuristic value, rather than the path-cost plus heuristic value. This search is also called *greedy* best-first search. In this paper, when we use the term best-first search, it means greedy best-

first search and we will add *greedy* in front of best-first search to remind the readers, as necessary, in the following texts.

Recent progress in the development of domain-independent heuristic functions for planning has led to a new generation of state-of-the-art planners based on forward state-space heuristic search (Bonet and Geffner, 2001; Hoffmann and Nebel, 2001; Nguyen et al., 2002). However, in many domains these heuristics can still have low accuracy, for example, significantly underestimating the distance to goal, resulting in poor guidance during search. In this study, we will attempt to find regular pattern of heuristic inaccuracy (either due to over or under estimation) through machine learning and compensate the heuristic function accordingly.

We will focus our attention on linear heuristics that are represented as weighted linear combinations of features, that is, $H(s,A,g) = \Sigma_i w_i \cdot f_i(s,A,g)$, where the w_i are weights and the f_i are functions. In particular, for each domain we would like to learn a distinct set of features and their corresponding weights that lead to good planning performance in that domain. Note that some of the feature functions can correspond to existing domain-independent heuristics, allowing for our learned heuristics to exploit the useful information they already provide, while overcoming deficiencies by including additional features. The representation that we use for features is discussed in Section 5 and our approach to learning linear heuristics over those features is described in Section 6.

In all of our experiments, we use the learned heuristics to guide (greedy) best-first search when solving new problems.

4.2 Reactive Policies in Forward Search

The second general form of control knowledge that we consider in this study is of reactive policies. A reactive policy is a computationally efficient function $\pi(s,A,g)$, possibly stochastic, that maps a planning problem (s,A,g) to an action in A. Given an initial problem (s_0,A,g), we can use a reactive policy π to generate a *trajectory* of pairs of problems and actions $(((s_0,A,g),a_0),((s_1,A,g),a_1),((s_2,A,g),a_2)\ldots)$, where $a_i = \pi(s_i,A,g)$ and $s_{i+1} = a_i(s_i)$. Ideally, given an optimal or near-optimal policy for a planning domain, the trajectories represent high-quality solution plans. In this sense, reactive policies can be viewed as efficient domain-specific planners that avoid unconstrained search. Later in the paper, in Section 7, we will introduce two formal representations for policies: decision rule lists and measures of progress, and describe learning algorithms for each representation. Below we describe some of the prior approaches to using policies to guide forward search and the new approach that we propose in this work.

The simplest approach to using a reactive policy as control knowledge is to simply avoid search altogether and follow the trajectory suggested by the policy. There have been a number of studies (Khardon, 1999; Martin and Geffner, 2000; Yoon et al., 2002, 2005; Fern et al., 2006) that consider using learned policies in this way in AI planning context. While there have been some positive results, for many planning domains the results have been mostly negative. One reason for these failures is that inductive, or statistical, policy learning can result in imperfect policies, particularly with limited training data. Although these policies may select good actions in many states, the lack of search prevents them from overcoming the potentially numerous bad action choices.

In an attempt to overcome the brittleness of simply following imperfect reactive policies, previous researchers have considered more sophisticated methods of incorporating imperfect policies into search. Two such methods include discrepancy search (Harvey and Ginsberg, 1995) and policy

rollout (Bertsekas and Tsitsiklis, 1996). Unfortunately, our initial investigation showed that in many planning competition domains these techniques were not powerful enough to overcome the flaws in our learned polices. With this motivation we developed a novel approach that is easy to implement and has proven to be quite powerful.

The main idea is to use reactive policies during the node expansion process of a heuristic search, which in our work is greedy best-first search. Typically in best-first search, only the successors of the current node being expanded are added to the priority queue, where priority is measured by heuristic value. Rather, our approach first executes the reactive policy for h steps from the node being expanded and adds the nodes of the trajectory along with their neighbors to the queue. In all of our experiments, we used a value of $h = 50$, though we found that the results were quite stable across a range of h (we sampled a range from 30 to 200).

Note that when $h = 0$ we get standard (greedy) best-first search. In cases, where the policy can solve a given problem from the current node being expanded, this approach will solve the problem without further search provided that h is large enough. Otherwise, when the policy does not directly lead to the goal, it may still help the search process by putting heuristically better nodes in the search queue in a single node expansion. Without the policy (i.e., $h = 0$) such nodes would only appear in the queue after many node expansions. Intuitively, given a reasonably good heuristic, this approach is able to leverage the good choices made by a policy, while overcoming the flaws. While this technique for incorporating policies into search is simple, our empirical results, show that it is very effective, achieving better performance than either pure heuristic search or search-free policy execution.

5. A Relaxed-Plan Feature Space

A key challenge toward learning control knowledge in the form of heuristics and policies is to develop specific representations that are rich enough to capture important properties of search nodes. In this section, we describe a novel feature space for representing such properties. This feature space will be used as a basis for our policy and heuristic representations described in Sections 6 and 7. Note that throughout, for notational convenience we will describe each search node by its implicit planning problem (s,A,g), where s is the current state of the node, g is the goal, and A is the action set.

Each feature in our space is represented via an expression in taxonomic syntax, which as described in Section 5.4, provides a language for describing sets of objects with common properties. Given a search node (s,A,g) and a taxonomic expression C, the value of the corresponding feature is computed as follows. First, a database of atomic facts $D(s,A,g)$ is constructed, as described in Section 5.3, which specifies basic properties of the search node. Next, we evaluate the class expression C relative to $D(s,A,g)$, resulting in a class or set of objects. These sets, or features, can then be used as a basis for constructing control knowledge in various ways—for example, using the set cardinalities to define a numeric feature representation of search nodes.

Our feature space, is in the spirit of prior work (Martin and Geffner, 2000; Yoon et al., 2002; Fern et al., 2006) that also used taxonomic syntax to represent control knowledge. However, our approach is novel in that we construct databases $D(s,A,g)$ that contain not only facts about the current state and goal, but also facts derived via a bounded reasoning process known as relaxed planning. Prior work, considered only databases that included information about the current state and goal. By

defining features in terms of taxonomic expressions built from our extended databases, we are able to capture properties that are difficult to represent in terms of the state and goal predicates alone.

In the remainder of this section, we first review the idea of relaxed planning, which is central to our feature space. Next, we describe the construction of the database $D(s,A,g)$ for search nodes. Finally, we introduce taxonomic syntax, which is used to build complex features on top of the database.

5.1 Relaxed Plans

Given a planning problem (s,A,g), we define the corresponding relaxed planning problem to be the problem (s,A^+,g) where the new action set A^+ is created by copying A and then removing the delete list from each of the actions. Thus, a relaxed planning problem is a version of the original planning problem where it is not necessary to worry about delete effects of actions. A relaxed plan for a planning problem (s,A,g) is simply a plan that solves the relaxed planning problem.

Relaxed planning problems have two important characteristics. First, although a relaxed plan may not necessarily solve the original planning problem, the length of the shortest relaxed plan serves as an admissible heuristic for the original planning problem. This is because preconditions and goals are defined in terms of positive state facts, and hence removing delete lists can only make it easier to achieve the goal. Second, in general, it is computationally easier to find relaxed plans compared to solving general planning problems. In the worst case, this is apparent by noting that the problem of plan existence can be solved in polynomial time for relaxed planning problems, but is PSPACE-complete for general problems. However, it is still NP-hard to find minimum-length relaxed plans (Bylander, 1994). Nevertheless, practically speaking, there are very fast polynomial time algorithms that typically return short relaxed plans whenever they exist, and the lengths of these plans, while not admissible, often provide good heuristics.

The above observations have been used to realize a number of state-of-the-art planners based on heuristic search. HSP (Bonet and Geffner, 2001) uses forward state-space search guided by an admissible heuristic that estimates the length of the optimal relaxed plan. FF (Hoffmann and Nebel, 2001) also takes this approach, but unlike HSP, estimates the optimal relaxed-plan length by explicitly computing a relaxed plan. FF's style of relaxed plan computation is linear with the length of the relaxed plan, thus fast, but the resulting heuristics can be inadmissible. Our work builds on FF, using the same relaxed-plan construction technique, which we briefly describe below.

FF computes relaxed plans using a relaxed plan graph (RPG). An RPG is simply the usual plan graph created by Graphplan (Blum and Furst, 1995), but for the relaxed planning problem rather than the original problem. Since there are no delete lists in the relaxed plan, there will be no mutex relations in the plan graph. The RPG is a leveled graph alternating between action levels and state-fact levels, with the first level containing the state facts in the initial state. An action level is created by including any action whose preconditions are satisfied in the preceding state-fact level. A state-fact level is created by including any fact that is in the previous fact level or in the add list of an action in the preceding action level. RPG construction stops when a fixed point is reached or the goal facts are all contained in the most recent state-fact level. After constructing the RPG for a planning problem, FF starts at the last RPG level and uses a backtrack-free procedure that extracts a sequence of actions that correspond to a successful relaxed plan. All of this can be done very efficiently, allowing for fast heuristic computation.

While the length of FF's relaxed plan often serves as an effective heuristic, for a number of planning domains, ignoring delete effects leads to severe underestimates of the distance to a goal. The result is poor guidance and failure on all but the smallest problems. One way to overcome this problem would be to incorporate partial information about delete lists into relaxed plan computation, for example, by considering mutex relations. However, to date, this has not born out as a practical alternative. Another possibility is to use more information about the relaxed plan than just its length. For example, Vidal (2004) uses relaxed plans to construct macro actions, which help the planner overcome regions of the state space where the relaxed-plan length heuristic is flat. However, that work still uses length as the sole heuristic value. In this work, we give a novel approach to leveraging relaxed planning, in particular, we use relaxed plans as a source of information from which we can compute complex features that will be used to learn heuristic functions and policies. Interestingly, as we will see, this approach will allow for features that are sensitive to delete lists of actions in relaxed plans, which can be used to help correct for the fact that relaxed plans ignore delete effects.

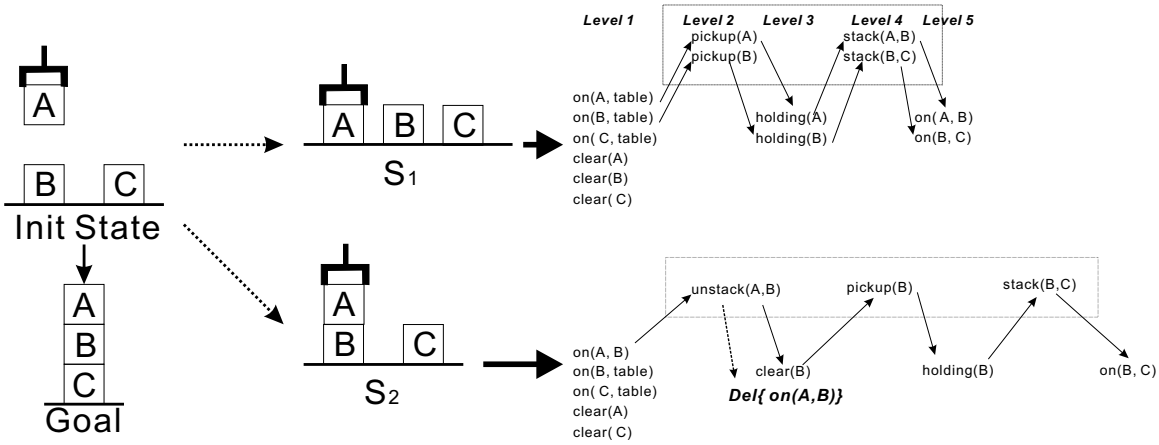


Figure 1: Blocksworld Example

5.2 Example of using Relaxed Plan Features for Learning Heuristics

As an example of how relaxed plans can be used to define useful features, consider a problem from the Blocksworld in Figure 1. Here, we show two states S_1 and S_2 that can be reached from the initial state by applying the actions putdown(A) and stack(A,B) respectively. From each of these states we show the optimal relaxed plans for achieving the goal. For these states, the relaxed-plan length heuristic is 3 for S_2 and 4 for S_1, suggesting that S_2 is the better state. However, it is clear that, in fact, S_1 is better.

Notice that in the relaxed plan for S_2, on(A,B) is in the delete list of the action unstack(A,B) and at the same time it is a goal fact. One can improve the heuristic estimation by adding together the relaxed-plan length and a term related to such deleted facts. In particular, suppose that we had a feature that computed the number of such "on" facts that were both in the delete list of some relaxed plan action and in the goal, giving a value of 0 for S_1 and 1 for S_2. We could then weight this feature by two and add it to the relaxed-plan length to get a new heuristic. This would assign a value of 4 for S_1 and 5 for S_2, correctly ranking the states. Using taxonomic syntax, one can define such a

feature as the cardinality of a certain class expression over a database of facts defined in the next section.

While this is an over-simplified example, it is suggestive as to the utility of features derived from relaxed plans. Below we describe a domain-independent feature space that can be instantiated for any planning domain. Our experiments show that these features are useful across a range of domains used in planning competitions.

5.3 Constructing Databases from Search Nodes

Recall that each feature in our feature space corresponds to a taxonomic syntax expression (see next section) built from the predicate symbols in databases of facts constructed for each search node encountered. We will denote the database for search node (s,A,g) as $D(s,A,g)$, which will simply contain a set of ground facts over some set of predicate symbols and objects derived from the search node. Whereas prior work defined $D(s,A,g)$ to include only facts about the goal g and state s, we will also include facts about the relaxed plan corresponding to problem (s,A,g), denoted by $(a_1,\ldots,a_n)$. Note that in this work we use the relaxed plan computed by FF's heuristic calculation.

Given any search node (s,A,g) we now define $D(s,A,g)$ to be the database that contains the following facts:

- All of the state facts in s.
- The name of each action a_i in the relaxed plan. Recall that each name is an action type Y from domain definition $\mathcal{D}$ applied to the appropriate number of objects, for example, unstack(A,B).
- For each state fact in the add list of some action a_i in the relaxed plan, add a fact to the database that is the result of prepending an **a** to the fact's predicate symbol. For example, in Figure 1, for state S_2 the fact holding(B) is in the add list of pickup(B), and thus we would add the fact **a**holding(B) to the database.
- Likewise for each state fact in the delete list of some a_i, we prepend a **d** to the predicate symbol and add the resulting fact to the database. For example, in Figure 1, for S_2, we would add the fact **d**on(A,B).
- For each state fact in the goal g, we prepend a **g** to the predicate symbol and add it to the database. For example, in Figure 1, we would add the facts **g**on(A,B) and **g**on(B,C).
- For each predicate symbol that appears in the goal, we prepend a **c** to the predicate symbol and add a corresponding fact to the database whenever it is true in the current state s and appears in the goal. For example, the predicate **c**on represents the relation "correctly on", and in Figure 1, for state S_2 we would add the fact **c**on(A,B) since A is currently on B and is supposed to be on B in the goal. The '**c**' predicates provide a useful mechanism for expressing concepts that relate the current state to the goal.

Figure 2, shows an example of the database that would be constructed for the state S_2 in Figure 1. Note that taxonomic syntax class expressions will be constructed from the predicates in this database which include the set of planning domain predicates and action types, along with a variant of each planning domain predicate prepended with an '**a**', '**d**', '**g**', or '**c**'. The database captures information about the state and goal using the planning domain predicates, the '**g**' predicates, and

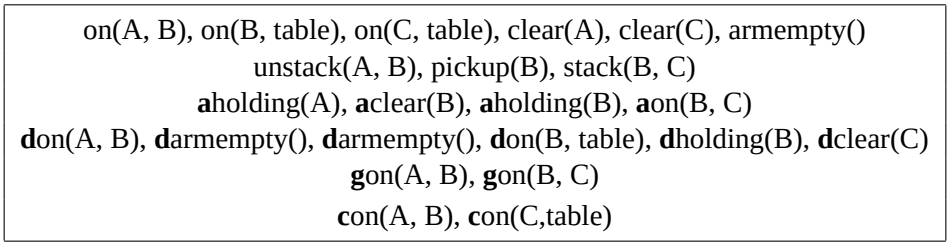


Figure 2: Database for state S_2 in Figure 1

'**c**' predicates. It captures information about the relaxed plan using the action type predicates and the '**a**' and '**d**' predicates. Notice that the database does not capture information about the temporal structure of the relaxed plan. Such temporal information may be useful for describing features, and is a natural extension of this work.

5.4 Defining Complex Features with Taxonomic Syntax

For a given search node (s,A,g) with database $D(s,A,g)$ we now wish to define more complex features of the search node. We will do this using taxonomic syntax (McAllester and Givan, 1993) which is a first-order language for writing class expressions C that are used to denote sets of objects. In particular, given a class expression C, the set of objects it represents relative to $D(s,A,g)$ will be denoted by $C[D(s,A,g)]$. Thus, each C can be viewed as defining a feature that for search node (s,A,g) takes the value $C[D(s,A,g)]$. Below we describe the syntax and semantics of the taxonomic syntax fragment we use in this paper.

Syntax. Taxonomic class expression are built from a set of predicates $\mathcal{P}$, where $n(P)$ will be used to denote the arity of predicate P. For example, in our application the set of predicates will include all predicates used to specify facts in database $D(s,A,g)$ as described above. The set of possible class expressions over $\mathcal{P}$ are given by the following grammar:

$$C := \textbf{a-thing} \mid P_1 \mid C \cap C \mid \neg C \mid (P\ C_1 \ldots C_{i-1}\ ?\ C_{i+1} \ldots C_{\mathrm{n}(P)})$$

where C and C_j are class expressions, P_1 is any arity one predicate, and P is any predicate symbol of arity two or greater. Given this syntax we see that the primitive class expressions are the special symbol **a-thing**, which will be used to denote the set of all objects, and single arity predicates, which will be used to denote the sets of objects for which the predicates are true. One can then obtain compound class expressions via complementation, intersection, or relational composition (the final rule). Before defining the formal semantics of class expressions, we introduce the concept of depth, which will be used in our learning procedures. We define the depth of a class expression C, denoted depth(C) as follows. The depth of **a-thing** or a single arity predicate is 0, depth$(C_1 \cap C_2) = 1 + \max(\text{depth}(C_1), \text{depth}(C_2))$, depth$(\neg C) = 1 + \text{depth}(C)$, and the depth of the expression $(P\ C_1 \ldots C_{i-1}\ ?\ C_{i+1} \ldots C_{\mathrm{n}(P)})$, is $1 + \max\big(\text{depth}(C_1), \ldots, \text{depth}(C_{\mathrm{n}(P)})\big)$. Note that the number of class expressions can be infinite. However, we can limit the number of class expressions under consideration by placing an upper bound on the allowed depth, which we will often do when learning.

Semantics. We now describe the semantics of class expressions, which are defined relative to a finite database D of ground facts over the set of predicates $\mathcal{P}$ and a finite set of constant symbols, or

objects. For example, D might correspond to one of the databases described in the previous section. One can simply view the database D as a finite first-order model, or Herbrand interpretation. Given a class expression C and a database D, we use $C[D]$ to denote the set of objects represented by C with respect to D. We also use $P[D]$ to denote the set of tuples of objects corresponding to predicate P in D, that is, the tuples that make P true.

If $C = \textbf{a-thing}$ then $C[D]$ denotes the set of all objects in D. For example, in a database constructed from a Blocksworld state, **a-thing** would correspond to the set of all blocks. If C is a single arity predicate symbol P, then $C[D]$ is the set of all objects in D for which P is true. For example, if D again corresponds to the Blocksworld then clear$[D]$ and ontable$[D]$ denote the sets of blocks that are clear and on the table respectively in D. If $C = C_1 \cap C_2$ then $C[D] = C_1[D] \cap C_2[D]$. For example, $(\text{clear} \cap \text{ontable})[D]$ denotes the set of blocks that are clear and on the table. If $C = \neg C'$ then $C[D] = \textbf{a-thing} - C'[D]$. Finally, for relational composition, if $C = (P\ \ C_1 \ldots C_{i-1}\ ?\ C_{i+1} \ldots C_{\text{n}(P)})$ then $C[D]$ is the set of all constants c such that there exists $c_j \in C_j[D]$ such that the tuple $(c_1, \ldots, c_{i-1}, c, c_{i+1}, \ldots, c_{\text{n}(R)})$ is in $P[D]$. For example, if D again contains facts about a Blocksworld problem, $(\text{on clear } ?)[D]$ is the set of all blocks that are directly under some clear block.

As some additional Blocksworld examples, $C = (\textbf{con}\ \ \textbf{a-thing}\ ?)$ describes the blocks that are currently directly under the block that they are supposed to be under in the goal. So if D corresponds to the database in Figure 2 then $C[D] = \{B, \text{table}\}$. Recall that as described in the previous section the predicate **con** is true of block pairs (x,y) that are correctly on each other, that is, x is currently on y and x should be on y in the goal. Likewise $(\textbf{con}\ ?\ \textbf{a-thing})$ is the set of blocks that are directly above the block they are supposed to be on in the goal and would be interpreted as the set $\{A, C\}$ in database D. Another useful concept is $\neg(\textbf{con}\ ?\ \textbf{a-thing})$ which denotes the set of blocks that are not currently on their final destination block and would be interpreted as $\{B, \text{table}\}$ with respect to D.

6. Learning Heuristic Functions

Given the relaxed plan feature space, we will now describe how to use that space to represent and learn heuristic functions for use as control knowledge in forward state-space search. Recall from Section 4.1 that we will use the learned heuristics to control (greedy) best-first search in our experiments. Below we first review our heuristic representation followed by a description of our learning algorithm. Recall that heuristic functions are just one of two general forms of control knowledge that we consider in this paper. Our second form, reactive policies, will be covered in the next section.

6.1 Heuristic Function Representation

Recall that a heuristic function $H(s, A, g)$ is simply a function of a state s, action set A, and goal g that estimates the cost of achieving the goal from s using actions in A. In this section, we will consider learning heuristic functions that are represented as weighted linear combinations of functions f_i, that is, $H(s, A, g) = \Sigma_i w_i \cdot f_i(s, A, g)$. In particular, for each planning domain we would like to learn a distinct set of functions f_i and their corresponding weights that lead to good planning performance in that domain when guided by the resulting linear heuristic function. In this work, each function will correspond to a class expression C_i defined over the relaxed-plan database as described in the previous section, and will be denoted by f_{C_i}. We will take the numeric value of f_{C_i} given a

search node (s,A,g) to be the cardinality of C_i with respect to $D(s,A,g)$—that is, $f_{C_i}(s,A,g) = |C_i[D(s,A,g)]|$.

6.2 Heuristic Function Learning

The input to our learning algorithm is a set of planning problems, each paired with an example solution plan, taken from a target planning domain. We do not assume that these solutions are optimal, though there is an implicit assumption that the solutions are reasonably good. Our learning objective is to learn a heuristic function that closely approximates the observed distance-to-goal for each state in the training solutions. To do this, we first create a derived training set $\mathbb{J}$ that contains a training example for each state in the solution set. In particular, for each training problem (s_0,A,g) and corresponding solution trajectory $(s_0,s_1,\ldots,s_n)$ we add to $\mathbb{J}$ a set of n examples $\{((s_i,A,g),n-i) \mid i=0,\ldots,n-1\}$, each example being a pair of a planning problem and the observed distance-to-goal in the training trajectory. Given the derived training set $\mathbb{J}$, we then attempt to learn a real valued function $\Delta(s,A,g)$ that closely approximates the difference between the distances recorded in $\mathbb{J}$ and the value of FF's relaxed-plan length (RPL) heuristic, $\mathrm{RPL}(s,A,g)$. We then take the final heuristic function to be $H(s,A,g) = \mathrm{RPL}(s,A,g) + \Delta(s,A,g)$.

We represent $\Delta(s,A,g)$ as a finite linear combination of functions f_{C_i} with the C_i selected from the relaxed plan feature space, that is, $\Delta(s,A,g) = \Sigma_i w_i \cdot f_{C_i}(s,A,g)$. Note that the overall representation for $H(s,A,g)$ is a linear combination of features, where the feature weight of $\mathrm{RPL}(s,A,g)$ has been clamped to one. Another design choice could have been to allow the weight of the RPL feature to also be learned, however, an initial exploration showed that constraining the value to be one and learning the residual $\Delta(s,A,g)$ gives moderately better performance in some domains.

Learning the above representation involves selecting a set of class expressions from the above infinite space defined in Section 5 and then assigning weights to the corresponding features. One approach to this problem would be to impose a depth bound on class expressions and then learn the weights (e.g., using least squares) for a linear combination that involves all features whose depths of class expression are within the bound. However, the number of such features is exponential in the depth bound, making this approach impractical for all but very small bounds. Such an approach will also have no chance of finding important features beyond the fixed depth bound. In addition, we would prefer to use the smallest possible number of features, since the time complexity of evaluating the learned heuristic grows in proportion to the number of selected features. Thus, we consider a greedy learning approach where we heuristically search through the space of features, without imposing apriori depth bounds. The procedure described below is a relatively generic approach that we found to work well, however, alternative more sophisticated search approaches are an important direction for future work.

Figure 3 gives our algorithm for learning $\Delta(s,A,g)$ from a derived training set $\mathbb{J}$. The main procedure **Learn-Delta** first creates a modified training set $\mathbb{J}'$ that is identical to $\mathbb{J}$ except that the distance-to-goal of each training example is changed to the difference between the distance-to-goal and FF's relaxed-plan length heuristic. Each iteration of **Learn-Delta** maintains a set of class expressions Φ, which represents the set of features that are currently under consideration. Initially Φ is equal to the set of expressions of depth 0 and 1. Each iteration of the loop has two main steps. First, we use the procedure **Learn-Approximation** (described below) to select a subset of class expressions from Φ and to compute their feature weights. Second, we create a new candidate feature set Φ that includes higher depth class expressions. This is done by using the selected features

as seeds and then calling the procedure **Expand-Features**. This results in a larger candidate feature set, including the seed features, which is again used by **Learn-Approximation** to find a possibly improved approximation. We continue alternating between feature space expansion and learning an approximation until the approximation accuracy does not improve. Here we measure the accuracy of the approximation by the R-square value, which is the fraction of the variance in the data that is explained by the linear approximation.

Learn-Approximation uses a simple greedy procedure. Starting with an empty feature set, on each iteration the feature from Φ that can most improve the R-square value of the current feature set is included in the approximation. This continues until the R-square value can no longer be improved. Given a current feature set, the quality of a newly considered feature is measured by calling the function *lm* from the statistics tool R, which outputs the R-square value and weights for a linear approximation that includes the new feature. After observing no improvement, the procedure returns the most recent set of selected features along with their weights, yielding a linear approximation of $\Delta(s,A,g)$.

The procedure **Expand-Features** creates a new set of class expressions that includes the seed set, along with new expressions generated from the seeds. There are many possible ways to generate an expanded set of features from a given seed C. Here we consider three such expansion functions that worked well in practice. The first function **Relational-Extension** takes a seed expression C and returns all expressions of the form $(P\ c_0 \ldots c_{j-1}\ C\ c_{j+1} \ldots c_{i-1}\ ?\ c_{i+1} \ldots c_{\mathrm{n}(P)})$, where P is a predicate symbol of arity larger than one, the c_i are all **a-thing**, and $i, j \leq \mathrm{n}(P)$. The result is all possible ways of constraining a single argument of a predicate by C and placing no other constraints on the predicate. For example in Blocksworld, a relational extension of the class expression, holding, is (**g**on holding ?). The extended expression describes the block in the goal state that should be under the block currently being held.

The second procedure for generating new expressions given a class expression C is **Specialize**. This procedure simply generates all class expressions that can be created by replacing a single depth zero or one sub-expression c' of C with the intersection of c' and another depth zero or one class expression. Note that all expressions that are produced by **Specialize** will be subsumed by C. That is, for any such expression C', we have that for any database D, $C'[D] \subseteq C[D]$. As an example, given the Blocksworld expression (on ? **a-thing**), one of the class expression generated by **Specialize** would be (on ? (**a-thing** $\cap$ **g**clear)). The input class expression describes all the blocks on some block, and the example output class expression describes the blocks that are currently on blocks that should be clear in the goal state. Finally, we add the complement of the seed classes into the expanded feature set. For example in Logisticsworld, for the input class expression (**c**in ? **a-thing**), the complement output is $\neg$(**c**in ? **a-thing**). The input describes packages that are already in the goal location, and the output describes packages that are not in the goal location.

7. Learning Reactive Policies

In this section, we present two representations and associated learning algorithms for reactive policies: taxonomic decision lists and measures of progress. Recall that Section 4.2 describes in detail our novel approach for using the resulting policies as search control knowledge.

Learn-Delta $(\mathbb{J}, \mathcal{D})$
// $\mathbb{J}$ is pairs of problem states and plan length from them
// $\mathcal{D}$ is domain definition to enumerate class expressions
$\mathbb{J}' \leftarrow \{((s,A,g), d - \text{RPL}(s,A,g)) \mid ((s,A,g),d) \in \mathbb{J}\}$
d is the plan length, the remaining states in the solution trajectories
$\Phi \leftarrow \{C \mid C$ is a class expression of depth 0 or 1$\}$
repeat until no R-square value improvement observed
$(\Phi', W) \leftarrow$ **Learn-Approximation($\mathbb{J}'$, Φ)**
// Φ' is newly selected features, W is the set of weights for Φ'
$\Phi \leftarrow$ **Expand-Features($\mathcal{D}$, Φ')**
Return Φ', W

Learn-Approximation $(\mathbb{J}, \Phi)$
$\Phi' \leftarrow \{\}$ *// return features*
repeat until no improvement in R-square value
$C \leftarrow \arg\max_{C \in \Phi}$ R-square$(\mathbb{J}, \Phi' \cup \{C\})$
// R-square is computed after linear approximation with the features
$\Phi' \leftarrow \Phi' \cup \{C\}$
$W \leftarrow lm(\mathbb{J}, \Phi')$
// lm, least square approximation, returns weights
Return Φ', W

Expand-Features $(\mathcal{D}, \Phi')$
$\Phi \leftarrow \Phi'$ *// return features*
for-each $C \in \Phi'$
$\Phi \leftarrow \Phi \cup$ **Relational-Extension**$(\mathcal{D}, C) \cup$ **Specialize**$(\mathcal{D}, C) \cup \{\neg C\}$
Return Φ

Figure 3: Pseudo-code for learning heuristics: The learning algorithm used to approximate the difference between the relaxed plan length heuristic and the observed plan lengths in the training data.

7.1 Taxonomic Decision Lists

We first consider representing and learning policies as taxonomic decision lists. Similar representations have been considered previously (Martin and Geffner, 2000; Yoon et al., 2002), though this is the first work that builds such lists from relaxed-plan-based features.

7.1.1 REPRESENTATION

A taxonomic decision list policy is a list of taxonomic *action-selection rules.* Each rule has the form

$$a(x_1, \ldots, x_k) : L_1, L_2, \ldots L_m$$

where a is a k-argument action type, the L_i are *literals,* and the x_i are action-argument variables. Each literal has the form $x \in C$, where C is a taxonomic syntax class expression and x is an action-argument variable.

Given a search node (s,A,g) and a list of action-argument objects $O = (o_1,\ldots,o_k)$, we say that a literal $x_i \in C$ is satisfied if $o_i \in C[D(s,A,g)]$, that is, object o_i satisfies the constraint imposed by the class expression C. We say that a rule $R = a(x_1,\ldots,x_k) : L_1,L_2,\ldots L_m$ *suggests* action $a(o_1,\ldots o_k)$ in (s,A,g) if each literal in the rule is true given (s,A,g) and O, and the preconditions of the action are satisfied in s. Note that if there are no literals in a rule for action type a, then all legal actions of type a are suggested by the rule. A rule can be viewed as placing mutual constraints on the tuples of objects that an action type can be applied to. Note that a single rule may suggest no action or many actions of one type. Given a decision list of such rules we say that an action is suggested by the list if it is suggested by some rule in the list, and no previous rule suggests any actions. Again, a decision list may suggest no action or multiple actions of one type.

A decision list $\mathbb{L}$ defines a deterministic policy $\pi[\mathbb{L}]$ as follows. If $\mathbb{L}$ suggests no action for node (s,A,g), then $\pi[\mathbb{L}](s,A,g)$ is the lexicographically least action in s, whose preconditions are satisfied; otherwise, $\pi[\mathbb{L}](s,A,g)$ is the least action suggested by $\mathbb{L}$. It is important to note that since $\pi[\mathbb{L}]$ only considers legal actions, as specified by action preconditions, the rules do not need to explicitly encode the preconditions, which allows for simpler rules and learning. In other words, we can think of each rule as implicitly containing the preconditions of its action type.

As an example of a taxonomic decision list policy, consider a simple Blocksworld domain where the goal is to place all of the blocks on the table. The following policy will solve any problem in the domain.

$$\begin{aligned} \text{putdown}(x_1) &: x_1 \in \text{holding}, \\ \text{pickup}(x_1) &: x_1 \in (\text{on ? (on ? } \textbf{a-thing})). \end{aligned}$$

The first rule will cause the agent to putdown any block that is being held. Otherwise, if no block is being held, then the second rule will pickup a block x_1 that is directly on top of a block that is directly on top of another object (either the table or another block). In particular, this will pickup a block at the top of a tower of height two or more, as desired.[3]

7.1.2 DECISION LIST LEARNING

Figure 4 depicts the learning algorithm we use for decision list policies. The training set $\mathbb{J}$ passed to the main procedure **Learn-Decision-List** is a multi-set that contains all pairs of search nodes and corresponding actions observed in the solution trajectories. The objective of the learning algorithm is to find a taxonomic decision list that for each search node in the training set suggests the corresponding action.

The algorithm takes a Rivest-style decision list learning approach (Rivest, 1987) where one rule is learned at a time, from highest to lowest priority, until the resulting rule set "covers" all of the training data. Here we say that a rule *covers* a training example if it suggests an action for the corresponding state. An ideal rule is one that suggests only actions that are in the training data.

The main procedure **Learn-Decision-List** initializes the rule list to () and then calls the procedure **Find-Best-Rule** in order to select a rule that covers many training examples and that correctly covers a high fraction of those examples—that is, a rule with high coverage and high precision. The resulting rule is then added to the tail of the current decision list, and at the same time the training examples that it covers are removed from the training set. The procedure then searches for another

3. If the second rule is changed to pickup$(x_1) : x_1 \in$ (on ? **a-thing**), then the decision rule list may find the loop, since it might try to pick up a block on the table that has just been put down.

Learn-Decision-List $(\mathbb{J}, d, b)$
// $\mathbb{J}$: set of training instances where each instance is a search node labeled by an action
// d: the depth limit for class expressions
// b: beam width, used in search for the rules
$\mathbb{L} \leftarrow ()$
while ($\mathbb{J} \neq \{\}$)
$\mathbb{R} \leftarrow$ **Find-Best-Rule** $(\mathbb{J}, d, b)$
$\mathbb{J} \leftarrow \mathbb{J} - \{j \in \mathbb{J} \mid \mathbb{R}$ suggests an action for $j\}$
$\mathbb{L} \leftarrow \mathbb{L} : \mathbb{R}$; *// append rule to end of current list*
Return $\mathbb{L}$

Find-Best-Rule$(\mathbb{J}, d, b)$
Hvalue-best-rule $\leftarrow -\infty$; $\mathbb{R} \leftarrow ()$
for-each action type a
$\mathbb{R}_a \leftarrow$**Beam-Search(**$a, \mathbb{J}, d, b$**)**
if $H(\mathbb{J}, \mathbb{R}_a) >$ **Hvalue-best-rule**
// $H(J, R_a)$ is learning heuristic function in Equation 2
$\mathbb{R} \leftarrow \mathbb{R}_a$
Hvalue-best-rule $\leftarrow H(\mathbb{J}, \mathbb{R}_a)$
Return $\mathbb{R}$

Beam-Search$(a, \mathbb{J}, d, b)$
$L_{set} \leftarrow \{(x_k \in C) | k \leq \mathbf{n}(a), \text{depth}(C) \leq d\}$
// the set of all possible literals involving class expressions of depth d or less
beam $\leftarrow \{a(x_1, \ldots, a_k)\}$ *// initial beam contains rule with empty rule body*
Hvalue-best $\leftarrow -\infty$; **Hvalue-best-new** $\leftarrow 0$
while (**Hvalue-best** $<$ **Hvalue-best-new**)
Hvalue-best $\leftarrow$ **Hvalue-best-new**
candidates $\leftarrow \{R, l | l \in L_{set}, R \in$ **beam**$\}$
// the set of all possible rules resulting from adding one literal to a rule in the beam
beam $\leftarrow$ set of b best rules in **candidates** according to heuristic H from Equation 2
Hvalue-best-new $\leftarrow H$ value of best rule in **beam**
Return best rule in **beam**

Figure 4: Pseudo-code for learning policy

rule with high coverage and high precision with respect to the reduced training set. The process of selecting good rules and reducing the training set continues until no training examples remain uncovered. Note that by removing the training examples covered by previous rules we force **Find-Best-Rule** to focus on only the training examples for which the current rule set does not suggest an action.

The key procedure in the algorithm is **Find-Best-Rule**, which at each iteration does a search through the exponentially large rule space for a good rule. Recall that each rule has the form

$$a(x_1, \ldots, x_k) : L_1, L_2, \ldots L_m$$

where a is one of the action types and the L_i are of the form $x \in C$. Since this rule space is exponentially large we use a greedy beam-search approach. In particular, the main loop of **Find-Best-Rule**

loops over each action type a and then uses a beam search to construct a set of literals for that particular a. The best of these rules, as measured by an evaluation heuristic, is then returned. It remains to describe the beam search over literal sets and our heuristic evaluation function.

The input to the procedure **Beam-Search** is the action type a, the current training set, a beam width b, and a depth bound d on the maximum size of class expressions that will be considered. The beam width and the depth bound are user specified parameters that bound the amount of search. We used $d = 2, b = 10$ in all of our experiments. The search is initialized so that the current beam contains only the empty rule, that is, a rule with head $a(x_1, \ldots, x_k)$ and no literals. On each iteration of the search a candidate rule set is constructed that contains one rule for each way of adding a new literal, with depth bound d, to one of the rules in the beam. If there are n possible literals of depth bound d this will resulting in a set of nb rules. Next the rule evaluation heuristic is used to select the best b of these rules which are kept in the beam for the next iteration, with all other candidates being discarded. The search continues until the search is unable to uncover an improved rule as measured by the heuristic.

Finally, our rule evaluation heuristic $H(\mathbb{J}, R)$ is shown in Equation 2, which evaluates a rule R with respect to a training set $\mathbb{J}$. There are many good choices for heuristics and this is just one that has shown good empirical performance in our experience. Intuitively this function will prefer rules that suggest correct actions for many search nodes in the training set, while at the same time minimizing the number of suggested actions that are not in the training data. We use $R(s,A,g)$ to represent the set of actions suggested by rule R in (s,A,g). Using this, Equation 1, evaluates the "benefit" of rule R on training instance $((s,A,g),a)$ as follows. If the training set action a is not suggested by R then the benefit is zero. Otherwise the benefit decreases with the size of $R(s,A,g)$. That is, the benefit decreases inversely with the number of actions other than a that are suggested. The overall heuristic H is simply the sum of the benefits across all training instances. In this way the heuristic will assign small heuristic values to rules that cover only a small number of examples and rules that cover many examples but suggest many actions outside of the training set.

$$\text{benefit}(((s,A,g),a),R) = \begin{cases} 0 & : \quad a \notin R(s,A,g) \\ \frac{1}{|R(s,A,g)|} & : \quad a \in R(s,A,g), \end{cases} \tag{1}$$

$$H(\mathbb{J}, R) = \sum_{j \in \mathbb{J}} \text{benefit}(j, R). \tag{2}$$

7.2 Measures of Progress

In this section, we describe the notion of measures of progress and how they can be used to define policies and learned from training data.

7.2.1 REPRESENTATION

Good plans can often be understood as seeking to achieve specific subgoals en route to the goal. In an object-oriented view, these subgoals, along with the goal itself, can be seen as properties of objects, where in each case we wish to increase the number of objects with the given property. Taking this view, we consider control knowledge in the form of compact descriptions of object classes (properties) that a controller is to select actions to enlarge. For example in Blocksworld, in an optimal trajectory, the number of blocks well placed from the table up never decreases, or in Logisticsworld the number of solved packages never decreases in an optimal plan.

Good trajectories also often exhibit locally monotonic properties: properties that increase monotonically for identifiable local periods, but not all the time during the trajectory. For example, in Blocksworld, consider a block "solvable" if its desired destination is clear and well placed from the table up. Then, in good trajectories, while the number of blocks well placed from the table up stays the same, the number of solvable blocks need never decrease locally; but, globally, the number of solvable blocks may decrease as the number of blocks well placed from the table up increases. Sequences of such properties can be used to define policies that select actions in order to improve the highest-priority property possible, while preserving higher-priority properties.

Previously, the idea of monotonic properties of planning domains have been identified by Parmar (2002) as "measures of progress" and we inherit the term and expand the idea to ensembles of measures where the monotonicity is provided via a prioritized list of functions. Let $F = (F_1, \ldots, F_n)$ be an ordered list where each F_i is a function from search nodes to integers. Given an F we define an ordering relation on search nodes (s,A,g) and (s',A,g) as $F(s,A,g) \succ F(s',A,g)$ if $F_i(s,A,g) > F_i(s',A,g)$ while $F_j(s,A,g) = F_j(s',A,g)$ for all $j < i$. F is a strong measure of progress for planning domain $\mathcal{D}$ iff for any reachable problem (s,A,g) of $\mathcal{D}$, either $g \subseteq s$ or there exists an action a such that $F(a(s),A,g) \succ F(s,A,g)$. This definition requires that for any state not satisfying the goal there must be an action that increases some component heuristic F_i while maintaining the preceding, higher-priority components. In this case we say that such an action has priority i. Note that we allow the lower priority heuristics that follow F_i to decrease so long as F_i increases. If an action is not able to increase some F_i, while maintaining all higher-priority components, we say that the action has null priority. In this work we represent our prioritized lists of functions $F = (F_1, \ldots, F_n)$ using a sequence of class expressions $\mathbb{C} = (C_1, \ldots, C_n)$ and just as was the case for our heuristic representation we take the function values to be the cardinalities of the corresponding sets of objects, that is, $F_i(s,A,g) = |C_i[D(s,A,g)]|$.

Given a prioritized list $\mathbb{C}$, we define the corresponding policy $\pi[\mathbb{C}]$ as follows. Given search node (s,A,g), if all legal actions in state s have null priority, then $\pi[\mathbb{C}](s,A,g)$ is just the lexicographically least legal action. Otherwise $\pi[\mathbb{C}](s,A,g)$ is the lexicographically least legal action that achieves highest priority among all other legal actions.

As a simple example, consider again a Blocksworld domain where the objective is to always place all the blocks on the table. A correct policy for this domain is obtained using a prioritized class expression list (C_1, C_2) where $C_1 = \neg(\text{on } ? \text{ (on } ? \textbf{ a-thing}))$ and $C_2 = \neg\text{holding}$. The first class expression causes the policy to prefer actions that are able to increase the set of objects that are *not* above at least two other objects (objects directly on the table are in this set). This expression can always be increased by picking up a block from a tower of height two or greater when the hand is empty. When the hand is not empty, it is not possible to increase C_1 and thus actions are preferred that increase the second expression while not decreasing C_1. The only way to do this is to putdown the block being held on the table, as desired.

7.2.2 LEARNING MEASURES OF PROGRESS

Figure 5 describes the learning algorithm for measures of progress. The overall algorithm is similar to our Rivest-style algorithm for learning decision lists. Again each training example is a pair $((s,A,g),a)$ of a search node and the corresponding action selected in that node. Each iteration of the main procedure **Learn-Prioritized-Measures** finds a new class expression, or measure, that is added to the tail of the prioritized list and then removes any newly "covered" examples from the

training set. Here we say that a measure C covers a training example $((s,A,g),a)$ if $|C(D[s,A,g])| \neq |C(D[a(s),A,g])|$. It covers the example positively, if $|C(D[s,A,g])| < |C(D[a(s),A,g])|$ and covers it negatively otherwise. Intuitively if a class expression positively covers an example then it increases across the state transition caused by the action of the example. Negative coverage corresponds to decreasing across the transition. We stop growing the prioritized list when we are unable to find a new measure with positive heuristic value.

The core of the algorithm is the procedure **Find-Best-Measure**, which is responsible for finding a new measure of progress that positively covers as many training instances as possible, while avoiding negative coverage. To make the search more tractable we restrict our attention to class expressions that are intersections of class expressions of depth d or less, where d is a user specified parameter. The search over this space of class expressions is conducted using a beam search of user specified width b which is initialized to a beam that contains only the universal class expression **a-thing**. We used $d = 2, b = 10$ in all of our experiments. Given the current beam of class expressions, the next set of candidates contains all expressions that can be formed by intersecting an element of the beam with a class expression of depth d or less. The next beam is then formed by selecting the best b candidates as measured by a heuristic. The search ends when it is unable to improve the heuristic value, upon which the best expression in the beam is returned.

To guide the search we use a common heuristic shown in Equation 3, which is known as weighted accuracy (Furnkranz and Flach, 2003). This heuristic evaluates an expression by taking a weighted difference between the number of positively covered examples and negatively covered examples. The weighting factor ω measures the relative importance of negative coverage versus positive coverage. In all of our experiments, we have used $\omega = 4$ which results in a positive value when the positive coverage is at least four times the negative coverage.

$$H_m(\mathbb{J},C,\omega) = |\{j|C \text{ covers positively } j \in \mathbb{J}\}| - \omega \times |\{j|C \text{ covers negatively } j \in \mathbb{J}\}|. \quad (3)$$

We note that one shortcoming of our current learning algorithm is that it can be fooled by properties that monotonically increase along all or many trajectories in a domain, even those that are not related to distinguishing between good and bad plans. For example, consider a domain with a class expression C, where $|C|$ never decrease and frequently increases along any trajectory. Our learner will likely output this class expression as a solution, although it does not in any way distinguish good from bad trajectories. In many of our experimental domains, such properties do not seem to exist, or at least are not selected by our learner. However, in PHILOSOPHER from IPC 4, this problem did appear to arise and hurt the performance of policies based on measures of progress.

There are a number of possible approaches for dealing with this pitfall. For example, one idea would be to generate a set of random (legal) trajectories and reward class expressions that can distinguish between the random and training trajectories.

8. Experiments

We evaluated our learning techniques on the traditional benchmark domain Blocksworld and then on a subset of the STRIPS/ADL domains from two recent international planning competitions (IPC3 and IPC4). We included all of the IPC3 and IPC4 domains where FF's RPL heuristic was sufficiently inaccurate on the training data, so as to afford our heuristic learner the opportunity to learn. That is, for domains where the FF heuristic is very accurate as measured in the first 15 training problems,

Learn-Prioritized-Measures $(\mathbb{J}, d, b, \omega)$
// $\mathbb{J}$: states and corresponding actions in the solution plan trajectories
// d: the depth limit of class expressions
// b is the beam width for the search for best measures
// ω is the weight used for the heuristic defined in Equation 3

$\mathbb{C} \leftarrow ()$ // $\mathbb{C}$ *is the list of prioritized measures*
while ($\mathbb{J} \neq \{\}$)
  $C \leftarrow$ **Find-Best-Measure** $(\mathbb{J}, d, b, \omega)$
  if $H_m(\mathbb{J}, C) \leq 0$ // *see Equation 3 for* H_m
    Return $\mathbb{C}$
  $\mathbb{J} \leftarrow \mathbb{J} - \{j \in \mathbb{J} \mid C \text{ covers } j\}$
  $\mathbb{C} \leftarrow \mathbb{C} : C$ // *append new expression to list*

Return $\mathbb{C}$

Find-Best-Measure$(\mathbb{J}, d, b, \omega)$

beam $\leftarrow \{$**a-thing**$\}$
Hvalue-best $\leftarrow -\infty$
Hvalue-best-new $\leftarrow 0$
while (**Hvalue-best-new** $>$ **Hvalue-best**)
  Hvalue-best $\leftarrow$ **Hvalue-best-new**
  candidates $\leftarrow \{C' \cap C | C \in$ **beam**$, \text{depth}(C') \leq d\}$
  beam $\leftarrow$ best b elements of **candidates** according to $H_m(J, C, \omega)$
  Hvalue-best-new $\leftarrow H_m$ value of best element of **beam**

Return best element of **beam**

Figure 5: Pseudo-code for learning measures of progress

our heuristic learning has nothing to learn since its training signal is the difference between the observed distance in the training set and FF's heuristic. Thus, we did not include such domains.

For each domain, we used 15 problems as training data, with the solutions being generated by FF. We then learned all three types of control knowledge (heuristics, taxonomic decision lists, and measures of progress) in each domain and used that knowledge to solve the remaining problems, which were typically more challenging than the training problems. We used each form of control knowledge as described in Section 4. For the case of the policy representations (taxonomic decision lists and measures of progress), we used FF's RPL heuristic as the heuristic function and used a fixed policy-execution horizon of 50.

To study the utility of our proposed relaxed-plan feature space, we also conducted separate experiments that removed the relaxed plan features from consideration. As the experiments will show, the relaxed-plan features were critical to achieving good performance in a number of domains. The time cutoff for each planning problem was set to 30 CPU minutes and a problem was considered unsolved after reaching the cutoff. For all of the experiments, we used a Linux box with 2 Gig RAM and 2.8 Ghz Intel Xeon CPU.

8.1 Table Mnemonics

Before we present our results, we first explain the mnemonics used in our data tables. We provide one table for each of our domains, each having the same structure, for example, Figure 6 gives our Blocksworld results. Each row corresponds to a distinct planning technique, some using learned control knowledge and some not. The mnemonics used for these planners are described below. These mnemonics are also used in the main text.

- **FF**: the planner FF (Hoffmann and Nebel, 2001). FF adds goal-ordering, enforced-hill climbing, and helpful action pruning on top of relaxed plan heuristic search. If all these fail, FF falls back on best first search guided by relaxed plan length.
- **RPL**: greedy best-first search using FF's RPL heuristic.
- **Best**: best performer of the corresponding domain during the competition (only available for IPC4).
- **DL**: greedy best-first search guided by RPL and *learned decision list policy* using the full relaxed-plan feature space, refer to Section 7 for the representation and learning algorithm and Section 4 for how to use the knowledge.
- **H**: greedy best-first search guided by a *learned heuristic function* using the full relaxed-plan feature space, refer to Section 6 for representation and learning.
- **MoP**: greedy best-first search guided by RPL and *learned measures-of-progress policy* using the full relaxed-plan feature space, refer to Section 7 for the representation and learning and to Section 4 for use of the knowledge to guide the search.
- **DL-noRP, H-noRP, MoP-noRP**: identical to **DL**, **H**, and **MoP** except that the control knowledge is learned from a feature space that does not include relaxed plan information. That is, the database construction described in Section 5.3 does not include any facts related to the relaxed plan into the search node databases. These experiments are conducted to check the usefulness of the relaxed plan information.

The columns of the results tables show various performance measures for the planners. In the following, we list mnemonics for the performance measures and their descriptions.

- **Solved** (n): gives the number of problems solved within 30 minutes out of n test problems.
- **Time**: the average CPU time in seconds consumed across all problems that were solved within the 30 minute cutoff.
- **Length**: average solution length of the problems solved within the 30 minute cutoff.
- **LTime**: the time used for learning (only applies to the learning systems). Unless followed by H, the number is in seconds.
- **Greedy**: number of problems solved using greedy execution of the decision list or measures-of-progress policies (only applies to systems that learn decision list rules and measures of progress). This allows us to observe the quality of the policy without the integration of heuristic search.

- **Evaluated**: the average number of states evaluated on the problems that were solved with the 30 minute cutoff.

8.2 Blocksworld Results

For Blocksworld, we have used the competition problems from track 1 of IPC2. There were 35 problems giving us 15 problems for training and 20 problems for testing. For this domain only, we included one additional predicate symbol "above" that is not part of the original domain definition. The above predicate is computed as the transitive closure of the "on" predicate and is true of tuple (x,y) if x is above y in a stack of towers. This is an important concept to be able to represent in the Blocksworld and is not expressible via the fragment of taxonomic syntax used in this work. Note that in prior work (Yoon et al., 2002; Fern et al., 2006), we included the Kleene-star operator into the taxonomic syntax, which allowed for concepts such as above to be expressed in terms of the primitive predicates. However, we have decided to not include Kleene star in this work as it did not appear necessary in most of the other domains and increases the size of the search space over taxonomic expressions and hence learning time.

Blocksworld (IPC2)						
Techniques	Solved (20)	Time	Length	LTime	Greedy	Evaluated
FF	16	0.64	38.12	-	-	15310
RPL	20	11.74	116	-	-	12932
DL	20	0.05	44	100	13	215
MoP	20	0.13	51.7	10	20	757
H	20	12.94	82.7	600	-	31940
DL-noRP	20	0.12	60.8	12	0	915
MoP-noRP	20	0.15	49.4	1	20	984
H-noRP	12	113.3	352	88	-	185808

Figure 6: Blocksworld (IPC2, Track 1) Results. For information on mnemonics, please refer sub-Section 8.1

Figure 6 shows Blocksworld results for various learning and non-learning systems. For this domain, **DL**, **H**, and **MoP** were all able to solve all of the problems. Note that greedy application of the learned decision list policy manages to solve only 13 of the 20 problems, indicating that the learned policy has a significant error rate. Despite this error rate, however, incorporating the policy into search as implemented in **DL** allows for all 20 problems to be solved. Furthermore, the incorporation of the policy into search significantly speeds up search, achieving an average search time of 0.05 seconds compared to a time of 11.74 seconds achieved by **RPL**, which uses the same RPL heuristic as **DL** but ignores the policy. The number of evaluated states partially shows why running policies in the best first search helps. The average number of evaluated states is significantly lower for **DL** compared to other techniques. Later in our discussion of the Depots domain we will give empirical evidence that one reason for this reduction in the amount of search is that the policies are able to quickly move through plateaus to find states with low heuristic values.

We see that for this domain the measures of progress are learned quite accurately, allowing for greedy search to solve all of the problems. Again, as for the decision-list policies we see that the incorporation of the measures of progress into search significantly speeds up planning time compared to **RPL**. The fact that measures of progress are learned more accurately is possibly due to the fact that the training data for decision-list policy learning is quite noisy. That is, all actions not in the training plans are treated as negative examples, while in fact many of those actions are just as good as the selected action, since there are often many good action choices in a state. The training data for learning measures of progress does not include such noisy negative examples. Note that prior work (Yoon et al., 2002) has learned highly accurate Blocksworld policies, however, there the training data contained the set of *all* optimal actions for each state, with all other actions labeled as negative. Thus, the training data was not nearly as noisy in that work.

Considering now the performance of the heuristic learner **H**, we see that overall its solution times were larger than for **RPL** and also considers more states than **RPL**. However, the heuristic learner **H** did find significantly better solutions than **RPL**, which used only the RPL heuristic, reducing the average length from 116 to 83. Thus, by attempting to learn a more accurate heuristic, **H** is able to find higher quality solutions at the expense of more search.

Overall we see that the learners **DL** and **MoP** are more effective in this domain. This is likely because it is possible to learn very good decision list rules and measure of progress in the Blocksworld, which guide the heuristic search to good solutions very quickly. Note that FF solves fewer problems than other systems, but the average solution length of FF is the best, noting that it is difficult to compare averages between planners that solve different sets of problems. Apparently enforced hill-climbing and the goal-ordering mechanisms of FF help facilitate shorter solutions when they are able to find solutions.

Our feature comparison results indicate that when the relaxed plan features are removed, decision-list and measures-of-progress learning still solve all of the problems, but the heuristic learner **H-noRP** only solves 12 problems. This indicates that the relaxed plan information is important to the heuristic learner in this domain. We note that previous work on learning Blocksworld decision list rules and measures did not use relaxed-plan features and also had reasonable success. Thus, for the Blocksworld it was not surprising that relaxed plan features were not critical for the policy learners.

The learning time for policies and measures are negligible for Blocksworld. As will be revealed later, the learning time for decision-list policies is quite significant in many other domains. For the Blocksworld domain, the number of predicates, the number of actions, and the arities of predicates and actions are all small, which greatly reduces the complexity of feature enumeration during learning, as described in Section 7. For many domains in the following experiments those numbers increase sharply, and accordingly so does the learning time. It is important to remember, however, that learning time is a price we pay once, and can then be amortized over all future problem solving.

8.3 IPC3 Results

IPC3 included 6 STRIPS domains: Zenotravel, Satellite, Rover, Depots, Driverlog, and FreeCell. FF's heuristic is very accurate for the first three domains, trivially solving the 15-training-problems and leaving little room for learning. Thus, here we report results on Depots, Driverlog, and FreeCell. Each domain includes 20 problems, and FF was able to generate training data for all 15 training problems. Figures 7, 8, 9 summarizes the results.

Depots (IPC3)						
Techniques	Solved (5)	Time	Length	LTime	Greedy	Evaluated
FF	5	4.32	54.2	-	-	1919
RPL	1	0.28	29	-	-	1106
DL	5	3.73	63.2	8H	0	2500
MoP	5	48.5	81.2	950	0	16699
H	5	174	68.4	325	-	71795
DL-noRP	2	0.83	30	8H	0	437
MoP-noRP	4	94.5	124.7	900	0	101965
H-noRP	3	5	61.7	258	-	4036

Figure 7: Depots results

Driverlog (IPC3)						
Techniques	Solved (5)	Time	Length	LTime	Greedy	Evaluated
FF	1	37.62	149	-	-	171105
RPL	1	1623	167	-	-	165454
DL	4	75.9	177	6H	0	15691
MoP	5	546	213	600	0	126638
H	3	199	402	274	-	98801
DL-noRP	3	86	142	8H	0	33791
MoP-noRP	3	583	177	900	0	253155
H-noRP	1	37	149	252	-	22164

Figure 8: Driverlog results

For Depots, FF performed the best, solving all of the problems. At the same time the average plan length was short. Clearly, the goal-ordering and the enforced hill climbing were key factors to this success since **RPL**, which carries out just the best-first search component of FF, only solved one problem. The learning systems **DL**, **MoP** and **H** were all able to solve all of the problems, showing that the learned knowledge was able to overcome the poor performance of **RPL**. However, the learning systems consumed more time than FF and/or resulted in longer plans. In the Driverlog domain, **MoP** was the only system able to solve all of the problems, with **H** and **DL** solving 4 and 3 problems respectively compared to the single solved problem of the non-learning systems. In FreeCell, both non-learning systems were able to outperform the learning systems in terms of problems solved and plan lengths. The learning system **DL** is close behind these systems, solving one less problem, with about the same average length. This domain is a clear example of how learned control knowledge can sometimes hurt performance. In practice, one might use a validation process to determine whether to use learned control knowledge or not, and also to select among various forms of control knowledge.

Aggregating the number of problems solved across the three domains in IPC3, the learning and planning systems **DL**, **MoP** and **H** all solved more problems than FF and **RPL**. Overall the learned control knowledge generally speeds up planning. The solution lengths are sometimes longer,

Freecell (IPC3)						
Techniques	Solved (5)	Time	Length	LTime	Greedy	Evaluated
FF	5	574	108	-	-	26344
RPL	5	442	109	-	-	25678
DL	4	217	108	6H	0	12139
MoP	3	371	192	1000	0	44283
H	5	89	145	4214	-	5096
DL-noRP	4	200	111	8H	0	11132
MoP-noRP	3	368	129	900	0	18299
H-noRP	2	29.87	85.5	3892	-	1521

Figure 9: FreeCell results

though it is quite likely that post processing techniques could be used to help reduce length by removing wasteful parts of the plan. This for example has been done effectively in the planning by rewriting framework (Ambite and Knoblock, 2001). In such frameworks, simply finding a sub-optimal solution quickly is a key requirement that our learning approaches help facilitate.

Interestingly, greedy action selection according to both the learned decision list policies and measures of progress is unable to solve any testing problem and very few training problems. This indicates that the learned policies and measures have significant flaws. Yet, when incorporated into our proposed search approach, they are still able to improve planning performance. This shows that even flawed control knowledge can be effectively used in our framework, assuming it provides some amount of useful guidance.

The **DL** system took a significant amount of time to learn in all of the domains, on the order of hours. These domains have more predicates, actions and larger arities for each action than Blocksworld, leading to a bigger policy search space. Still, the benefit of the policy cannot be ignored, since once it is learned, the execution of a policy is much faster than measures of progress or heuristic functions. This is verified by the solution time. **DL** consumes the least planning time among all the learning systems. Finally, we see that for all the domains, the use of the relaxed-plan-based features improved performance as exhibited by the better performance of the systems **DL**, **MoP**, and **H** compared to **DL-noRP**, **MoP-noRP**, and **H-noRP** respectively.

Our approach to incorporating policies into search appears to help speed-up the search process by adding nodes that are far from the current node being expanded, helping to overcome local minima of the heuristic function. To help illustrate this, Figures 10 and 11 show the heuristic value trace during the search in problem 20 of Depot, where **RPL** performed poorly and **DL** solved the problem quickly. The figures plot the heuristic values of each newly expanded node. Figure 10 shows the heuristic trace for best-first search as used by **RPL**. The search stays at heuristic value 40 for more than 10000 node expansion, stuck in some local minima. Rather, Figure 11 shows that **DL**, may have been stuck in a local minima from node expansion 5 to 8, but quickly finds its way out and finds the goal after only 16 node expansions.

In contrast, recall that incorporating our learned policies into search in the Freecell domain hurt performance. To help understand this, we again plotted the heuristic trace during the search in Figures 12 and 13. As is the case with Depots problem 20, the learned policy leads to a heuristic value jump (though small), but the jump did not help and we conjecture that the jump has led the

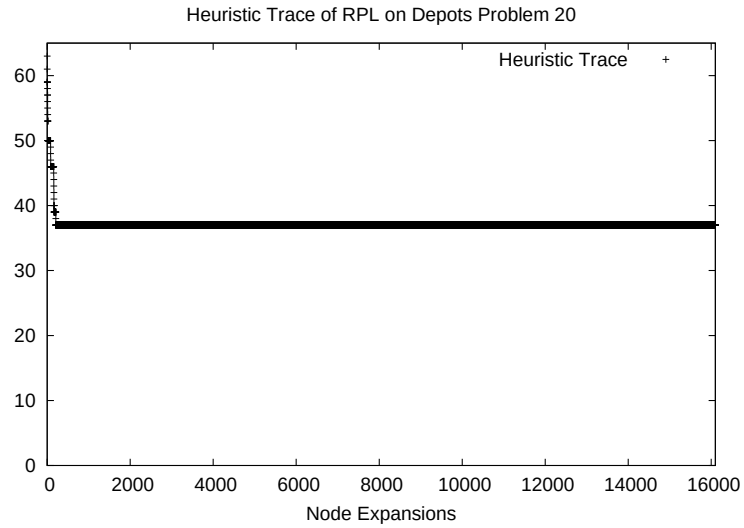


Figure 10: Heuristic trace of RPL on Depots problem 20: The RPL search found some local minimum heuristic around 38 at about 200th node expansion and remain in that region for over 15000 search nodes.

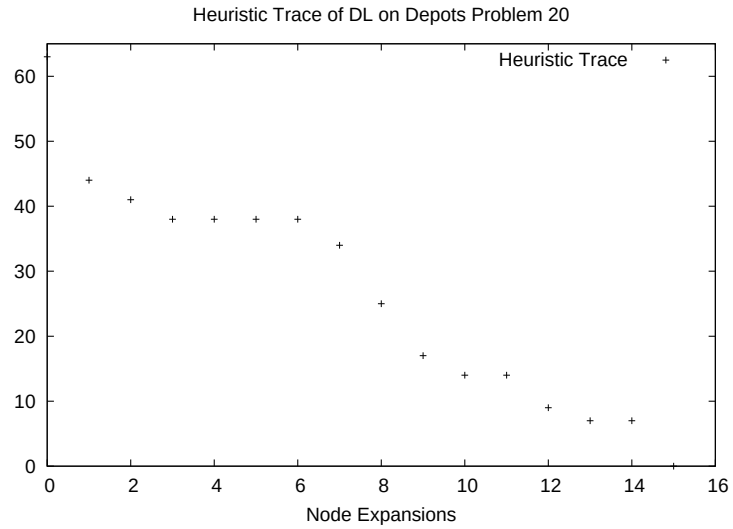


Figure 11: Heuristic trace of DL on Depots problem 20: The DL search hit a local minimum from node expansion 5 to 8, but quickly found a way out of it and reached the goal in just 16 node expansions.

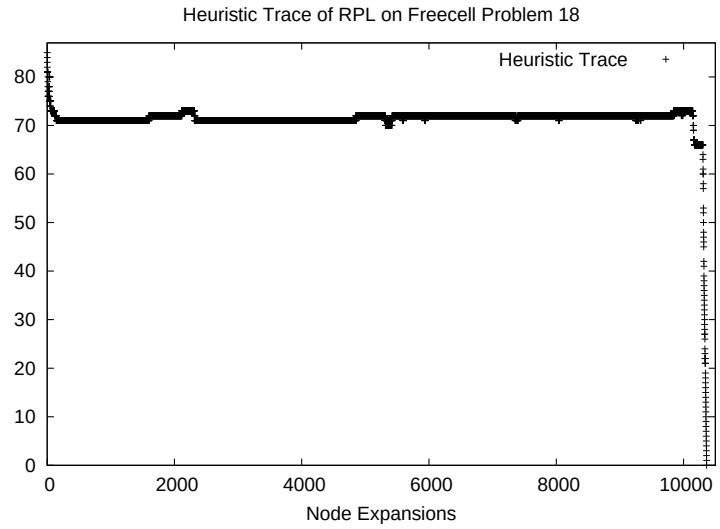


Figure 12: Heuristic trace of RPL on FreeCell problem 18: The RPL search hit a local minimum of around 70 for a long sequence of node expansions, but in the end, the search found a way out after 10000 node expansions.

search to some local minimum for both the heuristic and policy, which would not necessarily be visited with the heuristic alone, causing poor performance here for the policy-based approach.

8.4 IPC4 Results

IPC4 includes seven STRIPS/ADL domains: Airport, Satellite, Pipesworld, Pipesworld-with-Tankage, Philosophers (Promela), Optical Telegraph (Promela), and PSR (middle-compiled). FF's heuristic is very accurate for the first two domains, where for all of the solved problems the solution length and FF's heuristic are almost identical, leaving little room for learning. Thus, we only give results for the latter five domains. Each domain includes either 48 or 50 problems, giving a total of 33 or 35 testing problems, using the 15 lower numbered problems as training examples. Figures 14, 15, 16, 17, 18 present the results.

For the IPC4 results, we show the performance of the competition's best performer in each domain, labeled as **Best**. Note that the best planner varies from domain to domain. The CPU times for the best performer were taken from the official competition results, and thus are not exactly comparable to the CPU times of the other systems which were run on our local machine. In order to provide a rough comparison between the times reported for **Best** from the IPC4 results, and the times on our own machine, we ran Yahsp (Vidal, 2004) on our machine for a number of benchmark problems. In most cases, the times were quite similar. For example, on problems 40 and 45 from Pipesworld with Tankage the IPC4 results reported CPU times of 112.08 and 60.45, while we recorded times of 127.35 and 68.76 on our machine

Overall, as for the IPC4 domains, our learning systems **DL** and **H** solved more problems than FF or **RPL**, showing that the learning and planning approaches are useful. **DL** and **H** solved 127

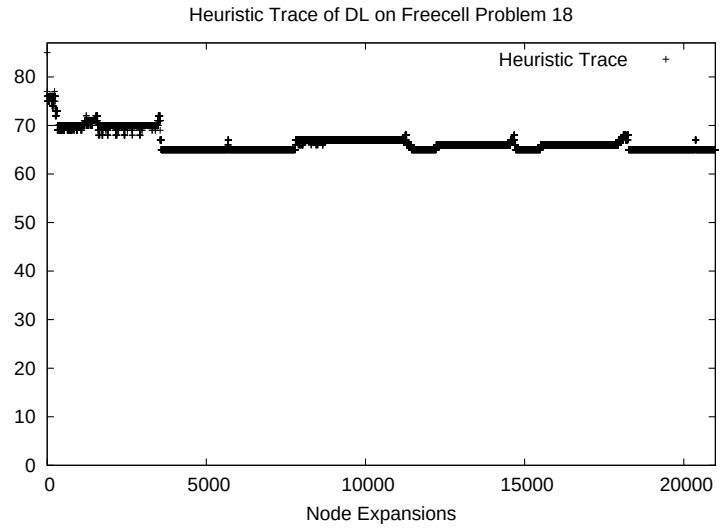


Figure 13: Heuristic trace of DL on FreeCell problem 18: Compared to Figure 12, DL search found lower heuristic states faster but did not find a way out of it. The states may not necessarily been found by RPL search.

Pipesworld (IPC4)						
Techniques	Solved (35)	Time	Length	LTime	Greedy	Evaluated
FF	21	71.2	48.2	-	-	63077
RPL	15	71.3	48.2	-	-	26830
Best	35	4.94	74.6	-	-	-
DL	28	129	76.7	18H	0	37778
MoP	25	155	79.2	1000	0	48460
H	24	17.6	98.4	709	-	22448
DL-noRP	26	10.6	67.7	12H	0	8453
MoP-noRP	23	137	87.6	870	0	67733
H-noRP	15	276	319	685	-	134366

Figure 14: Pipesworld Results

problems among 171 testing problems from all of the domains while FF solved 42 and **RPL** solved 41. Quite surprisingly, **DL** and **H** were even able to outperform the collective results of the best performers, which solved a total of 105 problems. The **MoP** system did not perform as well as the other learners. In PSR, **MoP** was unable to learn any monotonic properties, and so was not even run. In Philosophers and Optimal Telegraph, **MoP** did find monotonic properties, but those properties hurt performance. Compared to **DL** and **H**, **DL-noRP** and **H-noRP** significantly underperformed. **DL-noRP** solved 37 and **H-noRP** solved 45 showing the usefulness of the relaxed plan feature space.

Pipesworld-with-Tankage (IPC4)						
Techniques	Solved (35)	Time	Length	LTime	Greedy	Evaluated
FF	4	532	62	-	-	120217
RPL	4	333	62	-	-	100952
Best	28	221	165	-	-	-
DL	16	124	86	36H	0	49455
MoP	14	422	138	1200	0	101833
H	12	281	39	2091	-	137996
DL-noRP	10	553	52	33H	0	159113
MoP-noRP	13	287	116	1038	0	77926
H-noRP	7	441	719	1970	-	153108

Figure 15: Pipesworld Tankage Results

PSR (IPC4)						
Techniques	Solved (35)	Time	Length	LTime	Greedy	Evaluated
FF	17	692	108	-	-	13706
RPL	22	710	116	-	-	12829
Best	18	134	111	-	-	-
DL	17	736	102	1800	0	15448
MoP	-	-	-	-	-	-
H	25	568	109	2848	-	3189
DL-noRP	11	795	94	1530	0	26022
MoP-noRP	-	-	-	-	-	-
H-noRP	23	565	296	2685	-	4807

Figure 16: PSR Results

Finally, note that for most of these domains, greedy execution of the learned policies does not solve any problems. Again, however, our approach to incorporating the policies into search is still able to exploit them for significant benefits.

9. Discussion and Future Work

This study provided two primary contributions. First, we introduced a novel feature space for representing control knowledge based on extracting features from relaxed plans. Second, we showed how to learn and use control knowledge over this feature space for forward state-space heuristic search planning, a planning framework for which little work has been done in the direction of learning. We have shown that the combined approach is competitive with state-of-the-art planners across a wide range of benchmark problems. To the best of our knowledge, no prior learning-to-plan system has competed this well across such a wide set of benchmarks.

One natural extension to the relaxed-plan feature space introduced in this paper is to consider properties based on the temporal structure of relaxed plans. This could be accomplished by extend-

Philosophers (IPC4)						
Techniques	Solved (33)	Time	Length	LTime	Greedy	Evaluated
FF	0	-	-	-	-	-
RPL	0	-	-	-	-	-
Best	14	0.2	258	-	-	-
DL	33	2.59	363	3H	33	727
MoP	0	-	-	-	0	-
H	33	58.2	363	340	-	30325
DL-noRP	0	-	-	-	-	-
MoP-noRP	0	-	-	-	-	-
H-noRP	0	-	-	-	-	-

Figure 17: Philosophers Results

Optical Telegraph (IPC4)						
Techniques	Solved (33)	Time	Length	LTime	Greedy	Evaluated
FF	0	-	-	-	-	-
RPL	0	-	-	-	-	-
Best	10	721	387	-	-	-
DL	33	501	594	1H	33	930
MoP	0	-	-	-	-	-
H	33		594	826	-	9777
DL-noRP	0	-	-	-	-	-
MoP-noRP	0	-	-	-	-	-
H-noRP	0	-	-	-	-	-

Figure 18: Optical Telegraph Results

ing our current feature language to include temporal modalities. Regarding the learning of heuristics, our learning approach reduces the problem to one of standard function approximation. There are a number of ways in which we might further improve the quality of the learned heuristic. One approach would be to use ensemble-learning techniques such as bagging (Breiman, 1996), where we learn and combine multiple heuristic functions. Another more interesting extension would be to develop a learning technique that explicitly considers the search behavior of the heuristic, focusing on parts of the state space that need improvement the most. An initial step in this direction has been considered by Xu et al. (2007).

A key problem in applying learned control knowledge in planning is to robustly deal with imperfect knowledge resulting from the statistical nature of the learning process. Here we have shown one approach to help overcome imperfect policies by incorporating them into a search process. However, there are many other planning settings and forms of control knowledge for which we are interested in developing robust mechanisms for applying control knowledge. For example, stochastic planning domains and planning with richer cost functions are of primary interest. Learning control knowledge for regression-based planners is also of interest—it is not clear how the forms of knowledge

we learn here could be used in a regression based setting. As another example, we are interested in robust methods of incorporating control knowledge into SAT-based planners. Huang et al. (2000) have considered an approach to learning and incorporating constraints into a SAT-based planner. However, the approach has not been widely evaluated and it appears relatively easy for imperfect knowledge to make the planner incomplete by ruling out possible solutions.

Another research direction is to consider extending the relaxed-plan feature space to stochastic planning domains. Here one might determinize the stochastic domain in one or more ways and compute the corresponding relaxed plans. The resulting features could then be used to learn policies or value functions, using an approach such as approximate policy iteration (Fern et al., 2006). It would also be interesting to consider extending our approach for incorporating imperfect policies into search in the context of real-time dynamic programming (Barto et al., 1995).

In summary, we have demonstrated that it is possible to use machine learning to improve the performance of forward state-space search planners across a range of planning domains. However, the results are still far from the performance of human-written control knowledge in most domains, for example, TL-Plan (Bacchus and Kabanza, 2000) and SHOP (Nau et al., 1999). Also the results have still not shown large performance gains over state-of-the-art non-learning systems. Demonstrating this level of performance should be a key goal of future work in learning-based planning systems.

Acknowledgments

We thank Subbarao Kambhampati for valuable discussion and support. This work was supported in part by NSF grant IIS-0546867, DARPA contract FA8750-05-2-0249 and the DARPA Integrated Learning Program (through a sub-contract from Lockheed Martin).

References

Ricardo Aler, Daniel Borrajo, and Pedro Isasi. Using genetic programming to learn and improve control knowledge. *Artificial Intelligence Journal*, 141(1-2):29–56, 2002.

Jose Luis Ambite and Craig Knoblock. Planning by rewriting. *Journal of Artificial Intelligence Research*, 15:207–261, 2001.

Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence Journal*, 16:123–191, 2000.

Andy Barto, Steven Bradtke, and Satinder Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

J. Benton, Minh Do, and Subbarao Kambhampati. Oversubscription planning with metric goals. In *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling*, 2006.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

Avrim Blum and Merrick Furst. Fast planning through planning graph analysis. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1636–1642, 1995.

Blai Bonet and Hector Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33, 2001.

Adi Botea, Markus Enzenberger, Martin Muller, and Jonathan Schaeffer. Macro-FF: Improving AI planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research*, 24:581–621, 2005.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Daniel Bryce and Subbarao Kambhampati. Planning graph heuristics for belief space search. *Journal of Artificial Intelligence Research*, (26):35–99, 2006.

Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

Andrew I. Coles and Amanda J. Smith. Marvin: A heuristic search planner with online macro-action learning. *Journal of Artificial Intelligence Research*, 28:119–156, February 2007. ISSN 11076-9757.

Minh Do and Subbarao Kambhampati. Sapa: A scalable multi-objective heuristic metric temporal planner. *Journal of Artificial Intelligence Research*, (20):155–194, 2003.

Saso Dzeroski, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. *Machine Learning Journal*, 43:7–52, 2001.

Tara A. Estlin and Rymond J. Mooney. Multi-strategy learning of search control for partial-order planning. In *Proceedings of 13th National Conference on Artificial Intelligence*, 1996.

Alan Fern, Sungwook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias: Solving relational markov decision processes. *Journal of Artificial Intelligence Research*, 25:85–118, 2006.

Richard Fikes, Peter Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence Journal*, 3(1–3):251–288, 1972.

Maria Fox and Derek Long. The automatic inference of state invariants in TIM. *Journal of Artificial Intelligence Research*, 9:367–421, 1998.

Johannes Furnkranz and Peter A. Flach. An analysis of rule evaluation metrics. In *Proceedings of Twentieth International Conference on Machine Learning*, 2003.

Alfonso Gerevini and Lenhart K. Schubert. Discovering state constraints in DISCOPLAN: Some new results. In *Proceedings of National Conference on Artificial Intelligence*, pages 761–767. AAAI Press / The MIT Press, 2000.

William D. Harvey and Matthew L. Ginsberg. Limited discrepancy search. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1995, August 20-25 1995.

Malte Helmert, Patrik Haslum, and Jrg Hoffmann. Flexible abstraction heuristics for optimal sequential planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling*, 9 2007.

Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:263–302, 2001.

Yi-Cheng Huang, Bart Selman, and Henry Kautz. Learning declarative control rules for constraint-based planning. In *Proceedings of Seventeenth International Conference on Machine Learning*, pages 415–422, 2000.

Roni Khardon. Learning action strategies for planning domains. *Artificial Intelligence Journal*, 113 (1-2):125–148, 1999.

Mario Martin and Hector Geffner. Learning generalized policies in planning domains using concept languages. In *Proceedings of Seventh International Conference on Principles of Knowledge Representation and Reasoning*, 2000.

David McAllester and Robert Givan. Taxonomic syntax for first-order inference. *Journal of the ACM*, 40:246–283, 1993.

Steve Minton, Jaime Carbonell, Craig A. Knoblock, Daniel R. Kuokka, Oren Etzioni, and Yolanda Gil. Explanation-based learning: A problem solving perspective. *Artificial Intelligence Journal*, 40:63–118, 1989.

Steven Minton. Quantitative results concerning the utility of explanation-based learning. In *Proceedings of National Conference on Artificial Intelligence*, 1988.

Steven Minton, editor. *Machine Learning Methods for Planning*. Morgan Kaufmann Publishers, 1993.

Dana Nau, Yue Cao, Amnon Lotem, and Héctor Munoz-Avila. Shop: Simple hierarchical ordered planner. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 968–973, 1999.

XuanLong Nguyen, Subbarao Kambhampati, and Romeo Sanchez Nigenda. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. *Artificial Intelligence*, 135(1-2):73–123, 2002.

Aarati Parmar. A logical measure of progress for planning. In *Proceedings of Eighteenth National Conference on Artificial Intelligence*, pages 498–505. AAAI Press, July 2002.

Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

Vincent Vidal. A lookahead strategy for heuristic search planning. In *International Conference on Automated Planning and Scheduling*, 2004.

Yuehua Xu, Alan Fern, and Sungwook Yoon. Discriminative learning of beam-search heuristics for planning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.

Sungwook Yoon, Alan Fern, and Robert Givan. Inductive policy selection for first-order MDPs. In *Proceedings of Eighteenth Conference in Uncertainty in Artificial Intelligence*, 2002.

Sungwook Yoon, Alan Fern, and Robert Givan. Learning measures of progress for planning domains. In *Proceedings of Twentieth National Conference on Artificial Intelligence*, 2005.

Terry Zimmerman and Subbarao Kambhampati. Learning-assisted automated planning: Looking back, taking stock, going forward. *AI Magazine*, 24(2)(2):73–96, 2003.

Journal of Machine Learning Research 9 (2008) 719-758 Submitted 1/07; Revised 1/08; Published 4/08

Multi-class Discriminant Kernel Learning via Convex Programming

Jieping Ye JIEPING.YE@ASU.EDU
Shuiwang Ji SHUIWANG.JI@ASU.EDU
Jianhui Chen JIANHUI.CHEN@ASU.EDU
Department of Computer Science and Engineering
Center for Evolutionary Functional Genomics
The Biodesign Institute
Arizona State University
Tempe, AZ 85287, USA

Editor: Isabelle Guyon and Amir Saffari

Abstract

Regularized kernel discriminant analysis (RKDA) performs linear discriminant analysis in the feature space via the kernel trick. Its performance depends on the selection of kernels. In this paper, we consider the problem of multiple kernel learning (MKL) for RKDA, in which the optimal kernel matrix is obtained as a linear combination of pre-specified kernel matrices. We show that the kernel learning problem in RKDA can be formulated as convex programs. First, we show that this problem can be formulated as a semidefinite program (SDP). Based on the equivalence relationship between RKDA and least square problems in the binary-class case, we propose a convex quadratically constrained quadratic programming (QCQP) formulation for kernel learning in RKDA. A semi-infinite linear programming (SILP) formulation is derived to further improve the efficiency. We extend these formulations to the multi-class case based on a key result established in this paper. That is, the multi-class RKDA kernel learning problem can be decomposed into a set of binary-class kernel learning problems which are constrained to share a common kernel. Based on this decomposition property, SDP formulations are proposed for the multi-class case. Furthermore, it leads naturally to QCQP and SILP formulations. As the performance of RKDA depends on the regularization parameter, we show that this parameter can also be optimized in a joint framework with the kernel. Extensive experiments have been conducted and analyzed, and connections to other algorithms are discussed.

Keywords: model selection, kernel discriminant analysis, semidefinite programming, quadratically constrained quadratic programming, semi-infinite linear programming

1. Introduction

Formulation of machine learning problems as convex programs has been one of the recent trends in machine learning research. Such formulations offer global solutions and avoid some difficulties encountered by traditional learning algorithms (Lanckriet et al., 2003, 2004b; d'Aspremont et al., 2007). Kernel methods (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004) work by embedding the input data into some high-dimensional feature space, and they are generally formulated as convex optimization problems. The key fact underlying the success of kernel methods is that the embedding into feature space can be determined uniquely by specifying a kernel function that computes the dot product between data points in the feature space. In other words, the kernel

function implicitly defines the nonlinear mapping to the feature space, and expensive computations in the high-dimensional feature space can be avoided by evaluating the kernel function. Thus, one of the central issues in kernel methods is the selection of kernels.

To automate kernel-based learning algorithms, it is desirable to integrate the tuning of kernels into the learning process. This problem has been addressed from different perspectives recently. Lanckriet et al. (2004b) pioneered the work of multiple kernel learning (MKL) in which the optimal kernel matrix is obtained as a linear combination of pre-specified kernel matrices. It was shown (Lanckriet et al., 2004b) that the coefficients in MKL can be determined by solving convex programs in the case of support vector machines (SVM) (Vapnik, 1998; Cristianini and Taylor, 2000). This MKL problem was formulated as support kernel machines (SKM) in Bach et al. (2004), and the sequential minimal optimization (SMO) algorithm (Platt, 1999) was proposed to solve it. Recently, this SKM was reformulated as semi-infinite linear program (SILP) which was shown to be scalable to large data sets and a large number of kernels (Sonnenburg et al., 2006; Rakotomamonjy et al., 2007). Micchelli and Pontil (2005, 2007) studied the problem of finding an optimal kernel from a prescribed convex set of kernels by regularization. To deal with problems with structured output, MKL for joint feature map was proposed in Zien and Ong (2007). While most existing work focuses on learning kernels for SVM, Fung et al. (2004) proposed to learn kernels for discriminant analysis. Based on ideas from MKL, this problem was reformulated as SDP in Kim et al. (2006). In general, approaches based on learning linear combination of kernel matrices offer the additional advantage of facilitating heterogeneous data integration from multiple sources. Such formulations have been applied to combine various biological data, for example, amino acid sequences, hydropathy profiles, and gene expression data, for enhanced biological inference (Lanckriet et al., 2004a).

Ong et al. (2005) showed that the learning of kernels can be accomplished by defining a reproducing kernel Hilbert space on the space of kernels itself, and the resulting optimization problem is an SDP. This formulation was recast into second order cone program (SOCP) (Lobo et al., 1998) in Tsang and Kwok (2006). Hoi et al. (2007) showed that the kernel matrix can be learned in a nonparametric manner by solving SDP. The kernel learning problem in the context of multiple tasks was considered in Jebara (2004). Some recent extensions of kernel learning produced nonstationary combinations (Lewis et al., 2006) and potentially infinite number of kernels (Argyriou et al., 2006).

This paper addresses the issue of kernel learning for regularized kernel discriminant analysis (RKDA) (Mika et al., 1999; Baudat and Anouar, 2000; Mika et al., 2001, 2003). Our proposed methods belong to the MKL framework, and they can thus be used for heterogeneous data integration. We systematically extend the kernel learning problem for RKDA in several directions. First, we extend the formulation in Kim et al. (2006) by proposing a simplified SDP formulation. Based on this simplified form and the equivalence relationship between KRDA and least square problems in the binary-class case, we propose a convex quadratically constrained quadratic programming (QCQP) formulation for this problem. To improve the efficiency of our formulations, we further develop a semi-infinite linear programming (SILP) formulation. While most existing work on kernel learning only deals with binary-class problems, we show that all of our formulations can be extended naturally to the multi-class setting. In particular, we show that the kernel learning problem for multi-class RKDA can be decomposed into a set of binary-class kernel learning problems that are constrained to share a common kernel. It is worth noting that the optimal kernel is the same for the original and the decomposed formulations, though the optimal transformation matrices may not coincide. In other words, the decomposed form is equivalent to the original one for the purpose of kernel learning. We further develop an approximate scheme to reduce the computational cost of

multi-class SDP formulation. Finally, we propose to optimize the regularization parameter along with the kernels in a joint framework. This joint optimization framework is derived from and similar to the work in De Bie et al. (2003); Lanckriet et al. (2004b).

The key contributions of this paper can be highlighted as follows:

- We propose a simplified SDP formulation for the RKDA kernel learning problem in the binary-class case. Based on this simplified form and the equivalence relationship between RKDA and least square problems in the binary-class case, we derive QCQP and SILP formulations for this problem.

- We show that the multi-class RKDA kernel learning problem can be decomposed into k binary-class kernel learning problems where k is the number of classes. This leads to two (exact and approximate) SDP formulations in the multi-class case. Based on this decomposition property, we show that the QCQP and SILP formulations for binary-class problems can be extended naturally to the multi-class case.

- We show that all the proposed formulations can be recast to optimize the regularization parameter simultaneously. This joint learning framework further automates the learning algorithms.

- We conduct extensive experiments using a collection of benchmark data sets to compare several relevant algorithms under a unified experimental setup. To demonstrate the effectiveness of the proposed formulations for heterogeneous data integration, we apply these formulations to combine multiple data sources derived from gene expression pattern images (Tomancak et al., 2002).

The rest of this paper is organized as follows: We derive the SDP, QCQP, and SILP formulations for the binary-class case in Section 2. Section 3 extends these formulations to the multi-class case. The joint optimization of regularization parameter is presented in Section 4. Section 5 presents the experimental evaluation, and this paper concludes with discussion and conclusion in Section 6.

Notation

$x \in \mathbb{R}^n$ denotes an n-dimensional vector. Similarly, $A \in \mathbb{R}^{m \times n}$ denotes a matrix with m rows and n columns. I is used to denote the identity matrix of an appropriate dimension and e_m denotes the vector of all ones of length m. For a square symmetric matrix S, $S \succeq 0$ means it is positive semidefinite. We also use the short-hand $x \geq 0$ to denote that each component of the vector x is non-negative.

2. Convex Formulations for Binary-class Problems

We use $\mathcal{X}$ to denote the input or instance space, which is a subspace of $\mathbb{R}^d$, and $\mathcal{Y} = \{-1, +1\}$ to denote the output or class label set. An input-output pair (x, y), where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, is called an example. An example is called positive (negative) if its class label is $+1\,(-1)$. We assume that the examples are drawn randomly and independently from a fixed, but unknown, underlying probability distribution over $\mathcal{X} \times \mathcal{Y}$.

A symmetric function $K : \mathcal{X} \times \mathcal{X} \to R$ is called a kernel function (Schölkopf and Smola, 2002) if it satisfies the finitely positive semidefinite property. That is, for any $x_1, \cdots, x_m \in \mathcal{X}$, the *Gram*

matrix $G \in \mathbb{R}^{m\times m}$, defined by $G_{ij} = K(x_i, x_j)$ is positive semidefinite. Any kernel function K implicitly maps the input set $\mathcal{X}$ to a high-dimensional (possibly infinite) Hilbert space $\mathcal{H}_K$ equipped with the inner product $(\cdot,\cdot)_{\mathcal{H}_K}$ through a mapping ϕ_K from $\mathcal{X}$ to $\mathcal{H}_K$:

$$K(x,z) = (\phi_K(x), \phi_K(z))_{\mathcal{H}_K}.$$

In kernel-based classification, the algorithms learn a classifier $f : \mathcal{X} \to \{-1,+1\}$ whose decision boundary between the two classes is affine in the feature space $\mathcal{H}_K$:

$$f(x) = sgn(w^T \phi_K(x) + b),$$

where $w \in \mathcal{H}_K$ is the vector of feature weights, $b \in \mathbb{R}$ is the intercept, and $sgn(u) = +1$, if $u > 0$, and -1 otherwise.

Let $\{x_1^+, \cdots, x_{m_+}^+\}$ and $\{x_1^-, \cdots, x_{m_-}^-\}$ denote the collections of data points from the positive and negative classes, respectively. The total number of data points in the training set is $m = m_+ + m_-$. For a given kernel function K, the basic idea of RKDA in the binary-class case is to find a direction in the feature space $\mathcal{H}_K$ onto which the projections of the two sets $\{\phi_K(x_i^+)\}_{i=1}^{m_+}$ and $\{\phi_K(x_i^-)\}_{i=1}^{m_-}$ are well separated. Define the centroids of the two classes as follows:

$$\mu_K^+ = \frac{1}{m_+}\sum_{i=1}^{m_+}\phi_K(x_i^+),$$
$$\mu_K^- = \frac{1}{m_-}\sum_{i=1}^{m_-}\phi_K(x_i^-),$$

and the two sample class covariance matrices as follows:

$$S_K^+ = \frac{1}{m_+}\sum_{i=1}^{m_+}(\phi_K(x_i^+) - \mu_K^+)(\phi_K(x_i^+) - \mu_K^+)^T,$$
$$S_K^- = \frac{1}{m_-}\sum_{i=1}^{m_-}(\phi_K(x_i^-) - \mu_K^-)(\phi_K(x_i^-) - \mu_K^-)^T.$$

Specifically, in RKDA the separation between the two classes is measured by the ratio of the variance $(w^T(\mu_K^+ - \mu_K^-))^2$ between the classes to the variance $w^T\left(m_+/mS_K^+ + m_-/mS_K^-\right)w$ within the classes. Thus, RKDA maximizes the following objective function:

$$F_1(w,K) = \frac{(w^T(\mu_K^+ - \mu_K^-))^2}{w^T\left(m_+/mS_K^+ + m_-/mS_K^- + \lambda I\right)w}, \tag{1}$$

where $\lambda > 0$ is the regularization parameter. The optimal weight vector

$$w^* \equiv \underset{w}{\operatorname{argmax}}\{F_1(w,K)\}$$

that maximizes the objective function in Equation (1) for a fixed kernel function K and a fixed regularization parameter λ is given by

$$w^* = (m_+/mS_K^+ + m_-/mS_K^- + \lambda I)^{-1}(\mu_K^+ - \mu_K^-).$$

The maximum value

$$F_1^*(K) \equiv \max_w \{F_1(w,K)\}$$

of the objective function in Equation (1) achieved by the optimal weight vector w^* is given by

$$F_1^*(K) = (\mu_K^+ - \mu_K^-)^T \left(m_+/mS_K^+ + m_-/mS_K^- + \lambda I\right)^{-1} (\mu_K^+ - \mu_K^-). \tag{2}$$

It follows from the Representer Theorem (Schölkopf and Smola, 2002) that the optimal weight vector in RKDA is in the span of the images of the training points in the feature space. In other words, there exists a vector

$$\alpha^* = [\alpha_1^+, \cdots, \alpha_{m_+}^+, \alpha_1^-, \cdots, \alpha_{m_-}^-]^T \in \mathbb{R}^m$$

such that

$$w^* = \sum_{i=1}^{m_+} \alpha_i^+ \phi_K(x_i^+) + \sum_{i=1}^{m_-} \alpha_i^- \phi_K(x_i^-) = \phi_K(X)\alpha^*,$$

where $\phi_K(X)$ is the data matrix in the feature space given by

$$\phi_K(X) = \left[\phi_K(x_1^+), \cdots, \phi_K(x_{m_+}^+), \phi_K(x_1^-), \cdots, \phi_K(x_{m_-}^-)\right].$$

The optimal vector α^* is given by Kim et al. (2006) as

$$\alpha^* = \frac{1}{\lambda}(I - J(\lambda I + JGJ)^{-1}JG)a,$$

where I is the identity matrix, a is an m-dimensional vector given by

$$a = [1/m_+, \cdots, 1/m_+, -1/m_-, \cdots, -1/m_-]^T \in \mathbb{R}^m, \tag{3}$$

the matrix J is defined as:

$$J = \begin{pmatrix} \frac{1}{\sqrt{m_+}}(I - \frac{1}{m_+} e_{m_+} e_{m_+}^T) & 0 \\ 0 & \frac{1}{\sqrt{m_-}}(I - \frac{1}{m_-} e_{m_-} e_{m_-}^T) \end{pmatrix},$$

G is restricted to be a linear combination of the p given kernel matrices $G_1, \cdots, G_p$ as

$$G \in \mathcal{G} = \left\{ G = \sum_{i=1}^p \theta_i G_i \;\middle|\; \sum_{i=1}^p \theta_i = 1\,, \quad \theta_i \geq 0 \quad \forall i \right\},$$

and e_{m_+} and e_{m_-} are vectors of all ones of length m^+ and m^-, respectively.

The optimal value $F_1^*(K)$ in Equation (2) is thus given by

$$\begin{aligned} F_1^*(K) &= (\mu_K^+ - \mu_K^-)^T \left(m_+/mS_K^+ + m_-/mS_K^- + \lambda I\right)^{-1} (\mu_K^+ - \mu_K^-) \\ &= (\mu_K^+ - \mu_K^-)^T w^* = (\mu_K^+ - \mu_K^-)^T \phi_K(X)\alpha^* = a^T \phi_K(X)^T \phi_K(X)\alpha^* \\ &= \frac{1}{\lambda} a^T G(I - J(\lambda I + JGJ)^{-1}JG)a. \end{aligned} \tag{4}$$

It was shown in Kim et al. (2006) that the optimal Gram matrix G based on the kernel function K that maximizes $F_1^*(K)$ given in Equation (4) can be obtained by solving a semidefinite program (SDP)

(Vandenberghe and Boyd, 1996; Boyd and Vandenberghe, 2004). General-purpose optimization packages such as SeDuMi (Sturm, 1999) use the interior-point methods (Nesterov and Nemirovskii, 1994) to solve SDP. However, for problems of moderate size in machine learning, this overhead of optimal kernel learning is large, and its computation time can easily exceed that of the learning algorithm itself.

We propose a new SDP formulation for this problem in the next subsection. The proposed SDP formulation has a simplified form. Experimental results presented in Section 5 show that the proposed formulation is comparable to the one in Kim et al. (2006). More importantly, this simplified formulation lays the foundation for the extensions to multi-class problems in Section 3 and the joint optimization of regularization parameter in Section 4.

2.1 Simplified SDP Formulation

In the rest of this paper, we work on the centered version of kernel matrices. This is equivalent to centering the data as preprocessed in linear discriminant analysis (LDA) and principal component analysis (PCA). More precisely, given a set of p kernel matrices $G_1, \cdots, G_p$, the proposed algorithms learn an optimal kernel matrix $\tilde{G} \in \tilde{\mathcal{G}}$, where

$$\tilde{\mathcal{G}} = \left\{ \tilde{G} = \sum_{i=1}^{p} \theta_i \tilde{G}_i \;\middle|\; \sum_{i=1}^{p} \theta_i r_i = 1, \;\; \theta_i \geq 0 \right\},$$

$\tilde{G}_i = PG_iP$, $r_i = \mathrm{trace}(\tilde{G}_i)$, and $P \in \mathbb{R}^{m \times m}$ is the centering matrix defined as

$$P = I - \frac{1}{m} e_m e_m^T, \tag{5}$$

and e_m is the vector of all ones of size m.

Consider the maximization of the following objective function:

$$F_2(w,K) = \frac{(w^T(\mu_K^+ - \mu_K^-))^2}{w^T(\Sigma_K + \lambda I)w}, \tag{6}$$

where Σ_K is defined as follows:

$$\begin{aligned} \Sigma_K &= m_+ S_K^+ + m_- S_K^- + \frac{m_+ m_-}{m} (\mu_K^+ - \mu_K^-)(\mu_K^+ - \mu_K^-)^T \\ &= \sum_{i=1}^{m_+} (\phi_K(x_i^+) - \mu_K)(\phi_K(x_i^+) - \mu_K)^T + \sum_{i=1}^{m_-} (\phi_K(x_i^-) - \mu_K)(\phi_K(x_i^-) - \mu_K)^T \\ &= \phi_K(X) P \phi_K(X)^T, \end{aligned} \tag{7}$$

P is defined in Equation (5), and

$$\mu_K = \frac{1}{m} \left(\sum_{i=1}^{m_+} \phi_K(x_i^+) + \sum_{i=1}^{m_-} \phi_K(x_i^-) \right)$$

is the global centroid of the data in the feature space. Note that the scaling factor $1/m$ has been omitted in the definition of Σ_K in Equation (7). It turns out that for fixed K and λ, Equations (1) and (6) are equivalent in terms of the computation of the optimal weight vector w. We show in the following theorem that optimizing $F_2(w,K)$ in Equation (6) with respect to the kernel function leads to a simplified SDP formulation.

Theorem 2.1 *Given a set of p centered kernel matrices $\tilde{G}_1, \cdots, \tilde{G}_p$, the optimal kernel matrix $\tilde{G} \in \tilde{\mathcal{G}}$ that maximizes the objective function in Equation (6) can be found by solving the following semidefinite programming problem:*

$$\begin{array}{lll} \min\limits_{\theta,t} & t & \\ \textit{subject to} & \begin{pmatrix} I + \frac{1}{\lambda}\sum_{i=1}^{p} \theta_i \tilde{G}_i & a \\ a^T & t \end{pmatrix} \succeq 0, & (8) \\ & \theta \geq 0, & \\ & \theta^T r = 1, & \end{array}$$

where a is defined in Equation (3), $\theta = [\theta_1, \cdots, \theta_p]^T$, *and* $r = \left[trace(\tilde{G}_1), \cdots, trace(\tilde{G}_p)\right]^T$.

Proof The optimal weight vector

$$w^* \equiv \underset{w}{\text{argmax}} \{F_2(w,K)\}$$

is given by

$$w^* = (\Sigma_K + \lambda I)^{-1}(\mu_K^+ - \mu_K^-).$$

The maximum value of the objective function in Equation (6) achieved by w^* is given by

$$F_2^*(K) \equiv F_2(w^*,K) = (\mu_K^+ - \mu_K^-)^T (\Sigma_K + \lambda I)^{-1}(\mu_K^+ - \mu_K^-).$$

It follows from Appendix A that

$$w^* = \frac{1}{\lambda}\phi_K(X)\left(I - P\left(\lambda I + PGP\right)^{-1} PG\right) a,$$

and

$$\begin{aligned} F_2^*(K) &= (\mu_K^+ - \mu_K^-)^T w^* = a^T \phi_K(X)^T w^* \\ &= \frac{1}{\lambda} a^T \left(G - GP\left(\lambda I + PGP\right)^{-1} PG\right) a. \end{aligned}$$

Since the vector a defined in Equation (3) is of zero mean, that is, $Pa = a$, we have

$$\begin{aligned} F_2^*(K) &= \frac{1}{\lambda} a^T P \left(G - GP(\lambda I + PGP)^{-1} PG\right) Pa \\ &= \frac{1}{\lambda} a^T \left(\tilde{G} - \tilde{G}(\lambda I + \tilde{G})^{-1}\tilde{G}\right) a, \end{aligned} \tag{9}$$

where $\tilde{G}$ is derived from G with both rows and columns centered as

$$\tilde{G} = PGP.$$

Since

$$\begin{aligned} \tilde{G} - \tilde{G}(\lambda I + \tilde{G})^{-1}\tilde{G} &= \tilde{G} - \tilde{G}(\lambda I + \tilde{G})^{-1}(\tilde{G} + \lambda I - \lambda I) \\ &= \lambda \tilde{G}(\lambda I + \tilde{G})^{-1} \\ &= \lambda(\tilde{G} + \lambda I - \lambda I)(\lambda I + \tilde{G})^{-1} \\ &= \lambda - \lambda^2(\lambda I + \tilde{G})^{-1}, \end{aligned}$$

the optimal value $F_2^*(K)$ in Equation (9) can be simplified as

$$F_2^*(K) = a^T a - \lambda a^T (\lambda I + \tilde{G})^{-1} a. \tag{10}$$

It follows that the optimal kernel learning problem in RKDA, which maximizes $F_2^*(K)$ in Equation (10) for a fixed regularization parameter λ, is equivalent to minimizing

$$\lambda a^T (\lambda I + \tilde{G})^{-1} a = a^T \left(I + \frac{1}{\lambda} \tilde{G} \right)^{-1} a, \tag{11}$$

subject to the constraint that $\tilde{G} \in \tilde{\mathcal{G}}$.

Mathematically, the optimal kernel learning problem can be formulated as follows:

$$\begin{aligned} \min_{\theta} \quad & a^T \left(I + \frac{1}{\lambda} \sum_{i=1}^{p} \theta_i \tilde{G}_i \right)^{-1} a \\ \text{subject to} \quad & \theta \geq 0, \\ & \theta^T r = 1. \end{aligned}$$

We can write the inequality

$$a^T \left(I + \frac{1}{\lambda} \tilde{G} \right)^{-1} a \leq t$$

equivalently as the linear matrix inequality (LMI) (Boyd and Vandenberghe, 2004)

$$\begin{pmatrix} I + \frac{1}{\lambda} \tilde{G} & a \\ a^T & t \end{pmatrix} \succeq 0,$$

via the Schur complement lemma (Golub and Van Loan, 1996; Lanckriet et al., 2004b). We complete the proof by a simple change of variable. ■

2.2 QCQP Formulation

The optimization problem proposed by Kim et al. (2006) and the one in Theorem 2.1 are both SDP problems, which are computationally very expensive to solve, even with the recent advances in interior point methods. In this subsection, we show that this kernel learning problem can be reformulated equivalently as a quadratically constrained quadratic program (QCQP) (Boyd and Vandenberghe, 2004), which can then be solved more efficiently than SDP.

It is known that discriminant analysis and least square problems are equivalent in the binary-class case (Mika, 2002). Consider the regularized least squares problem, which minimizes the following objective function:

$$F_3(w,K) = ||(\phi_K(X)P)^T w - a||^2 + \lambda ||w||^2. \tag{12}$$

The following lemma relates this problem to the problem of optimal kernel learning.

Lemma 2.1 *The optimal kernel function K solving the optimization problem in Equation (11) is also the minimizer of the objective function in Equation (12).*

Proof The optimal w^* that minimizes the objective function in Equation (12) for a fixed K and λ is given by

$$\begin{aligned} w^* &= \left(\lambda I + \phi_K(X) P \phi_K(X)^T\right)^{-1} \phi_K(X) P a \\ &= \frac{1}{\lambda}\phi_K(X)\left(I - P\left(\lambda I + PGP\right)^{-1} PG\right) a. \end{aligned}$$

The optimal value of the objective function in Equation (12) is therefore given by

$$F_3^*(K) = a^T \left(I + \frac{1}{\lambda}\tilde{G}\right)^{-1} a,$$

where $\tilde{G} = PGP$. This completes the proof of this lemma. ■

Based on this equivalence result, we can formulate the kernel learning problem as a QCQP problem, as summarized in the following theorem.

Theorem 2.2 *Given a set of p centered kernel matrices $\tilde{G}_1, \cdots, \tilde{G}_p$, the optimal kernel matrix, in the form of a convex linear combination of the given p kernel matrices, that minimizes the objective function in Equation (12) can be found by solving the following convex QCQP problem:*

$$\begin{aligned} \max_{\beta, t} \quad & -\frac{1}{4}\beta^T\beta + \beta^T a - \frac{1}{4\lambda}t \\ \textit{subject to} \quad & t \geq \frac{1}{r_i}\beta^T \tilde{G}_i \beta, \ \textit{for } i = 1, \cdots, p, \end{aligned} \tag{13}$$

where $r_i = trace(\tilde{G}_i)$.

Proof We consider the dual formulation of the minimization of $F_3(w,K)$ in terms of w. Denote

$$\eta = (\phi_K(X)P)^T w - a.$$

It follows that

$$F_3(w,K) = ||\eta||^2 + \lambda||w||^2.$$

Define the Lagrangian function of the following optimization problem:

$$\begin{aligned} \min_{w,\eta} \quad & F_3(w,K) = ||\eta||^2 + \lambda||w||^2 \\ \text{subject to} \quad & \eta = (\phi_K(X)P)^T w - a \end{aligned}$$

as follows:

$$L(\eta, w, \beta) = ||\eta||^2 + \lambda||w||^2 - \beta^T((\phi_K(X)P)^T w - a - \eta),$$

where β is the vector of Lagrangian dual variables. Taking the derivatives of $L(\eta, w, \beta)$ with respect to η and w and setting them equal to zero, we get

$$\begin{aligned} \frac{\partial L(\eta, w, \beta)}{\partial \eta} &= 2\eta + \beta = 0, \\ \frac{\partial L(\eta, w, \beta)}{\partial w} &= 2\lambda w - \phi_K(X)P\beta = 0. \end{aligned}$$

It follows that

$$\eta = -\frac{\beta}{2}, \text{ and } w = \frac{\phi_K(X)P\beta}{2\lambda}.$$

Thus, we obtain the following Lagrangian dual function:

$$g(\beta) = \min_{w,\eta} L(\eta, w, \beta) = -\frac{1}{4}\beta^T \left(I + \frac{1}{\lambda} PGP \right) \beta + \beta^T a.$$

The optimal β^* is computed by maximizing $g(\beta)$ as

$$\beta^* = \operatorname*{argmax}_{\beta} g(\beta) = \operatorname*{argmax}_{\beta} \left\{ -\frac{1}{4}\beta^T \left(I + \frac{1}{\lambda} PGP \right) \beta + \beta^T a \right\}.$$

Since strong duality holds, the optimal kernel is given by solving the following optimization problem:

$$\min_{\tilde{G} \in \tilde{\mathcal{G}}} \max_{\beta} \left\{ -\frac{1}{4}\beta^T \left(I + \frac{1}{\lambda} \tilde{G} \right) \beta + \beta^T a \right\}.$$

We can rewrite the above optimization problem as

$$\begin{aligned}
& \min_{\theta:\theta\geq 0, \theta^T r = 1} \max_{\beta} \left\{ -\frac{1}{4}\beta^T \left(I + \frac{1}{\lambda} \sum_{i=1}^{p} \theta_i \tilde{G}_i \right) \beta + \beta^T a \right\} \qquad (14) \\
= & \max_{\beta} \min_{\theta:\theta\geq 0, \theta^T r = 1} \left\{ -\frac{1}{4}\beta^T \left(I + \frac{1}{\lambda} \sum_{i=1}^{p} \theta_i \tilde{G}_i \right) \beta + \beta^T a \right\} \\
= & \max_{\beta} \min_{\theta:\theta\geq 0, \theta^T r = 1} \left\{ -\frac{1}{4\lambda} \sum_{i=1}^{p} \theta_i \beta^T \tilde{G}_i \beta - \frac{1}{4}\beta^T \beta + \beta^T a \right\} \\
= & \max_{\beta} \left\{ -\frac{1}{4}\beta^T \beta + \beta^T a - \frac{1}{4\lambda} \max_{\theta:\theta\geq 0, \theta^T r = 1} \left(\sum_{i=1}^{p} \theta_i \beta^T \tilde{G}_i \beta \right) \right\} \\
= & \max_{\beta} \left\{ -\frac{1}{4}\beta^T \beta + \beta^T a - \frac{1}{4\lambda} \max_{i} \left(\frac{1}{r_i} \beta^T \tilde{G}_i \beta \right) \right\}. \qquad (15)
\end{aligned}$$

The exchange of minimization and maximization in deriving the second equation from the first holds since the objective function is convex in θ and concave in β, the minimization problem is strictly feasible in θ and the maximization problem is strictly feasible in β. Therefore, Slater's condition (Boyd and Vandenberghe, 2004) follows and strong duality holds (Lanckriet et al., 2004b; Boyd and Vandenberghe, 2004). By simply changing the last term in Equation (15) to t and moving it to the constraint, we prove this theorem. ■

Note that general-purpose optimization software packages like SeDuMi (Sturm, 1999) and MOSEK (Andersen and Andersen, 2000) employ the interior point methods, and they solve the primal and dual problems simultaneously. Thus, the coefficients, $\theta_1, \cdots, \theta_p$, can be obtained directly from the dual variables.

The formulation in Equation (13) is a quadratically constrained quadratic program (QCQP), which is a special form of second order cone program (SOCP) (Lobo et al., 1998; Alizadeh and

Goldfarb, 2003) and SDP. Theoretical results on interior point method (Nesterov and Nemirovskii, 1994) show that QCQP can be solved more efficiently than SDP, and it is therefore more scalable to large-scale problems. Similar ideas have been used in Lanckriet et al. (2004b) to learn a non-negative linear combination of kernel matrices.

2.3 SILP Formulation

Semi-infinite programming (SIP) (Hettich and Kortanek, 1993) refers to optimization problems that seek the maximum of the function $F(z)$ subject to a system of constraints on z, expressed as $g(z,t) \leq 0$, for all t in some set B. When both the objective and constraints are linear (and hence convex), it is known as semi-infinite linear programming (SILP). We show in this section that the kernel learning problem for RKDA can be formulated as an SILP problem, as summarized in the following theorem.

Theorem 2.3 *Given a set of p centered kernel matrices $\tilde{G}_1, \cdots, \tilde{G}_p$, the optimal kernel matrix, in the form of a convex linear combination of the given p kernel matrices, that maximizes the objective function in Equation (12) can be found by solving the following SILP problem:*

$$\begin{aligned} \max_{\theta,\gamma} \quad & \gamma && (16) \\ \textit{subject to} \quad & \theta \geq 0, \\ & \theta^T r = 1, \\ & \sum_{i=1}^{p} \theta_i S_i(\beta) \geq \gamma, \quad \textit{for all } \beta, && (17) \end{aligned}$$

where $S_i(\beta)$ is defined as

$$S_i(\beta) = \frac{r_i}{4}\beta^T\beta + \frac{1}{4\lambda}\beta^T\tilde{G}_i\beta - r_i\beta^T a, \quad \textit{for } i = 1, \cdots, p, \tag{18}$$

$r = (r_1, \cdots, r_p)^T$, and $r_i = trace(\tilde{G}_i)$.

Proof It follows from the definition of $S_i(\beta)$ in Equation (18) that the optimization problem in Equation (14) can be expressed equivalently as

$$\begin{aligned} \max_{\theta} \min_{\beta} \quad & \sum_{i=1}^{p} \theta_i S_i(\beta) && (19) \\ \text{subject to} \quad & \theta \geq 0, \\ & \theta^T r = 1. \end{aligned}$$

Assume β^* is the optimal solution to the problem in Equation (19) and define $\gamma^* = \sum_{i=1}^{p} \theta_i S_i(\beta^*)$ as the minimum objective value achieved by β^*. We have

$$\sum_{i=1}^{p} \theta_i S_i(\beta) \geq \gamma^*, \quad \text{for all } \beta.$$

By defining

$$\gamma = \min_{\beta} \sum_{i=1}^{p} \theta_i S_i(\beta)$$

and substituting γ into the objective, we prove this theorem. ■

Note that the optimization problem in Equation (16) is an SILP since both θ and γ are linearly constrained, and there are an infinite number of constraints, one for each possible value of β. As in Sonnenburg et al. (2006), we propose to use the *column generation* technique to solve this SILP problem. In this technique, the optimal θ and γ are computed for a restricted subset of constraints in Equation (17) and this problem is called the *restricted master problem*. Constraints that are not satisfied by current θ and γ are added successively to the restricted master problem until all constraints are satisfied. For fast convergence of the algorithm, it is desirable to add constraint that maximizes the violation for current θ and γ. That is, the β value that solves

$$\beta_\theta = \operatorname*{argmin}_\beta \sum_{i=1}^{p} \theta_i S_i(\beta), \tag{20}$$

is desired. If $\sum_{i=1}^{p} \theta_i S_i(\beta^\theta) \geq \gamma$, then all the constraints are satisfied, and θ and γ reach their optimal values. Otherwise, this constraint is added to the restricted master problem and the iteration continues.

It follows from the definition of $S_i(\beta)$ in Equation (18) that the problem in Equation (20) can be written as

$$\min_\beta \left\{ \frac{1}{4}\beta^T\beta + \frac{1}{4\lambda}\beta^T \left(\sum_{i=1}^{p} \theta_i \tilde{G}_i \right) \beta - \beta^T a \right\}. \tag{21}$$

For a fixed θ, the problem in Equation (21) is an unconstrained convex quadratic program whose solution can be obtained analytically. To avoid computing matrix inverse, we obtain β by solving the following system of linear equations:

$$\left(\frac{1}{2} I + \frac{1}{2\lambda} \sum_{i=1}^{p} \theta_i \tilde{G}_i \right) \beta = a.$$

After β is computed, the corresponding constraint is added to the restricted master problem to obtain the intermediate θ and γ. Note that the restricted master problem is a linear program. Thus, the proposed algorithm for solving the SILP problem proposed in Theorem 2.3 alternates between solving a linear system and a linear program. In contrast, the SILP formulation proposed in Sonnenburg et al. (2006) for SVM kernel learning involves solving a constrained quadratic program (QP) and a linear program. They shown that the constrained QP coincides with a single kernel SVM formulation, and thus existing software for solving SVM can be used directly.

The alternating algorithm for solving the proposed SILP problem belongs to a family of algorithms for solving general SIP problems called the *exchange methods*, in which the constraints are exchanged at each iteration. It follows from Theorem 7.2 in Hettich and Kortanek (1993) that these methods are guaranteed to converge. Similar to the convergence criterion used in Sonnenburg et al. (2006), the algorithm returns when

$$\left| 1 - \frac{\sum_{i=1}^{p} \theta_i^{(t-1)} S_i(\beta^{(t)})}{\gamma^{(t-1)}} \right| \leq \varepsilon, \tag{22}$$

where $\theta_i^{(t-1)}$, for $i = 1, \cdots, p$, and $\gamma^{(t-1)}$ are the optimal solutions to the restricted master problem at the $(t-1)$-th iteration, $\beta^{(t)}$ is the β value that maximizes the constraint violation at the t-th iteration, and ε is a user-specified tolerance parameter. We set $\varepsilon = 5 \times 10^{-4}$ in our experiments.

2.4 Time Complexity Analysis

We analyze the time complexity of the proposed formulations for the binary-class case. It follows from the analysis in Lanckriet et al. (2004b) that the proposed SDP and QCQP formulations have the worst-case time complexity of $O\left((p+n)^2 n^{2.5}\right)$ and $O\left(pn^2+n^3\right)$, respectively, where p is the number of candidate kernels and n is the number of training samples. The algorithm to solve the proposed SILP formulation alternates between solving a linear program (LP) and a linear system of equations. The LP formulation involved has a simple structure and its computation time is small, especially when p is much smaller than n. Note that the number of constraints in the LP depends on the number of iterations. Our experiments show that the algorithm converges within a small number of iterations. Thus, the time complexity of the SILP formulation is dominated by the time in solving the linear system which has a complexity of $O\left(n^3\right)$. Overall, the SILP formulation has a worst-case time complexity of $O\left(n^3 Ite\right)$ where Ite is the number of iterations.

All formulations discussed in Lanckriet et al. (2004b), Kim et al. (2006) and Sonnenburg et al. (2006) are constrained to binary-class problems. We show in the next section that our formulations in this section can be extended naturally to the multi-class case.

3. Convex Formulations for Multi-class Problems

In the multi-class case, we are given a data set that consists of m samples $\{(x_i,y_i)\}_{i=1}^m$, where $x_i \in \mathbb{R}^d$, and $y_i \in \{1,2,\cdots,k\}$ denotes the class label of the i-th sample, and $k>2$. Similar to the binary-class case, let $X=[x_1,\cdots,x_m]$ be the data matrix.

In the multi-class RKDA formulation, the maximization of the following objective function is commonly used (Ye, 2005):

$$F_4(W,K) = \text{trace}\left(\left(W^T\left(\Sigma_K+\lambda I\right)W\right)^{-1} W^T B_K W\right), \tag{23}$$

where W is the transformation matrix, and B_K, the so-called between-class scatter matrix is defined as

$$B_K = \phi_K(X) H H^T \phi_K(X)^T,$$

$H=[h_1,h_2,\cdots,h_k]$, and h_i is a vector whose j-th entry is given by

$$h_i(j) = \begin{cases} \sqrt{\frac{n}{n_j}} - \sqrt{\frac{n_j}{n}} & \text{if the } j\text{-th data point belongs to the } i\text{-th class} \\ -\sqrt{\frac{n_j}{n}} & \text{otherwise.} \end{cases} \tag{24}$$

The optimal W is given by computing the eigenvectors of the following matrix:

$$\left(\Sigma_K+\lambda I\right)^{-1} B_K.$$

Since the weight vectors are in the span of the images of the data points in the feature space, we can express W as $W=\phi_K(X)A$ for some matrix $A \in \mathbb{R}^{m\times\ell}$, where $A=[\alpha_1,\cdots,\alpha_\ell]$. Then

$$F_4(W,K) = \text{trace}\left(\left(A^T\left(GPG+\lambda G\right)A\right)^{-1} A^T GHH^T GA\right).$$

Define two matrices S_t^K and S_b^K as follows:

$$\begin{aligned} S_t^K &= GPG+\lambda G, \\ S_b^K &= GHH^T G. \end{aligned}$$

Since the null space of S_t^K lies in the null space of S_b^K (Ye and Xiong, 2006), there exists a nonsingular matrix Z such that

$$\begin{aligned} Z^T S_t^K Z &= \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}, \\ Z^T S_b^K Z &= \begin{pmatrix} \Sigma_b & 0 \\ 0 & 0 \end{pmatrix}, \end{aligned}$$

where Σ_b is diagonal with the diagonal entries sorted in non-decreasing order. The optimal A^* is given by

$$A^* = Z_q = [z_1, \cdots, z_q],$$

where Z_q consists of the first q columns of Z, and $q = \text{rank}(S_b^K)$. It follows that the optimal value of $F_4(W,K)$ achieved by the optimal A^* is given by

$$F_4^*(K) = \text{trace}(\Sigma_b) = \text{trace}\left(\left(S_t^K\right)^{-1} S_b^K\right). \tag{25}$$

Here we have assumed that $S_t^K = GPG + \lambda G$ is nonsingular. We could use the pseudo-inverse to deal with the singular case, and all the following arguments still follow.

Thus, in the multi-class case, the optimal kernel function K can be computed by maximizing $F_4^*(K)$ in Equation (25), which is however highly nonlinear and difficult to solve. In the following, we present an equivalent formulation as the one in Equation (25), which is more tractable computationally.

3.1 SDP Formulation

Consider the maximization of the following objective function:

$$F_5(W,K) = \sum_{i=1}^{k} \frac{(w_i^T \phi_K(X) h_i)^2}{w_i^T (\Sigma_K + \lambda I) w_i}, \tag{26}$$

where

$$W = [w_1, \cdots, w_k]$$

is the transformation matrix, and h_i is defined in Equation (24). The following lemma shows that the optimal kernel function K coincides for F_4 and F_5.

Lemma 3.1 *Let F_4 and F_5 be defined as in Equation (23) and Equation (26), respectively. Let W^* and K^* be the optimal solution to the following optimization problem:*

$$\max_K \max_W F_4(W,K), \tag{27}$$

and let $\tilde{W}^*$ and $\tilde{K}^*$ be the optimal solution to the following optimization problem:

$$\max_K \max_W F_5(W,K). \tag{28}$$

Then $K^* = \tilde{K}^*$.

Proof Since $W = \phi_K(X)A$, we have $w_i = \phi_K(X)\alpha_i$ and

$$F_5(W,K) = \sum_{i=1}^{k} \frac{(\alpha_i^T G h_i)^2}{\alpha_i^T (GPG + \lambda G)\alpha_i} = \sum_{i=1}^{k} \frac{(\alpha_i^T G h_i)^2}{\alpha_i^T S_t^K \alpha_i}.$$

The computation of α_i and α_j for $i \neq j$ is independent of each other when the kernel function K and λ are fixed. The optimal α_i^* is given by

$$\alpha_i^* = \left(S_t^K\right)^{-1} G h_i.$$

It follows that the maximum value of $F_5(W,K)$ achieved by the optimal $A^* = [\alpha_1^*, \cdots, \alpha_k^*]$ is given by

$$F_5^*(K) = \sum_{i=1}^{k} (G h_i)^T \left(S_t^K\right)^{-1} G h_i.$$

Based on the properties of matrix trace, we have

$$\begin{aligned}
F_5^*(K) &= \sum_{i=1}^{k} (G h_i)^T \left(S_t^K\right)^{-1} G h_i \\
&= \sum_{i=1}^{k} \text{trace}\left((G h_i)^T \left(S_t^K\right)^{-1} G h_i\right) \\
&= \sum_{i=1}^{k} \text{trace}\left(\left(S_t^K\right)^{-1} G h_i (G h_i)^T\right) \\
&= \text{trace}\left(\left(S_t^K\right)^{-1} \sum_{i=1}^{k} \left(G h_i h_i^T G^T\right)\right) \\
&= \text{trace}\left(\left(S_t^K\right)^{-1} \left(GHH^T G^T\right)\right) \\
&= \text{trace}\left(\left(S_t^K\right)^{-1} S_b^K\right) \\
&= F_4^*(K).
\end{aligned}$$

This completes the proof. ■

It is interesting to note that, in general, the optimal W^* and $\tilde{W}^*$ for the optimization problems in Equations (27) and (28) are different. However, it has been shown recently that, when the value of the regularization parameter is approaching zero, multi-class regularized least squares is equivalent to multi-class discriminant analysis under a mild condition (Ye, 2007). Empirical evidences show that when the value of the regularization parameter is small, which is usually the case in practice, their performance is similar.

The objective function in Equation (26) is closely related to its binary counterpart in Equation (6). Note that a variant of the Fisher discriminant ratio (FDR) (Kim et al., 2006) can be written as:

$$F_2(w,K) = \frac{(w^T \phi_K(X)\, a)^2}{w^T (\Sigma_K + \lambda I) w}.$$

Thus, $F_5(W,K)$ in Equation (26) can be interpreted as the weighted summation of the FDRs between the samples in the i-th class and the rest where $i = 1,\cdots,k$. The weights can be computed from the definition of H in Equation (24) as follows:

$$h_i = \begin{bmatrix} \vdots \\ \sqrt{\frac{n}{n_i}} - \sqrt{\frac{n_i}{n}} \\ \vdots \\ -\sqrt{\frac{n_i}{n}} \\ \vdots \end{bmatrix} = (n-n_i)\sqrt{\frac{n_i}{n}} \begin{bmatrix} \vdots \\ \frac{1}{n_i} \\ \vdots \\ -\frac{1}{n-n_i} \\ \vdots \end{bmatrix} = (n-n_i)\sqrt{\frac{n_i}{n}}\, a^{(i)},$$

where $a^{(i)}$ is obtained from Equation (3) by taking the samples from the i-th class as positive and the rest as negative. It follows that the weight for the i-th binary classification problem is: $(n-n_i)^2 n_i/n$.

Following the results from the last section for the binary-class case, the optimal kernel learning problem for multi-class RKDA can be formulated as follows:

$$\begin{aligned} \min_{t_1,\cdots,t_k,\theta} \quad & \sum_{j=1}^{k} t_j \\ \text{subject to} \quad & \begin{pmatrix} I + \frac{1}{\lambda}\sum_{i=1}^{p} \theta_i \tilde{G}_i & h_j \\ h_j^T & t_j \end{pmatrix} \succeq 0, \text{ for } j = 1,\cdots,k, \\ & \theta \geq 0, \\ & \theta^T r = 1. \end{aligned} \tag{29}$$

Unfortunately, the SDP problem given in Equation (29) is computationally prohibitive due to the presence of positive semidefinite constraints. To alleviate this computational problem, we put all the constraints in a single larger constraint. This imposes stronger constraints than those on the original problem, but the computational cost can be reduced dramatically. It is based on the following lemma.

Lemma 3.2 *Let $M \in \mathbb{R}^{m\times m}$ be any positive definite matrix, $a_1,\cdots,a_k \in \mathbb{R}^m$, $t_1,\cdots,t_k \in \mathbb{R}$. Then*

$$\begin{pmatrix} M & a_1 & a_2 & \cdots & a_k \\ a_1^T & t_1 & 0 & \cdots & 0 \\ a_2^T & 0 & t_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_k^T & 0 & 0 & \cdots & t_k \end{pmatrix} \succeq 0 \tag{30}$$

implies

$$\begin{pmatrix} M & a_j \\ a_j^T & t_j \end{pmatrix} \succeq 0, \text{ for all } j. \tag{31}$$

Proof For a symmetric and positive semidefinite matrix, it is known that all of its principal submatrices are also symmetric and positive semidefinite. Matrices in Equation (31) are all principal submatrices of the matrix in Equation (30). This can be seen by removing 2 to j and $j+2$ to $k+1$ rows and columns of the block matrix in Equation (30). This completes the proof of the lemma. ■

We summarize the main result of this section in the following theorem:

Theorem 3.1 *Given a set of p centered kernel matrices $\tilde{G}_1, \cdots, \tilde{G}_p$, the optimal kernel matrix, in the form of linear combination of the given p kernel matrices, that maximizes the objective function in Equation (26) can be found by solving the SDP problem in Equation (29). This problem can be approximated by the following more restricted formulation:*

$$\begin{aligned} \min_{t_1,\cdots,t_k,\theta} \quad & \sum_{j=1}^{k} t_j \\ \textit{subject to} \quad & \begin{pmatrix} I + \frac{1}{\lambda}\sum_{i=1}^{p}\theta_i\tilde{G}_i & h_1 & h_2 & \cdots & h_k \\ h_1^T & t_1 & 0 & \cdots & 0 \\ h_2^T & 0 & t_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h_k^T & 0 & 0 & \cdots & t_k \end{pmatrix} \succeq 0, \\ & \theta \geq 0, \\ & \theta^T r = 1, \end{aligned} \tag{32}$$

where $r_i = trace(\tilde{G}_i)$. The optimal solution to the formulation in Equation (32) satisfies the constraints in Equation (29).

The formulation in Equation (32) is an approximation to the exact formulation in Equation (29). We use the approximate formulation in our experiments in Section 5, and empirical results show that it achieves comparable performance with other exact formulations.

3.2 QCQP Formulation

Although the approximate SDP formulation in the last section is scalable in terms of the number of classes, interior point algorithms for solving SDP have an inherently large time complexity, and thus it cannot be applied to large-scale problems. In this subsection, we propose a QCQP formulation which is more efficient than its SDP counterpart. The derivations here are similar to those in Section 2.2.

In order to formulate the multi-class RKDA kernel learning problem into a QCQP problem, we first consider the minimization of the following objective function:

$$F_6(W,K) = \sum_{i=1}^{k} \left(||(\phi_K(X)P)^T w_i - h_i||^2 + \lambda ||w_i||^2 \right), \tag{33}$$

where $W = [w_1, \cdots, w_k]$. It is clear that for a fixed K and λ, the computation of w_i and w_j for $i \neq j$ is independent of each other. By extending the results from Lemma 2.1 and Lemma 3.1, it is easy to show that the optimal kernel function K minimizing the objective function in Equation (26) coincides the minimizer of $F_6(W,K)$ in Equation (33). Motivated by this equivalence result, we derive an efficient QCQP formulation for the multi-class RKDA kernel learning problem, as summarized in the following theorem.

Theorem 3.2 *Given a set of p centered kernel matrices $\tilde{G}_1, \cdots, \tilde{G}_p$, the optimal kernel matrix, in the form of a convex linear combination of the given p kernel matrices, that minimizes the objective*

function in Equation (33) can be found by solving the following convex QCQP problem:

$$\begin{aligned} \max_{\beta_1,\cdots,\beta_k,t} \quad & \sum_{j=1}^{k} \beta_j^T h_j - \frac{1}{4}\sum_{j=1}^{k} \beta_j^T \beta_j - \frac{1}{4\lambda} t \\ \textit{subject to} \quad & t \geq \frac{1}{r_i}\sum_{j=1}^{k} \beta_j^T \tilde{G}_i \beta_j,\ i = 1,\cdots,p. \end{aligned} \tag{34}$$

where $r_i = trace(\tilde{G}_i)$.

Proof We first consider the dual formulation of the minimization of $F_6(W,K)$ in terms of W for fixed K and λ. Denote

$$\eta_i = (\phi_K(X)P)^T w_i - h_i.$$

It follows that

$$F_6(w,K) = \sum_{i=1}^{k} ||\eta_i||^2 + \lambda \sum_{i=1}^{k} ||w_i||^2.$$

Define the Lagrangian function of this problem as follows:

$$L(\{\eta_i\}_{i=1}^k, w, \{\beta_i\}_{i=1}^k) = \sum_{i=1}^{k} ||\eta_i||^2 + \lambda \sum_{i=1}^{k} ||w_i||^2 - \sum_{i=1}^{k} \beta_i^T \left((\phi_K(X)P)^T w_i - h_i - \eta_i\right),$$

where the β_i's are the vectors of Lagrangian dual variables. Taking the derivatives of L with respect to η_i and w_i for all i, and setting them equal to zero, we get

$$\begin{aligned} \frac{\partial L}{\partial \eta_i} &= 2\eta_i + \beta_i = 0, \\ \frac{\partial L}{\partial w_i} &= 2\lambda w_i - \phi_K(X)P\beta_i = 0. \end{aligned}$$

Thus, we have

$$\eta_i = -\frac{\beta_i}{2}, \text{ and } w_i = \frac{\phi_K(X)P\beta_i}{2\lambda},$$

and we obtain the following Lagrangian dual function:

$$\begin{aligned} g(\beta_1,\cdots,\beta_k) &= \min_{w_i,\eta_i,i=1,\cdots,k} L(\{\eta_i\}_{i=1}^k, w, \{\beta_i\}_{i=1}^k) \\ &= \sum_{i=1}^{k} \left(-\frac{1}{4}\beta_i^T \left(I + \frac{1}{\lambda} PGP\right)\beta_i + \beta_i^T h_i\right). \end{aligned} \tag{35}$$

The optimal $\beta_1^*,\cdots,\beta_k^*$ can be computed by maximizing $g(\beta_1,\cdots,\beta_k)$ in Equation (35) as

$$(\beta_1^*,\cdots,\beta_k^*) = \operatorname*{argmax}_{\beta_1,\cdots,\beta_k} \left\{ \sum_{i=1}^{k} \left(-\frac{1}{4}\beta_i^T \left(I + \frac{1}{\lambda} PGP\right)\beta_i + \beta_i^T h_i\right)\right\}.$$

Since strong duality holds, the optimal kernel matrix $\tilde{G}$ is given by solving the following optimization problem:

$$\min_{\tilde{G}\in\tilde{\mathcal{G}}} \max_{\beta_1,\cdots,\beta_k} \left\{ \sum_{i=1}^{k} \left(-\frac{1}{4}\beta_i^T \left(I + \frac{1}{\lambda}\tilde{G}\right)\beta_i + \beta_i^T h_i\right)\right\}.$$

Similar to the binary-class case, the above optimization problem can be written as

$$\begin{aligned}
& \min_{\theta:\theta\geq 0,\theta^T r=1} \max_{\beta_1,\cdots,\beta_k} \left\{ \sum_{j=1}^{k} \left(-\frac{1}{4}\beta_j^T \left(I + \frac{1}{\lambda}\sum_{i=1}^{p} \left(\theta_i \tilde{G}_i\right) \right) \beta_j + \beta_j^T h_j \right) \right\} \qquad (36)\\
= & \max_{\beta_1,\cdots,\beta_k} \min_{\theta:\theta\geq 0,\theta^T r=1} \left\{ \sum_{j=1}^{k} \left(-\frac{1}{4}\beta_j^T \left(I + \frac{1}{\lambda}\sum_{i=1}^{p} \theta_i \tilde{G}_i \right) \beta_j + \beta_j^T h_j \right) \right\} \\
= & \max_{\beta_1,\cdots,\beta_k} \min_{\theta:\theta\geq 0,\theta^T r=1} \left\{ -\frac{1}{4}\sum_{j=1}^{k} \beta_j^T \beta_j - \frac{1}{4\lambda}\sum_{i=1}^{p} \theta_i \left(\sum_{j=1}^{k} \beta_j^T \tilde{G}_i \beta_j \right) + \sum_{j=1}^{k} \beta_j^T h_j \right\} \\
= & \max_{\beta_1,\cdots,\beta_k} \left\{ \sum_{j=1}^{k} \beta_j^T h_j - \frac{1}{4}\sum_{j=1}^{k} \beta_j^T \beta_j - \frac{1}{4\lambda} \max_{\theta:\theta\geq 0,\theta^T r=1} \left\{ \sum_{i=1}^{p} \theta_i \left(\sum_{j=1}^{k} \beta_j^T \tilde{G}_i \beta_j \right) \right\} \right\} \\
= & \max_{\beta_1,\cdots,\beta_k} \left\{ \sum_{j=1}^{k} \beta_j^T h_j - \frac{1}{4}\sum_{j=1}^{k} \beta_j^T \beta_j - \frac{1}{4\lambda} \max_i \left(\frac{1}{r_i} \sum_{j=1}^{k} \beta_j^T \tilde{G}_i \beta_j \right) \right\}.
\end{aligned}$$

By constraining

$$\max_i \left(\frac{1}{r_i} \sum_{j=1}^{k} \beta_j^T \tilde{G}_i \beta_j \right) \leq t$$

and putting t in the objective function, we prove the formulation in Equation (34). ■

3.3 SILP Formulation

The QCQP formulation in Theorem 3.2 has a worse-case time complexity of $O(pk^2n^2 + k^3n^3)$, which is cubic in terms of the number of classes and the number of data points. We show in this subsection that the RKDA kernel learning problem in the multi-class case can be formulated as an SILP problem, as summarized in the following theorem.

Theorem 3.3 *Given a set of p centered kernel matrices $\tilde{G}_1, \cdots, \tilde{G}_p$, the optimal kernel matrix, in the form of a convex linear combination of the given p kernel matrices, that minimizes the objective function in Equation (33) can be found by solving the following SILP problem:*

$$\begin{aligned}
\max_{\theta,\gamma} \quad & \gamma \qquad (37)\\
\textit{subject to} \quad & \theta \geq 0, \\
& \theta^T r = 1, \\
& \sum_{i=1}^{p} \theta_i S_i(\beta) \geq \gamma, \quad \textit{for all } \beta,
\end{aligned}$$

where $S_i(\beta)$ is defined as

$$S_i(\beta) = \sum_{j=1}^{k} \left(\frac{r_i}{4}\beta_j^T \beta_j + \frac{1}{4\lambda}\beta_j^T \tilde{G}_i \beta_j - r_i \beta_j^T h_j \right), \quad \textit{for } i = 1, \cdots, p, \qquad (38)$$

$r = (r_1, \cdots, r_p)^T$, and $r_i = trace(\tilde{G}_i)$.

Formulation	SDP	QCQP	SILP
Complexity	$O\left((p+n)^2(k+n)^{2.5}\right)$	$O(pk^2n^2+k^3n^3)$	$O(n^3Ite)$

Table 1: Time complexity of the proposed multi-class RKDA kernel learning formulations: p is the number of candidate kernels, n is the number of training samples, k is the number of classes, and Ite is the number of iterations in SILP.

Proof The proof follows the same procedure as in Theorem 2.3 by starting from Equation (36) and changing the definition of $S_i(\beta)$ from Equation (18) to Equation (38). ■

Note that the only difference between formulations in Theorem 2.3 and Theorem 3.3 lies in the definitions of $S_i(\beta)$. To find the β_j, for $j = 1,\cdots,k$, that maximize the constraint violation in the multi-class case, we need to solve the following k systems of linear equations:

$$\left(\frac{1}{2}I+\frac{1}{2\lambda}\sum_{i=1}^{p}\theta_i\tilde{G}_i\right)\beta_j = h_j, \quad \text{for } j = 1,\cdots,k.$$

Note that the coefficient matrix is the same for all of the k linear systems. Thus the LU decomposition (Golub and Van Loan, 1996) needs to be computed only once, and only the forward/backward substitution needs to be performed k times to obtain the solutions.

3.4 Time Complexity Analysis

In this subsection, we analyze the time complexity of the proposed formulations in the multi-class case. By following similar analysis in the binary-class case, we can show that the proposed (approximate) SDP and QCQP formulations have worse-case time complexity of $O\left((p+n)^2(k+n)^{2.5}\right)$ and $O(pk^2n^2+k^3n^3)$, respectively. For the SILP formulation in the multi-class case, the k linear systems involved in each iterative step share the same coefficient matrix, and they can be solved in $O(n^3)$ time. Thus, the overall complexity is still $O(n^3Ite)$ where Ite is the number of iterations. The complexity of multi-class RKDA kernel learning formulations is summarized in Table 1.

4. Joint Kernel and Regularization Parameter Learning

The formulations presented in the last two sections focus on the estimation of the kernel matrix only, while the regularization parameter λ is pre-specified. In some cases, the performance of RKDA algorithm depends critically on the value of λ. In this section, we show that all the formulations proposed in this paper can be reformulated equivalently, and this new formulation leads naturally to the estimation of the regularization parameter λ in a joint framework. The detailed derivations in this section are similar to those presented in Sections 2 and 3.

4.1 Joint Learning for Binary-class Problems

One key advantage of the kernel learning formulation in Equation (8) in comparison with the one in Kim et al. (2006) is that the regularization parameter λ can also be estimated in a joint optimization

framework. In particular, all the formulations (SDP, QCQP, and SILP) for the binary-class RKDA kernel learning problems, presented in Theorems 2.1–2.3, can be recast to optimize the regularization parameter simultaneously. The next three subsections provide details of these reformulations.

4.1.1 SDP FORMULATION

For the estimation of regularization parameter, we consider a slightly modified version of the regularized least squares formulation, which is equivalent to the standard formulation in Equation (12). The modified version minimizes the following objective function:

$$F_7(w,K,\tau) = \tau||(\phi_K(X)P)^T w - a||^2 + ||w||^2, \tag{39}$$

where $\tau = 1/\lambda$. We will first consider the case when τ is fixed. We will then extend to the general case when τ is optimized jointly.

The optimal w^* that minimizes the objective function in Equation (39) for a fixed K and a fixed τ is given by

$$\begin{aligned} w^* &= \left(\frac{1}{\tau}I + \phi_K(X)P\phi_K(X)^T\right)^{-1}\phi_K(X)Pa \\ &= \tau\phi_K(X)\left(I - P\left(\frac{1}{\tau}I + PGP\right)^{-1}PG\right)a. \end{aligned}$$

The optimal value of the objective function in Equation (39) is given by

$$F_7^*(K,\tau) = a^T\left(\frac{1}{\tau}I + \tilde{G}\right)^{-1}a, \tag{40}$$

where $\tilde{G} = PGP$.

We can observe from Equation (40) that the identity matrix appears in exactly the same form as other kernel matrices. We can thus treat the regularization parameter as one of the coefficients for the kernel matrix and optimize them simultaneously. This leads to the following formulation:

$$\begin{aligned} \min_{t,\tilde{\theta}} \quad & t \\ \text{subject to} \quad & \begin{pmatrix} \sum_{i=0}^p \tilde{\theta}_i\tilde{G}_i & a \\ a^T & t \end{pmatrix} \succeq 0, \\ & \tilde{\theta} \geq 0, \\ & \sum_{i=0}^p \tilde{\theta}_i \text{trace}(\tilde{G}_i) = 1, \end{aligned} \tag{41}$$

where $\tilde{\theta} = [\theta_0, \theta_1, \cdots, \theta_p]^T$, $\theta_0 = \frac{1}{\tau} = \lambda$, and $\tilde{G}_0 = I$.

4.1.2 QCQP FORMULATION

In order to cast the formulation in Theorem 2.2 to optimize the regularization parameter, we again start from the modified least square problem in Equation (39). By following the same procedure as

in Theorem 2.2, the optimization problem in Equation (15) can be expressed as

$$\begin{aligned} & \min_{\theta:\theta\geq 0,\theta^T r=1} \max_{\beta} \left\{ -\frac{1}{4}\beta^T \left(\frac{1}{\tau} I + \sum_{i=1}^{p} \theta_i \tilde{G}_i \right) \beta + \beta^T a \right\} \\ = & \max_{\beta} \min_{\tilde{\theta}:\tilde{\theta}\geq 0,\tilde{\theta}^T r=1} \left\{ -\frac{1}{4}\beta^T \left(\sum_{i=0}^{p} \tilde{\theta}_i \tilde{G}_i \right) \beta + \beta^T a \right\}, \end{aligned} \tag{42}$$

where $\theta_0 = \frac{1}{\tau}$, and $\tilde{G}_0 = I$. This can be formulated to optimize the regularization parameter as one of the coefficients for the kernel matrix as follows:

$$\begin{aligned} \max_{\beta,t} \quad & \beta^T a - \frac{1}{4} t \\ \text{subject to} \quad & t \geq \frac{1}{r_i} \beta^T \tilde{G}_i \beta, \; i = 0, \cdots, p. \end{aligned} \tag{43}$$

This problem is a quadratically constrained linear program.

4.1.3 SILP FORMULATION

The SILP formulation proposed in Theorem 2.3 for the binary-class problem can also be reformulated to optimize λ jointly. It follows from Equation (42) that this joint learning problem can be formulated as follows:

$$\begin{aligned} \max_{\tilde{\theta},\gamma} \quad & \gamma \\ \text{subject to} \quad & \tilde{\theta} \geq 0, \\ & \tilde{\theta}^T r = 1, \\ & \sum_{i=0}^{p} \theta_i S_i(\beta) \geq \gamma, \quad \text{for all } \beta, \end{aligned} \tag{44}$$

where $S_i(\beta)$ is defined as

$$S_i(\beta) = \frac{1}{4}\beta^T \tilde{G}_i \beta - r_i \beta^T a, \quad \text{for } i = 0, \cdots, p,$$

$r = (r_0, \cdots, r_p)^T$, $r_i = \text{trace}(\tilde{G}_i)$, $\tilde{\theta} = [\theta_0, \theta_1, \cdots, \theta_p]^T$, $\theta_0 = \frac{1}{\tau} = \lambda$, and $\tilde{G}_0 = I$.

4.2 Joint Learning for Multi-class Problems

All formulations for the multi-class RKDA kernel learning problems presented in Section 3 can be recast to optimize the regularization parameter jointly. The next three subsections provide details of these reformulations.

4.2.1 SDP FORMULATION

In order to incorporate λ in the optimization problem, we modify the objective function in Equation (26) as follows:

$$F_8(W,K,\tau) = \sum_{i=1}^{k} \frac{(w_i^T \phi_K(X) h_i)^2}{w_i^T (\tau \Sigma_K + I) w_i}.$$

By following the same derivation in Lemma 3.1 and noticing the relationship with the binary-class case, we derive the following SDP formulation for the multi-class RKDA kernel learning problem:

$$\begin{aligned}
\min_{t_1,\cdots,t_k,\tilde{\theta}} \quad & \sum_{j=1}^{k} t_j \\
\text{subject to} \quad & \begin{pmatrix} \sum_{i=0}^{p} \tilde{\theta}_i \tilde{G}_i & h_1 & h_2 & \cdots & h_k \\ h_1^T & t_1 & 0 & \cdots & 0 \\ h_2^T & 0 & t_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h_k^T & 0 & 0 & \cdots & t_k \end{pmatrix} \succeq 0, \\
& \tilde{\theta} \geq 0, \\
& \tilde{\theta}^T r = 1,
\end{aligned} \tag{45}$$

where $\tilde{\theta} = [\theta_0, \theta_1, \cdots, \theta_p]^T$, $\theta_0 = \frac{1}{\tau} = \lambda$, and $\tilde{G}_0 = I$.

4.2.2 QCQP FORMULATION

Similar to the binary-class case, we modify the least square problem in Equation (33) as follows:

$$F_9(W,K,\tau) = \sum_{i=1}^{k} \left(\tau ||(\phi_K(X)P)^T w_i - h_i||^2 + ||w_i||^2 \right),$$

where $\tau = 1/\lambda$. By following the same derivation as in Theorem 3.2, we obtain the following joint optimization problem:

$$\begin{aligned}
\max_{\beta_1,\cdots,\beta_k,t} \quad & \sum_{j=1}^{k} \beta_j^T h_j - \frac{1}{4} t \\
\text{subject to} \quad & t \geq \frac{1}{r_i} \sum_{j=1}^{k} \beta_j^T \tilde{G}_i \beta_j, \; i = 0, \cdots, p.
\end{aligned} \tag{46}$$

This is a quadratically constrained linear program.

4.2.3 SILP FORMULATION

Similar to the reformulation in the binary-class case, the SILP formulation for multi-class problems can also be formulated to optimize λ simultaneously as follows:

$$\begin{aligned} \max_{\tilde{\theta},\gamma} \quad & \gamma \\ \text{subject to} \quad & \tilde{\theta} \geq 0, \\ & \tilde{\theta}^T r = 1, \\ & \sum_{i=0}^{p} \theta_i S_i(\beta) \geq \gamma, \quad \text{for all } \beta, \end{aligned} \tag{47}$$

where

$$S_i(\beta) = \sum_{j=1}^{k} \left(\frac{1}{4} \beta_j^T \tilde{G}_i \beta_j - r_i \beta_j^T h_j \right), \quad \text{for } i = 0, \cdots, p,$$

$r = (r_0, \cdots, r_p)^T$, $r_i = \text{trace}(\tilde{G}_i)$, $\tilde{\theta} = [\theta_0, \theta_1, \cdots, \theta_p]^T$, $\theta_0 = \frac{1}{\tau} = \lambda$, and $\tilde{G}_0 = I$.

The reformulations to optimize λ simultaneously proposed in this section are motivated from Lanckriet et al. (2004b) and De Bie et al. (2003). As has been show in Lanckriet et al. (2004b), this joint optimization of λ works well in most cases in comparison with the simple approach of pre-specifying λ, but improved performance is not guaranteed.

5. Experimental Study

We conduct extensive experiments in this section to compare various aspects of relevant algorithms. The first part of the experiments focuses on combining kernel matrices derived from a single source of data. We demonstrate the effectiveness of the proposed MKL formulations for heterogeneous data integration in the second part of the experiments. The SDP formulations in Equations (8), (32), (41), and (45) are solved using the optimization package SeDuMi (Sturm, 1999). The QCQP formulations in Equations (13), (34), (43), and (46) are solved using the MOSEK package (Andersen and Andersen, 2000). The linear programs involved in the SILP formulations in Equations (16), (37), (44), and (47) are solved using the MATLAB[1] build-in function *linprog*. The tolerance parameter ε, defined in Equation (22), is set to 5×10^{-4}. The source codes of the proposed formulations for the experiments are available online.[2]

We first evaluate the proposed formulations for binary-class problems in Section 5.1. The experimental results and analysis for the multi-class formulations are presented in Section 5.2. We demonstrate the effectiveness of the proposed formulations for heterogeneous data integration in Section 5.3. In Section 5.4, we analyze the relationship between RKDA and SVM, and Section 5.5 studies the effect of regularization parameter on classification performance.

5.1 Experiments on Binary-class Problems

In the binary-class case, we compare our formulations with the 1-norm soft margin SVM, 2-norm soft margin SVM with and without the regularization parameter C optimized jointly as proposed in

1. The URL is `http://www.mathworks.com`.
2. The URL is `http://www.public.asu.edu/~jye02/Software/DKL/`.

sonar	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ/C	TSA
SDP_θ	0	0	0	0	0.550	0.307	0.022	0.041	0.029	0.050	5.0e-04	89.27±5.34
$SDP_{\theta,\lambda}$	0	0	0	0	0.550	0.307	0.022	0.041	0.029	0.050	3.1e-08	89.35±5.34
$QCQP_\theta$	0.003	0.003	0.004	0.011	0.444	0.375	0.046	0.034	0.035	0.048	5.0e-04	89.76±5.34
$QCQP_{\theta,\lambda}$	0	0	0	0	0.550	0.307	0.022	0.041	0.029	0.050	5.0e-02	89.35±5.34
$SILP_\theta$	0	0	0	0	0.459	0.406	0.011	0.034	0.032	0.059	5.0e-04	89.76±5.37
$SILP_{\theta,\lambda}$	0	0	0	0	0.547	0.313	0.023	0.031	0.034	0.052	4.2e-10	89.43±5.18
SDP_{Kim}	0.167	0.048	0.175	0.072	0.251	0.173	0.031	0.025	0.015	0.044	1.0e-08	88.46±5.28
SM1	0	0	0	0.040	3.953	5.514	0.491	0	0	0	1	89.75±4.90
SM2	0	0	0	0	2.875	6.765	0.359	0	0	0	1	89.59±5.24
$SM2_C$	0	0.011	0.014	0.084	4.253	6.038	0.570	0.004	0.001	0	5.5e+7	89.84±4.80
$RKDA_{K,\lambda}$[3]	0	0	0	0	0	3	14	11	2	0	–	89.67±6.62
$SVM_{K,C}$[4]	0	0	0	0	0	2	16	7	5	0	–	89.35±5.18
$RKDA_\lambda$[5]	53.65	54.95	60.24	73.57	84.95	90.56	89.91	86.99	85.52	84.95	–	–
SVM_C[6]	53.65	54.63	59.91	73.41	86.09	89.67	90.65	89.59	86.58	84.22	–	–

Table 2: Comparison of twelve methods on the *sonar* data set. The twelve methods, listed from top to bottom are: SDP formulation with λ fixed as proposed in Theorem 2.1, SDP formulation with λ optimized jointly as proposed in Equation (41), QCQP formulation with λ fixed as proposed in Theorem 2.2, QCQP formulation with λ optimized jointly as proposed in Equation (43), SILP formulation with λ fixed as proposed in Theorem 2.3, SILP formulation with λ optimized jointly as proposed in Equation (44), SDP formulation proposed in Kim et al. (2006), 1-norm soft margin SVM, 2-norm soft margin SVM without and with C optimized as proposed in Lanckriet et al. (2004b), RKDA and SVM with the kernels and regularization parameters selected by double cross-validation. Generally, subscripts of names in the first column are used to denote quantities that are optimized. The ten pre-specified kernels are all RBF kernels and the σ values used are 0.10, 0.22, 0.46, 1.00, 2.15, 4.46, 10.00, 21.54, 46.42, 100.00, as in Kim et al. (2006). The table is partitioned into three sections row-wise. In the first section, the columns headed with θ_i are the coefficients learned from the corresponding methods. The coefficients for the proposed six formulations are normalized to sum to one while those for other compared approaches are reported as obtained from their formulations. The column headed with λ/C provides the values of the regularization parameters, whether fixed or learned, and the test set accuracies and standard deviations are given in the last column. The second section includes RKDA and SVM with kernel and regularization parameter chosen by double cross-validation. We also report the number of times that a particular kernel is selected by cross-validation. The third section shows the accuracies of RKDA and SVM when the kernel is fixed and the regularization parameters chosen by cross-validation. Dashes are used to denote non-applicable items.

Lanckriet et al. (2004b), and the SDP formulation proposed in Kim et al. (2006). Also, we use double cross-validation to choose kernels and regularization parameters for SVM and RKDA. The 1-norm SVM classifier used is the LIBSVM package (Chang and Lin, 2001) and the 2-norm SVM code was obtained by adapting Anton Schwaighofer's implementation.[7]

Four data sets are used in the binary-class case. The *sonar, ionosphere*, and *cancer* data were retrieved from the UCI Machine Learning Repository (Newman et al., 1998). The *heart* data were

3. The number of times that a kernel is chosen by doubly cross-validated RKDA over 30 randomizations.
4. The number of times that a kernel is chosen by doubly cross-validated SVM over 30 randomizations.
5. Accuracy of RKDA when the kernel is fixed to each of the ten candidate kernels and λ is chosen by cross-validation.
6. Accuracy of SVM when the kernel is fixed to each of the ten candidate kernels and C is chosen by cross-validation.
7. The URL is `http://ida.first.fraunhofer.de/~anton/software.html`.

obtained from the STATLOG project.[8] All data are normalized. For each data set, we randomly partition the entire data set into training and test sets using the ratio 8:2. Ten RBF kernels are constructed from the training set data with different choices of the parameter σ as in Kim et al. (2006). Then the ten kernels are fed into the optimization software packages to obtain the corresponding coefficients for each kernel. Finally, the kernels are combined and used to compute the accuracy. For formulations SDP_θ, QCQP_θ, and SILP_θ, the λ value is fixed to 5.0×10^{-4}. For SDP_{Kim}, this value is fixed to 10^{-8}, as used in Kim et al. (2006). Following Lanckriet et al. (2004b), we fix C to 1 for SM1 and SM2.

Tables 2–5 present the experimental results on *sonar, heart, ionosphere*, and *cancer* data sets, respectively. In terms of performance, formulations that optimize λ jointly achieve similar accuracies to the ones with λ fixed. Note that for our experiments, all the data are normalized and the λ value is tuned manually for formulations with λ fixed. In practice, the optimal λ value is data-dependent. Thus, formulations that optimize λ jointly are expected to work better in such situations. In cases where no numerical problems have been reported, all the twelve compared methods achieve similar performance. However, for the first ten methods, there is no need for cross-validation, and they can be used for heterogeneous data integration from various sources.

For MKL formulations in Tables 2–5, we present the coefficients learned for each kernel. For doubly cross-validated methods, that is, $\text{RKDA}_{K,\lambda}$ and $\text{SVM}_{K,C}$, we record the number of times that a particular kernel has been selected in cross-validation. To understand the relative importance of each kernel when they are used individually, we fix the kernel to each of the ten pre-specified kernels and tune the regularization parameter using cross-validation and the accuracy of each kernel is recorded. We expect these quantities to have some relationship with the coefficients learned by solving convex programs. For the *sonar* data, RKDA_λ achieves the best performance on kernels corresponding to θ_6 and θ_7 while SVM_C achieves the highest accuracy on θ_6, θ_7 and θ_8. On the other hand, methods using linear combination of kernels favor kernels corresponding to θ_5 and θ_6. For the *heart* data, cross-validated SVM favors kernels corresponding to θ_9 and θ_{10} (they were chosen 9 and 17 times out of 30, respectively) while cross-validated RKDA uses kernels corresponding to θ_7, θ_8, and θ_9 most frequently. Our six formulations all give kernels corresponding to θ_1 and θ_{10} large weights, especially to θ_1, while SVM-based MKL formulations all set θ_{10} to zero. This may be due to the fact that RKDA and SVM optimize different criteria and thus favor different kernels. Another interesting observation is that all the ten MKL formulations give the first kernel a large weight while it is the worst kernel when used individually. This implies that the best individual kernel may not lead to a large weight when used in combination with others and poorly-performed individual kernel may contain complementary information that is useful when combined with other kernels. Such complementary information can not be incorporated when cross-validation is used to choose a single best kernel. For the *ionosphere* data, the best three individual kernels chosen by cross-validation are kernels corresponding to θ_5, θ_6 and θ_7. Interestingly, the kernel corresponding to θ_5 is assigned a zero weight by nine out of the ten MKL-based methods. For the *cancer* data, all kernels achieve similar performance when used separately while MKL-based formulations tend to assign a large weight to the kernel corresponds to θ_2.

To compare the efficiency of the proposed formulations with methods based on cross-validation, we record the computation time of the proposed QCQP and SILP formulations along with that of methods based on double cross-validation. Figure 1 plots the computation time of these six methods.

8. The URL is `http://www.liacc.up.pt/ML/old/statlog/datasets.html`.

Note that methods based on SDP have a much larger computation time than these six methods and their results are thus omitted. It can be seen that the proposed SILP formulations are more efficient than cross-validation based methods. Note that the convergence rate of the algorithm for solving the QCQP formulation depends on the data and parameter setting. Thus, it may take a relatively long time to converge in some cases, as shown by QCQP_θ on the *cancer* data in Figure 1.

heart	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ/C	TSA
SDP_θ	0.783	0.089	0	0	0	0.001	0	0	0.005	0.123	5.0e-4	81.98±4.27
$\text{SDP}_{\theta,\lambda}$	0.753	0.089	0	0	0	0.001	0	0	0.005	0.123	3.0e-2	81.67±4.49
QCQP_θ	0.734	0.117	0.003	0.001	0.001	0.001	0.002	0.004	0.008	0.129	5.0e-4	81.85±4.17
$\text{QCQP}_{\theta,\lambda}$	0.753	0.089	0	0	0	0.001	0	0	0.005	0.123	1.2e-1	81.67±4.47
SILP_θ	0.742	0.115	0	0	0	0.001	0	0	0.006	0.137	5.0e-4	81.98±4.27
$\text{SILP}_{\theta,\lambda}$	0.744	0.095	0	0	0	0	0	0	0.007	0.121	3.4e-2	81.73±4.23
SDP_{Kim}	0.881	0.036	0.002	0	0	0.001	0.003	0.004	0.009	0.065	1.0e-8	82.22±3.79
SM1	7.688	0.479	0.001	0.002	0.002	0.024	1.813	0	0	0	1	82.59±4.55
SM2	7.317	0.669	0	0	0	0.029	1.994	0	0	0	1	82.71±4.41
SM2_C	6.746	0.626	0	0	0	0.036	1.991	0	0	0	4.4e+5	82.53±4.58
$\text{RKDA}_{K,\lambda}$	0	0	0	0	1	2	7	9	7	4	–	77.35±5.83
$\text{SVM}_{K,C}$	0	0	0	0	0	0	2	2	9	17	–	81.73±4.48
RKDA_λ	58.64	65.06	69.62	73.33	77.28	79.13	78.70	77.65	76.79	75.92	–	–
SVM_C	57.96	64.75	71.79	76.60	79.93	80.30	81.66	81.54	82.22	82.59	–	–

Table 3: See the caption and footnotes of Table 2 for explanation.

ionosphere	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ/C	TSA
SDP_θ	0.362	0.073	0.033	0.108	0	0.147	0.277	0	0	0	5.0e-4	94.67±2.25
$\text{SDP}_{\theta,\lambda}$	0.362	0.073	0.033	0.108	0	0.147	0.277	0	0	0	1.4e-7	94.67±2.25
QCQP_θ	0.222	0.116	0.081	0.074	0.042	0.182	0.236	0.022	0.014	0.012	5.0e-4	94.86±2.39
$\text{QCQP}_{\theta,\lambda}$	0.362	0.073	0.033	0.108	0	0.147	0.277	0	0	0	2.2e-4	94.67±2.25
SILP_θ	0.261	0.080	0.061	0.116	0	0.167	0.316	0	0	0	5.0e-4	94.90±2.33
$\text{SILP}_{\theta,\lambda}$	0.364	0.073	0.028	0.112	0	0.145	0.279	0	0	0	3.6e-9	94.81±2.23
SDP_{Kim}	0.942	0	0	0	0	0.006	0.038	0.013	0.001	0	1.0e-8	89.43±3.98
SM1	3.553	0.672	0.482	0.240	0	4.828	0.221	0	0	0	1	95.28±2.09
SM2	2.883	0.682	0.683	0.196	0	5.305	0.248	0	0	0	1	94.81±2.07
SM2_C	3.910	0.714	0.561	0.255	0	5.300	0.256	0	0	0	1.4e+7	95.19±2.17
$\text{RKDA}_{K,\lambda}$	0	0	0	4	5	10	8	3	0	0	–	92.33±5.51
$\text{SVM}_{K,C}$	0	0	0	0	8	9	7	4	2	0	–	94.48±2.39
RKDA_λ	65.71	76.47	90.33	92.14	93.33	94.28	93.14	91.71	90.61	89.00	–	–
SVM_C	65.38	66.57	89.38	93.00	94.57	95.04	93.80	93.42	92.61	91.95	–	–

Table 4: See the caption and footnotes of Table 2 for explanation.

5.2 Experiments on Multi-class Problems

In the multi-class experiments, we compare our formulations with KRDA and SVM with kernels and regularization parameters tuned using double cross-validation. The methods proposed in Lanckriet et al. (2004b) and Kim et al. (2006) are only applicable to binary-class problems. Five data sets with different numbers of classes are used for this experiment. The *USPS* handwritten digits database was described in Hull (1994). We choose the first $3, 6$, and 8 classes with 100 data points in each class for the experiment. The *wine* data set was obtained from UCI Machine Learning Repository and the *satimage* and *segment* were obtained from the STATLOG project. We use the first $3, 5$, and 6 classes for the *satimage* data and the first 3 and 4 classes for the *segment* data. The *waveform*

cancer	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ/C	TSA
SDP_θ	0.013	0.006	0.014	0	0.018	0.044	0.061	0.101	0.280	0.463	5.0e-4	96.05±2.65
$\text{SDP}_{\theta,\lambda}$	0	0.532	0.096	0.040	0.008	0.020	0.244	0.020	0	0.048	1.0e-8	96.00±1.44
QCQP_θ	0.147	0.312	0.207	0.080	0.052	0.055	0.051	0.038	0.031	0.028	5.0e-4	97.01±1.31
$\text{QCQP}_{\theta,\lambda}$	0.003	0.662	0.111	0.042	0.010	0.015	0.134	0.007	0	0.004	4.3e-3	96.20±2.21
SILP_θ	0	0.468	0.298	0.022	0.010	0.020	0.170	0.007	0	0.005	5.0e-4	97.01±1.20
$\text{SILP}_{\theta,\lambda}$	0.003	0.663	0.105	0.047	0.009	0.014	0.132	0.009	0	0.005	1.3e-2	96.98±1.28
SDP_{Kim}	0.970	0.006	0.005	0.004	0.004	0.003	0.003	0.002	0.002	0.002	5.0e-4	73.43±4.28
SM1	1.797	5.706	0.179	0.008	0	2.308	0	0	0	0	1	97.08±1.27
SM2	1.483	5.541	0.402	0.023	0.006	2.527	0.013	0	0	0	1	97.15±1.22
SM2_C	1.690	4.855	0.546	0.047	0.003	2.521	0.015	0	0	0	1.0e+4	97.01±1.22
$\text{RKDA}_{K,\lambda}$	0	0	0	2	8	2	3	4	4	7	–	95.79±1.55
$\text{SVM}_{K,C}$	0	0	0	0	0	7	10	4	6	3	–	96.81±1.28
RKDA_λ	94.54	95.32	96.05	96.15	96.30	95.74	95.59	95.59	95.49	95.64	–	–
SVM_C	92.21	94.93	96.03	96.30	96.81	96.88	96.86	96.66	96.69	96.64	–	–

Table 5: See the caption and footnotes of Table 2 for explanation.

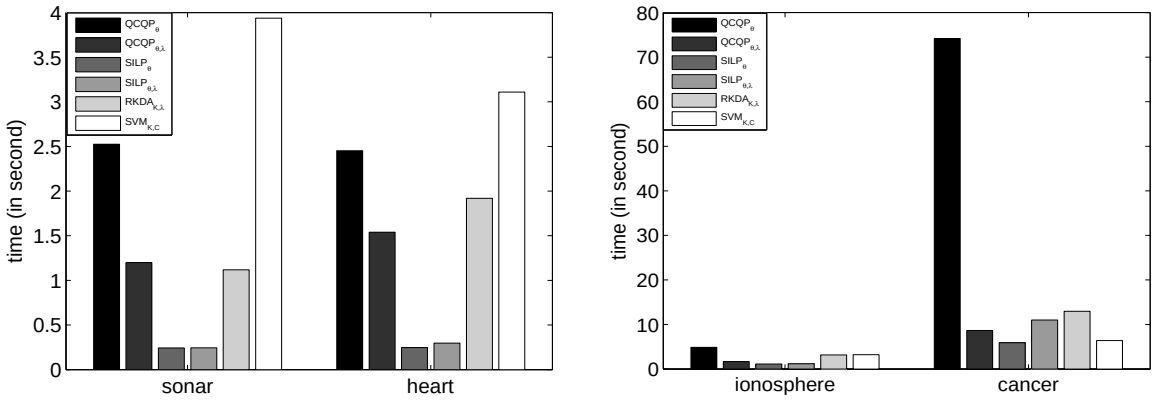


Figure 1: Computation time (in seconds) of the six methods on four binary-class data sets.

data set was described in Breiman et al. (1984) and are also available from UCI Machine Learning Repository. For each data set, we randomly partition the entire set into two subsets with 60% of the samples in the training set and 40% in the test set. Ten RBF kernels, with σ assigned the same values as in the binary-class case, are constructed from the training set.

Tables 6–15 present the experimental results on the ten data sets. In general, all the six proposed formulations achieve similar performance on the ten data sets. Compared to the QCQP and SILP formulations which are exact, our approximate SDP formulation for the multi-class problems work well in most cases. This implies that the approximate formulation is close to the exact one while the computational cost is lower. Furthermore, methods based on MKL and cross-validation achieve similar performance on all of the data sets.

In order to gain insights into the relative importance of each kernel when used in combination or separately, we use a similar experimental setup to the binary-class case. We found that for the *USPS*(3),[9] *USPS*(6), and *USPS*(8) data, all eight compared approaches favor the kernels corresponding to θ_9 and θ_{10}. Similar behavior has been observed for the *waveform*(3) data where only the last two kernels are selected by cross-validation and they are given large weights by all six MKL-based

9. The number in the parentheses denotes the number of classes used in the experiment.

USPS(3)[10]	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ	TSA
SDP_θ	0	0	0	0	0	0.027	0.023	0.012	0.518	0.420	5.0e-4	99.64±0.57
$SDP_{\theta,\lambda}$	0.007	0.004	0.015	0.016	0.013	0.036	0.022	0.014	0.493	0.379	6.3e-7	99.69±0.46
$QCQP_\theta$	0	0	0	0	0	0.037	0.052	0.040	0.372	0.498	5.0e-4	99.72±0.51
$QCQP_{\theta,\lambda}$	0.007	0.004	0.021	0.009	0.008	0.067	0.029	0.054	0.345	0.457	1.2e-5	99.64±0.47
$SILP_\theta$	0	0	0	0	0	0.037	0.052	0.043	0.370	0.499	5.0e-4	99.72±0.51
$SILP_{\theta,\lambda}$	0.007	0.005	0.019	0.011	0.006	0.069	0.027	0.057	0.343	0.457	3.6e-7	99.61±0.48
$RKDA_{K,\lambda}$[11]	0	0	0	0	0	0	0	0	8	22	–	98.97±1.11
$SVM_{K,C}$[12]	0	0	0	0	0	0	0	0	24	6	–	99.50±0.60

Table 6: Comparison of eight methods on the *USPS* data set when the first three classes are used. The eight methods, listed from top to bottom, are the SDP formulation with λ fixed as proposed in Theorem 3.1, the SDP formulation with λ optimized jointly as proposed in Equation (45), the QCQP formulation with λ fixed as proposed in Theorem 3.2, the QCQP formulation with λ optimized jointly as proposed in Equation (46), the SILP formulation with λ fixed as proposed in Theorem 3.3, the SILP formulation with λ optimized jointly as proposed in Equation (47), RKDA and SVM with kernels and regularization parameters chosen by double cross-validation. Generally, subscripts of names in the first column are used to denote quantities that are optimized. Ten RBF kernels are pre-specified and the values for σ are the same as those used in the binary-class case (see caption of Table 2). This table is partitioned into two sections row-wise. In the first section, the columns headed with θ_i present the coefficients learned from each method. Note that all coefficients are normalized to sum to one. This is followed by the values for the λ, whether fixed or learned. The test set accuracies are given in the last column. In the second section, we report the number of times that each kernel has been selected by double cross-validation and the accuracies. Dashes are used to denote non-applicable items.

USPS(6)	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ	TSA
SDP_θ	0	0	0	0	0	0.001	0.009	0.195	0.655	0.141	5.0e-4	98.40±0.80
$SDP_{\theta,\lambda}$	0.018	0.001	0.011	0.038	0.013	0.011	0.036	0.222	0.516	0.134	1.9e-6	98.33±0.88
$QCQP_\theta$	0	0	0	0	0	0.001	0.023	0.165	0.564	0.247	5.0e-4	98.36±0.82
$QCQP_{\theta,\lambda}$	0.020	0.002	0.003	0.035	0.025	0.008	0.063	0.165	0.463	0.216	2.8e-5	98.28±0.89
$SILP_\theta$	0	0	0	0	0	0.002	0.028	0.156	0.569	0.245	5.0e-4	98.35±0.84
$SILP_{\theta,\lambda}$	0.021	0	0.003	0.037	0.017	0.011	0.064	0.169	0.459	0.218	1.4e-8	98.29±0.88
$RKDA_{K,\lambda}$	0	0	0	0	0	0	0	0	20	10	–	98.08±0.85
$SVM_{K,C}$	0	0	0	0	0	0	0	0	26	4	–	98.11±1.02

Table 7: See the caption and footnotes of Table 6 for explanation.

approaches. Thus for these data sets, the kernels selected by cross-validation and multiple kernel learning (MKL) agree. In contrast, for the *satimage*(3), *satimage*(5), and *satimage*(6) data sets, the proposed MKL-based approaches assign large weights to the first five kernels. In particular, θ_2, θ_3, and θ_5 are given large values for the *satimage*(3) data; θ_2, θ_4, and θ_5 are given large values for the *satimage*(5) data; θ_1, θ_2, and θ_4 are given large values for the *satimage*(6) data. On the other hand,

10. The number in parenthesis denotes the number of classes used in the experiments.
11. RKDA with kernel and λ chosen by double cross-validation. The first ten columns show the number of times that a kernel is chosen by doubly cross-validated RKDA over 30 randomizations.
12. SVM with kernel and λ chosen by double cross-validation. The first ten columns show the number of times that a kernel is chosen by doubly cross-validated SVM over 30 randomizations.

USPS(8)	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ	TSA
SDP_θ	0	0	0	0	0	0.034	0.057	0.118	0.792	0	5.0e-4	97.60±0.83
$\text{SDP}_{\theta,\lambda}$	0.032	0.002	0.016	0.011	0.009	0.130	0.056	0.104	0.641	0	4.4e-6	97.64±0.70
QCQP_θ	0	0	0	0	0	0.001	0.116	0.053	0.697	0.133	5.0e-4	97.57±0.77
$\text{QCQP}_{\theta,\lambda}$	0.025	0.003	0.021	0.023	0.007	0.066	0.149	0.029	0.573	0.106	3.2e-5	97.65±0.72
SILP_θ	0	0	0	0	0	0	0.112	0.060	0.695	0.134	5.0e-4	97.51±0.77
$\text{SILP}_{\theta,\lambda}$	0.024	0	0.020	0.025	0.006	0.071	0.144	0.034	0.571	0.106	7.8e-9	97.64±0.74
$\text{RKDA}_{K,\lambda}$	0	0	0	0	0	0	0	0	12	18	–	97.53±0.78
$\text{SVM}_{K,C}$	0	0	0	0	0	0	0	0	18	12	–	97.10±0.82

Table 8: See the caption and footnotes of Table 6 for explanation.

wine(3)	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ	TSA
SDP_θ	0.044	0.147	0.104	0.415	0.104	0.012	0.008	0.016	0.018	0.133	5.0e-4	97.98±1.60
$\text{SDP}_{\theta,\lambda}$	0.065	0.177	0.086	0.394	0.100	0.011	0.008	0.018	0.015	0.126	2.4e-7	97.79±1.60
QCQP_θ	0.028	0.128	0.202	0.181	0.302	0.010	0.005	0.01	0.009	0.125	5.0e-4	98.12±1.49
$\text{QCQP}_{\theta,\lambda}$	0.046	0.169	0.171	0.177	0.289	0.009	0.004	0.011	0.004	0.123	9.3e-7	98.12±1.45
SILP_θ	0.023	0.133	0.205	0.178	0.305	0.010	0.003	0.009	0.010	0.125	5.0e-4	98.12±1.49
$\text{SILP}_{\theta,\lambda}$	0.045	0.162	0.182	0.172	0.287	0.010	0.008	0	0.009	0.124	2.2e-7	98.12±1.45
$\text{RKDA}_{K,\lambda}$	0	0	0	2	5	2	6	3	5	7	–	98.31±1.63
$\text{SVM}_{K,C}$	0	0	0	6	4	11	4	4	1	0	–	97.65±1.90

Table 9: See the caption and footnotes of Table 6 for explanation.

satimage(3)	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ	TSA
SDP_θ	0	0.157	0.247	0.046	0.353	0.08	0.024	0.046	0.017	0.032	5.0e-4	98.03±0.94
$\text{SDP}_{\theta,\lambda}$	0.003	0.251	0.193	0.033	0.349	0.06	0.02	0.05	0.012	0.029	9.0e-4	98.00±0.97
QCQP_θ	0	0.131	0.249	0.068	0.265	0.165	0.034	0.03	0.009	0.049	5.0e-4	98.06±0.96
$\text{QCQP}_{\theta,\lambda}$	0.002	0.229	0.193	0.049	0.281	0.115	0.055	0.023	0.003	0.048	1.8e-3	98.08±0.93
SILP_θ	0	0.133	0.246	0.073	0.252	0.181	0.026	0.033	0.006	0.051	5.0e-4	98.06±0.96
$\text{SILP}_{\theta,\lambda}$	0.003	0.226	0.197	0.047	0.274	0.133	0.042	0.019	0.004	0.053	1.6e-3	98.08±0.93
$\text{RKDA}_{K,\lambda}$	0	0	0	3	0	7	5	6	5	4	–	97.56±1.26
$\text{SVM}_{K,C}$	0	0	0	0	5	5	8	3	5	4	–	97.92±1.09

Table 10: See the caption and footnotes of Table 6 for explanation.

the two methods based on cross-validation tend to use the last five kernels more frequently than the first five kernels. This demonstrates that the best kernels used in combination and separately differ significantly for the *satimage* data set. We expect that complementary information exists among kernels for this data set such that a subset of kernels can be combined to obtain the optimal performance though none of them is the best kernel when used individually. Similar phenomenon can be observed from the *segment*(3) and *segment*(4) data sets in which the first kernel is assigned the largest weight by MKL-based formulations while it is never selected by cross-validation. This analysis shows that the information used by methods based on MKL and cross-validation may coincide or differ depending on the data.

To compare the efficiency of various methods, we report the computation time of the eight methods on the ten data sets in Table 16. It can be seen that the SDP formulations are much slower than methods based on cross-validation due to its inherent large complexity. The QCQP formulations are relatively efficient for data sets with a small number of classes. When the number of classes increases, their computation time increases rapidly. This is consistent with the theoretical

satimage(5)	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ	TSA
SDP_θ	0	0.302	0.244	0.196	0.225	0	0	0	0	0.034	5.0e-4	93.50±1.57
$SDP_{\theta,\lambda}$	0.002	0.477	0.083	0.225	0.184	0	0	0	0	0.030	2.1e-7	93.42±1.69
$QCQP_\theta$	0	0.165	0.469	0.001	0.251	0.068	0	0	0.004	0.043	5.0e-4	93.52±1.59
$QCQP_{\theta,\lambda}$	0.011	0.337	0.310	0.017	0.235	0.048	0	0.001	0.003	0.039	2.4e-6	93.28±1.51
$SILP_\theta$	0	0.162	0.470	0.005	0.247	0.071	0	0	0.004	0.042	5.0e-4	93.48±1.60
$SILP_{\theta,\lambda}$	0.014	0.331	0.316	0.010	0.242	0.044	0	0.001	0.003	0.039	5.9e-9	93.33±1.52
$RKDA_{K,\lambda}$	0	0	0	3	2	7	7	6	4	1	–	93.15±1.73
$SVM_{K,C}$	0	0	0	1	11	13	5	0	0	0	–	93.48±2.08

Table 11: See the caption and footnotes of Table 6 for explanation.

satimage(6)	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ	TSA
SDP_θ	0.131	0.478	0.031	0.249	0.072	0	0	0	0	0.039	5.0e-4	87.65±1.85
$SDP_{\theta,\lambda}$	0.293	0.338	0.04	0.212	0.06	0	0	0	0	0.033	2.3e-2	86.69±1.97
$QCQP_\theta$	0.102	0.454	0.096	0.138	0.128	0.043	0.001	0.002	0.009	0.029	5.0e-4	87.96±1.78
$QCQP_{\theta,\lambda}$	0.282	0.295	0.114	0.107	0.111	0.035	0.001	0.002	0.008	0.024	2.0e-2	87.22±1.84
$SILP_\theta$	0.108	0.448	0.094	0.143	0.123	0.046	0	0.002	0.006	0.031	5.0e-4	87.97±1.74
$SILP_{\theta,\lambda}$	0.277	0.299	0.112	0.106	0.114	0.032	0.003	0.005	0.006	0.024	2.2e-2	87.26±1.81
$RKDA_{K,\lambda}$	0	0	0	5	7	7	2	5	4	0	–	87.71±1.55
$SVM_{K,C}$	0	0	0	1	16	10	3	0	0	0	–	88.50±2.11

Table 12: See the caption and footnotes of Table 6 for explanation.

segment(3)	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ	TSA
SDP_θ	0.329	0.040	0.061	0.349	0.094	0.003	0	0	0	0.125	5.0e-4	99.17±0.62
$SDP_{\theta,\lambda}$	0.215	0.083	0.053	0.314	0.089	0.003	0	0	0	0.114	1.3e-1	99.00±0.83
$QCQP_\theta$	0.314	0.046	0.075	0.257	0.127	0.087	0.002	0	0	0.091	5.0e-4	99.19±0.67
$QCQP_{\theta,\lambda}$	0.215	0.075	0.079	0.218	0.127	0.079	0	0	0	0.084	1.2e-1	99.03±0.76
$SILP_\theta$	0.312	0.049	0.073	0.263	0.118	0.093	0.003	0	0	0.090	5.0e-4	99.17±0.69
$SILP_{\theta,\lambda}$	0.210	0.083	0.071	0.222	0.128	0.078	0	0	0	0.085	1.2e-1	99.03±0.76
$RKDA_{K,\lambda}$	0	0	2	8	1	3	4	6	4	2	–	98.86±1.08
$SVM_{K,C}$	0	0	5	9	8	3	4	1	0	0	–	99.06±0.81

Table 13: See the caption and footnotes of Table 6 for explanation.

analysis in Section 3.4. In contrast, the proposed SILP formulations are more efficient than methods based on cross-validation on all of the ten data sets.

5.3 Gene Expression Pattern Image Classification

In this experiment, we demonstrate the effectiveness of the proposed multiple kernel learning (MKL) formulations for data (feature) integration. Gene expression pattern images of *Drosophila melanogaster* embryo at a given developmental stage (time) capture the spatial and temporal distribution of gene expression patterns (Tomancak et al., 2002). The identification of genes showing spatial overlaps in their expression patterns is fundamentally important to formulating and testing gene interaction hypotheses (Kumar et al., 2002; Peng and Myers, 2004). Estimation of pattern overlap is most biologically meaningful when images from a similar time point (developmental stage) are compared. Thus, one of the central issues in gene expression pattern image analysis is the classification of images into different developmental stage ranges (Ye et al., 2006).

segment(4)	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ	TSA
SDP_θ	0.376	0.149	0.074	0.019	0.363	0.005	0	0	0	0.014	5.0e-4	97.00±1.09
$SDP_{\theta,\lambda}$	0.379	0.104	0.073	0.018	0.324	0.005	0	0	0	0.012	8.5e-2	96.77±1.25
$QCQP_\theta$	0.368	0.117	0.114	0.031	0.306	0.035	0	0	0	0.030	5.0e-4	97.00±1.17
$QCQP_{\theta,\lambda}$	0.373	0.073	0.111	0.028	0.271	0.033	0	0	0	0.027	8.5e-2	96.81±1.28
$SILP_\theta$	0.372	0.110	0.117	0.028	0.310	0.033	0	0	0	0.030	5.0e-4	97.02±1.12
$SILP_{\theta,\lambda}$	0.369	0.075	0.112	0.031	0.267	0.033	0	0	0	0.027	8.7e-2	96.81±1.26
$RKDA_{K,\lambda}$	0	0	1	1	2	3	2	7	6	8	–	97.31±0.93
$SVM_{K,C}$	0	0	0	4	5	7	4	6	1	3	–	96.83±1.38

Table 14: See the caption and footnotes of Table 6 for explanation.

waveform(3)	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	λ	TSA
SDP_θ	0.072	0.072	0.072	0.072	0.029	0	0.007	0.028	0.415	0.232	5.0e-4	83.03±2.68
$SDP_{\theta,\lambda}$	0.074	0.074	0.074	0.074	0.069	0	0.01	0.034	0.377	0.214	6.8e-7	83.22±2.61
$QCQP_\theta$	0.061	0.061	0.061	0.061	0.061	0.006	0.003	0.036	0.423	0.226	5.0e-4	83.03±2.68
$QCQP_{\theta,\lambda}$	0.071	0.071	0.071	0.071	0.071	0.006	0.006	0.041	0.385	0.209	6.2e-6	83.19±2.49
$SILP_\theta$	0.053	0.021	0.033	0.115	0.094	0	0	0.036	0.422	0.227	5.0e-4	83.08±2.74
$SILP_{\theta,\lambda}$	0.012	0.066	0.033	0.136	0.113	0	0.006	0.040	0.382	0.213	2.0e-7	83.22±2.50
$RKDA_{K,\lambda}$	0	0	0	0	0	0	0	0	6	24	–	84.17±3.14
$SVM_{K,C}$	0	0	0	0	0	0	0	0	12	18	–	81.86±2.99

Table 15: See the caption and footnotes of Table 6 for explanation.

Data	*USPS*			*wine*	*satimage*			*segment*		*waveform*
# of classes	3	6	8	3	3	5	6	3	4	3
SDP_θ	50.98	411.15	1021.16	18.91	95.95	415.58	753.07	74.38	163.26	64.09
$SDP_{\theta,\lambda}$	69.95	646.05	1642.17	27.02	130.51	710.70	1172.55	99.29	235.50	96.08
$QCQP_\theta$	5.19	81.24	276.27	1.15	4.23	36.56	79.61	4.16	14.89	4.09
$QCQP_{\theta,\lambda}$	5.96	88.05	286.49	1.29	4.67	39.09	82.93	4.50	16.20	4.65
$SILP_\theta$	0.32	1.52	3.03	0.30	0.59	1.97	3.65	0.62	1.03	0.24
$SILP_{\theta,\lambda}$	0.60	3.45	6.70	0.30	0.66	2.29	4.38	1.03	1.52	0.25
$RKDA_{K,\lambda}$	1.54	9.26	20.59	0.76	1.56	5.39	8.61	1.56	2.89	1.71
$SVM_{K,C}$	5.60	17.82	23.18	3.65	1.87	4.24	9.50	3.53	4.13	5.44

Table 16: Comparison of computation time (in seconds) of various methods. The reported time is averaged over 30 random partitions.

We collect 2705 gene expression pattern images in the first three stage ranges (1-3, 4-6, and 7-8) from the FlyExpress[13] database. The raw gene expression pattern images are of size 128×320. It has been observed (Gargesha et al., 2005) that across various developmental stages, a distinguishing feature is the image textural properties at sub-block level, because image texture at the sub-block level changes as embryonic development progresses. Gabor filters (Daugman, 1988) have been shown to be effective in detecting local texture features and are well suited for extracting textural features for gene expression pattern images.

We apply Log Gabor Filters to extract the texture features (Daugman, 1988). Gabor filters are the product of a complex sinusoidal function and a Gaussian-shaped function. We use Log Gabor filters with 4 different wavelet scales and 6 different filter orientations to extract the texture information. Hence, 24 Gabor images were obtained from the filtering operation. Note that all 24 Gabor images have the same size (i.e., 128×320) as the original one. Figure 2 plots the 24 Gabor im-

13. The URL is `http://www.flyexpress.net`.

ages extracted from a sample image. These images contain different but potentially complementary information for stage classification. Two RBF kernels are built from each of the 24 Gabor images with σ values assigned as 50 and 100, respectively. We thus obtain a total of 48 kernel matrices. To exploit the complementary information in kernels constructed from different Gabor images, we apply the proposed SILP formulation to learn a linear combination of the 48 kernel matrices.

The 2705 images are randomly partitioned into training and test sets using the ratio 1:9. Our experimental results show that SILP_θ achieves a classification accuracy of about 88.28%. To see how each of the 48 kernel matrices works when used individually, we fix the kernel matrix and tune the λ value using cross-validation. The maximum, minimum, and average accuracies achieved across the 48 kernel matrices are 72.03%, 54.37%, and 61.88%, respectively. We also assign a uniform weight of 1 to each of the 48 kernel matrices and the combined kernel matrix achieves an accuracy of about 72.65%. These results demonstrate that different Gabor images contain complementary information, which is critical for stage classification, and the proposed MKL formulations are effective in exploiting this information by combining different kernel matrices.

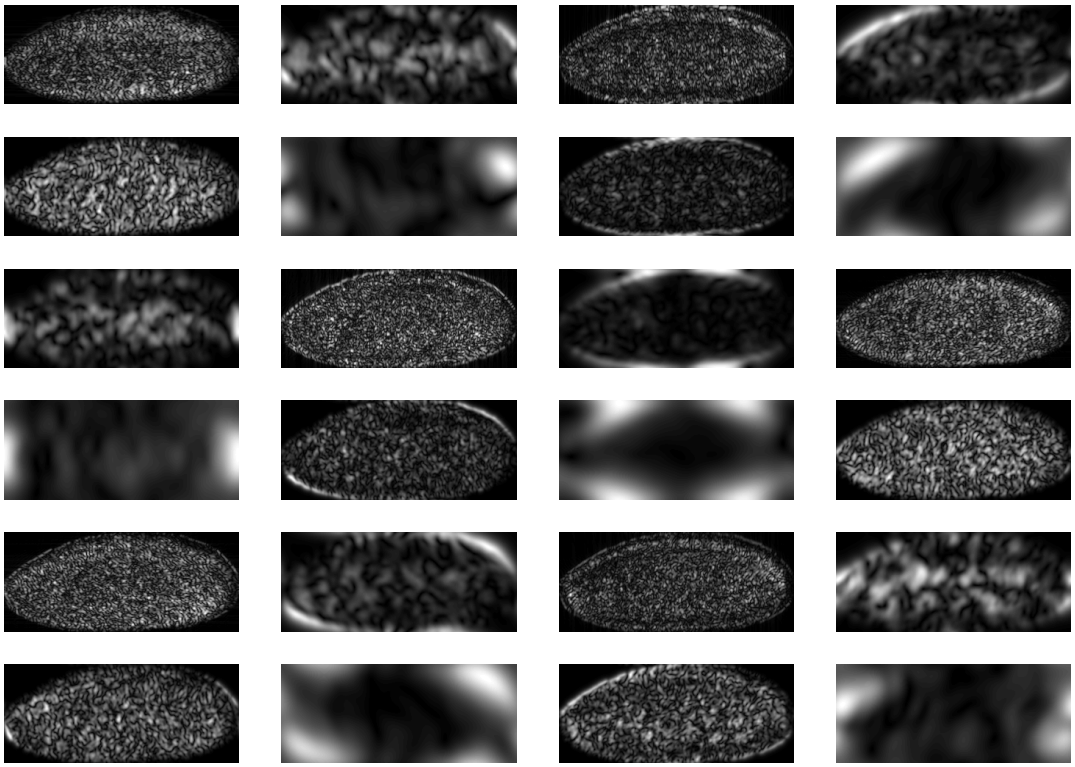


Figure 2: The 24 Gabor images extracted from a single sample image with 4 different wavelet scales and 6 different filter orientations.

		SM1			SM2			$SM2_C$		
data set	training size	C	SVs	PCT	C	SVs	PCT	C	SVs	PCT
sonar	167	1	155.4	93.05	1	152.2	91.12	2.43e7	156.8	93.89
heart	216	1	208.7	96.62	1	195.9	90.70	6.01e6	208.3	96.44
ionosphere	281	1	203.6	72.46	1	184.9	65.80	3.70e6	206.3	73.42
cancer	546	1	210.3	38.52	1	138.8	25.42	2.61e6	212.5	38.92

Table 17: The numbers of support vectors ("SVs") obtained from the 1-norm soft margin SVM, 2-norm soft margin SVM without and with C learned jointly that were proposed in Lanckriet et al. (2004b). These numbers are averaged over 30 random partitions. The total number of data points in the training set and the C values are also shown. The columns with title "PCT" show the percentage of support vectors over the training set.

5.4 SVM versus RKDA

It was shown (Shashua, 1999) that hard margin linear SVM is equivalent to linear discriminant analysis (LDA) when all the training points are support vectors. Through experiments, we found that the C values chosen by the 2-norm soft margin SVM proposed in Lanckriet et al. (2004b) are very large. Under such circumstances, soft margin SVM is approaching hard margin SVM. It has already been observed that SVM and kernel discriminant analysis usually have similar performance (Mika, 2002) and this has been confirmed by our experiments in the last two subsections. Thus it is interesting to report the number of support vectors for SVM. We record the average number of support vectors for 1-norm soft margin SVM, 2-norm soft margin SVM without and with C optimized jointly over the 30 random partitions reported in Section 5.1. As proposed in Lanckriet et al. (2004b), C is fixed to 1 for 1-norm and 2-norm soft margin SVM without C optimized. Table 17 reports the average C values obtained by the joint optimization 2-norm soft margin SVM and the average number of support vectors. For ease of comparison, we also report the size of training set and the averaged percentage of support vectors over 30 randomizations. It can be seen that for three out of four data sets, the percentages of support vectors are very high. This implies that SVM is similar to RKDA and explains why they have similar performance, as reported in the last two subsections.

5.5 The Effect of Regularization Parameter

In order to investigate the effect of regularization parameter in RKDA, we sampled 30 λ values between 10^{-10} and 10^2 uniformly over logarithmic scale and the accuracies of SDP_θ and QCQP_θ are plotted for two binary-class data sets (Figure 3) and two multi-class data sets (Figure 4). The results for SILP formulations are omitted since their performance is similar to their QCQP counterparts. It can be observed that as λ value changes, the accuracies oscillate in all cases. It can also be observed from the four figures that QCQP_θ tends to be less sensitive to the change of λ value than SDP_θ. This may be attributable to the fact that SDP is more computationally intensive and numerical problems may cause the poor performance. Indeed, we observed several reports of numerical problems from SeDuMi while conducting SDP experiments. The low accuracies of SDP_θ for some choices of λ in Figures 3 and 4 were caused by numerical problems and should be interpreted with caution.

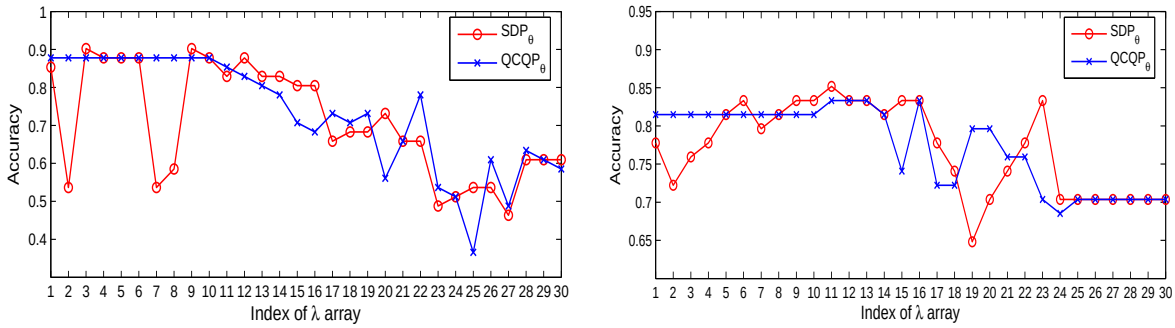


Figure 3: The change of accuracies for SDP_θ and QCQP_θ when λ varies from 10^{-10} to 10^2 for the *sonar* (left) and *heart* (right) data. The horizontal axis represents the indexes of the 30 λ values.

6. Discussion and Conclusion

We address the issue of learning appropriate kernels for RKDA in this paper. This problem is formulated as convex programs and thus globally optimal solutions are guaranteed. Practically, some convex optimization problems are computationally expensive and we propose approaches that are scalable and efficient to solve. While most existing work on kernel learning only deal with binary-class problems, we show that our binary-class formulations can be extended naturally to multi-class setting. Furthermore, we consider the problem of optimizing the kernel and regularization parameter in a joint framework, thus approaching the desirable goal of automated learning.

We have conducted extensive experiments to evaluate the proposed algorithms. When combining kernels from a single source of data, the proposed formulations have similar performance with approaches based on double cross-validation. When the candidate kernels contain complementary information, we show that the proposed formulations are effective to exploit such information. In terms of computation time, the SILP formulations are more efficient than approaches based on cross-validation. When evaluating the relative importance of each kernel (either used separately or in linear combination), we found that the best individual kernel sometimes coincides with the highly-weighted kernels in linear combination and sometimes disagrees considerably.

There are some directions for future work. Our experimental results have shown that the proposed approximate SDP formulation works well in most cases while it has a much lower computational cost in comparison with the exact formulation. We plan to compare the approximate formulation to the exact one in terms of complexity and performance. The derivation of multi-class formulations is based on an alternative criterion defined in Equation (23). This results in the same optimal transformation matrix as the original criterion in Equation (26) when a common (fixed) kernel matrix is used. However, they may differ when the kernel matrix is also optimized. We plan to investigate their differences further in the future. Most existing formulations for learning SVM kernels are restricted to the binary-class case. The idea from this paper may be useful for kernel learning in multi-class SVM. A more general problem is learning kernels for multi-label data in which each data point can be assigned to multiple classes. Such data are common in automatic image annotation problems (Lavrenko et al., 2004). We plan to explore these in the future.

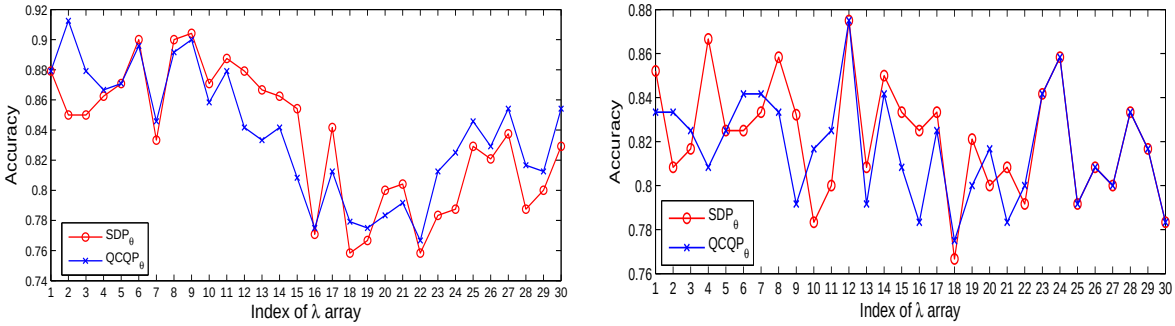


Figure 4: The change of accuracies for SDP_θ and QCQP_θ when λ varies from 10^{-10} to 10^2 for the *satimage*(6) (left) and *waveform*(3) (right) data. The horizontal axis represents the indexes of the 30 λ values.

Acknowledgments

The extensive experimental study in this paper was made possible with help from several researchers. Special thanks to Dr. Seung Jean Kim for helping in replicating their experiments. Dr. Johan Löfberg answered many questions about SeDuMi and his YALMIP through the online forum and E-mails. Dr. Anton Schwaighofer helped the authors to modify his SVM toolbox. The implementation of the proposed QCQP formulations is based on code for SVM kernel learning provided by Dr. Gert Lanckriet. This research is sponsored in part by funds from the Arizona State University and the National Science Foundation under Grant No. IIS-0612069.

Appendix A.

One of the basic tools used in our proof is the Sherman-Woodbury-Morrison formula (Golub and Van Loan, 1996): Let $S \in \mathbb{R}^{d\times d}$, and $Q, R \in \mathbb{R}^{d\times n}$. Assuming that both the matrices S and $(I + R^T S^{-1} Q)$ are nonsingular, we have

$$(S+QR^T)^{-1} = S^{-1} - S^{-1}Q(I+R^T S^{-1}Q)^{-1}R^T S^{-1}.$$

Since $P = PP$ and $P = P^T$, where P is the centering matrix defined in Equation (5), it follows that

$$\begin{aligned}
w^* &= (\Sigma_K + \lambda I)^{-1}(\mu_K^+ - \mu_K^-) \\
&= \left(\phi_K(X) P \phi_K(X)^T + \lambda I\right)^{-1} \phi_K(X) a \\
&= \left(\phi_K(X) PP \phi_K(X)^T + \lambda I\right)^{-1} \phi_K(X) a \\
&= \left((\phi_K(X)P)(\phi_K(X)P)^T + \lambda I\right)^{-1} \phi_K(X) a \\
&= \left(\frac{1}{\lambda} I - \frac{1}{\lambda^2}\phi_K(X) P \left(I + \frac{1}{\lambda} P\phi_K(X)^T \phi_K(X) P\right)^{-1} P \phi_K(X)^T\right) \phi_K(X) a \\
&= \frac{1}{\lambda}\phi_K(X)\left(I - P\left(\lambda I + PGP\right)^{-1} PG\right) a.
\end{aligned}$$

References

F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming*, 95: 3–51, 2003.

E. D. Andersen and K. D. Andersen. The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In T. Terlaky H. Frenk, K. Roos and S. Zhang, editors, *High Performance Optimization*, pages 197–232. Kluwer Academic Publishers, 2000.

A. Argyriou, R. Hauser, C. Micchelli, and M. Pontil. A DC-programming algorithm for kernel selection. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 41–48, 2006.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 41–48, 2004.

G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Compututation*, 12(10):2385–2404, 2000.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

L. Breiman, J. Friedman, C. J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

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`.

N. Cristianini and J.S. Taylor. *An Introduction to Support Vector Machines and other Kernel-based Learning Methods*. Cambridge University Press, 2000.

A. d'Aspremont, L. E. Ghaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448, 2007.

J.G. Daugman. Complete discrete 2-D Gabor transform by neural networks for image analysis and compression. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 36(7):1169–1179, 1988.

T. De Bie, G.R.G. Lanckriet, and N. Cristianini. Convex tuning of the soft margin parameter. Technical Report UCB/CSD-03-1289, EECS Department, University of California, Berkeley, 2003.

G. Fung, M. Dundar, J. Bi, and B. Rao. A fast iterative algorithm for Fisher discriminant using heterogeneous kernels. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

M. Gargesha, J. Yang, B. Van Emden, S. Panchanathan, and S. Kumar. Automatic annotation techniques for gene expression images of the fruit fly embryo. In S. Li, F. Pereira, H.-Y. Shum, and A. G. Tescher, editors, *Visual Communications and Image Processing 2005.*, pages 576–583, July 2005.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

R. Hettich and K. O. Kortanek. Semi-infinite programming: Theory, methods, and applications. *SIAM Review*, 35(3):380–429, 1993.

S. C. H. Hoi, R. Jin, and M. R. Lyu. Learning nonparametric kernel matrices from pairwise constraints. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, pages 361–368, 2007.

J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Analysis Machine Intelligence*, 16(5):550–554, 1994.

T. Jebara. Multi-task feature and kernel selection for SVMs. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

S.-J. Kim, A. Magnani, and S. Boyd. Optimal kernel selection in kernel Fisher discriminant analysis. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 465–472, 2006.

S. Kumar, K. Jayaraman, S. Panchanathan, R. Gurunathan, A. Marti-Subirana, and S. J. Newfeld. BEST: a novel computational approach for comparing gene expression patterns from early stages of *Drosophila melanogaster* develeopment. *Genetics*, 169:2037–2047, 2002.

G.R.G. Lanckriet, L. El Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2003.

G.R.G. Lanckriet, T. De Bie, N. Cristianini, M.I. Jordan, and W.S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004a.

G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004b.

V. Lavrenko, R. Manmatha, and J. Jeon. A model for learning the semantics of pictures. In *Advances in Neural Information Processing Systems 16*. 2004.

D. Lewis, T. Jebara, and W. S. Noble. Nonstationary kernel combination. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 553–560, 2006.

M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

C. A. Micchelli and M. Pontil. Feature space perspectives for learning the kernel. *Machine Learning*, 66(2-3):297–319, 2007.

C. A. Micchelli and M. Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

S. Mika. *Kernel Fisher Discriminants*. PhD thesis, University of Technology, Berlin, Oct. 2002.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.

S. Mika, G. Rätsch, and K.-R. Müller. A mathematical programming approach to the kernel Fisher algorithm. In *Advances in Neural Information Processing Systems 13*, pages 591–597, 2001.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A. Smola, and K. Müller. Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces. *IEEE Trans. Pattern Analysis Machine Intelligence*, 25(5):623–633, 2003.

Y. Nesterov and A. Nemirovskii. *Interior-point Polynomial Algorithms in Convex Programming*. SIAM Studies in Applied Mathematics. SIAM, 1994.

D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

C. S. Ong, A. J. Smola, and R. C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

H. Peng and E. W. Myers. Comparing *in situ* mRNA expression patterns of *Drosophila* embryos. In *Proceedings of the Eighth Annual International Conference on Research in Computational Molecular Biology*, pages 157–166, 2004.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods: Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, pages 775–782, 2007.

S. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines,Regularization, Optimization and Beyond*. MIT Press, 2002.

A. Shashua. On the relationship between the support vector machine for classification and sparsified Fisher's linear discriminant. *Neural Processing Letter*, 9(2):129–139, 1999.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, July 2006.

J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999.

P. Tomancak, A. Beaton, R. Weiszmann, E. Kwan, S. Shu, S. E Lewis, S. Richards, M. Ashburner, V. Hartenstein, S. E Celniker, and G. M Rubin. Systematic determination of patterns of gene expression during *Drosophila* embryogenesis. *Genome Biology*, 3(12), 2002.

I. W. Tsang and J. T. Kwok. Efficient hyperkernel learning using second-order cone programming. *IEEE Trans. on Neural Networks*, 17(1):48–58, 2006.

L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.

V.N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

J. Ye. Least squares linear discriminant analysis. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, pages 1087–1093, 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 and T. Xiong. Computational and theoretical analysis of null space and orthogonal linear discriminant analysis. *Journal of Machine Learning Research*, 7:1183–1204, 2006.

J. Ye, J. Chen, Q. Li, and S. Kumar. Classification of *Drosophila* embryonic developmental stage range based on gene expression pattern images. In *Proceedings of the Computational Systems Bioinformatics Conference*, pages 293–298, 2006.

A. Zien and C. S. Ong. Multiclass multiple kernel learning. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, pages 1191–1198, 2007.

Journal of Machine Learning Research 9 (2008) 759-813 Submitted 10/07; Revised 2/08; Published 4/08

Bayesian Inference and Optimal Design for the Sparse Linear Model

Matthias W. Seeger SEEGER@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, Tübingen, Germany

Editor: Martin Wainwright

Abstract

The linear model with sparsity-favouring prior on the coefficients has important applications in many different domains. In machine learning, most methods to date search for maximum a posteriori sparse solutions and neglect to represent posterior uncertainties. In this paper, we address problems of Bayesian optimal design (or experiment planning), for which accurate estimates of uncertainty are essential. To this end, we employ expectation propagation approximate inference for the linear model with Laplace prior, giving new insight into numerical stability properties and proposing a robust algorithm. We also show how to estimate model hyperparameters by empirical Bayesian maximisation of the marginal likelihood, and propose ideas in order to scale up the method to very large underdetermined problems.

We demonstrate the versatility of our framework on the application of gene regulatory network identification from micro-array expression data, where both the Laplace prior and the active experimental design approach are shown to result in significant improvements. We also address the problem of sparse coding of natural images, and show how our framework can be used for compressive sensing tasks.

Part of this work appeared in Seeger et al. (2007b). The gene network identification application appears in Steinke et al. (2007).

Keywords: sparse linear model, Laplace prior, expectation propagation, approximate inference, optimal design, Bayesian statistics, gene network recovery, image coding, compressive sensing

1. Introduction

In many settings favoured in current machine learning work, the model and data set are given in advance, and predictions with low error are sought. Many methods from different paradigms have successfully been applied to these problems. While Bayesian approaches, such as the one we describe here, enjoy some benefits in this regime, they can be more difficult to implement, less algorithmically robust, and often require more computation time than, for example, penalised estimation methods, whose computation often reduces to a standard optimisation problem. In our opinion, the real practical power of the Bayesian way is revealed better in higher-level tasks such as making optimally cost-efficient decisions or *experimental design*. In the latter, aspects of the model and measurement experiments are adapted based on growing knowledge about the current situation, and data is sampled in a sequential and actively controlled manner, with the aim of obtaining answers as quickly as possible. Our main motivation in the present work is to demonstrate how Bayesian experimental design can be implemented in a computationally efficient and robust way, and how a range of challenging applications can benefit from selectively sampling data where it is most needed.

A number of characteristics of the framework we propose here, are especially useful, if not essential, to drive efficient experimental design for the applications we consider. The latter, at least in the sequential variant discussed here, proceeds through a significant number of individual decisions (say, where to sample data next). In order to make each decision, our current uncertainty in variables of interest needs to be estimated quantitatively, and for a large number of candidates we have to consider how, and by how much, each of them would reduce this uncertainty estimate. As will become clear in the sequel, the uncertainty estimate is given by the posterior distribution, an approximation to which can be obtained robustly and efficiently by our method. The estimate is given as a Gaussian distribution, whose change after one more experiment can robustly and very efficiently be quantified. These points motivate our insistence on robustness[1] and efficiency below. Another key aspect of the models treated here is sparsity. This regularisation principle allows us to start from an overparameterised model, forcing parameters close to zero if they are not required. In our experiments, we demonstrate that the interplay between sparsity regularisation and experimental design seems to be particularly successful. In sequential design, most of the decisions have to be done early, without a lot of data available, and the focus (under a sparsity prior) on a few relevant effects only seems particularly useful in that respect.[2] In contrast, if the models of interest here are used with Gaussian priors, as is usually done, then sequential design is not different from optimising X beforehand. Although observations become available along the way, these are not used at all. We come back to this important point below.

In this work, we consider the *linear model*

$$u = Xa + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2 I), \tag{1}$$

where $X \in \mathbb{R}^{m,n}$ is the design matrix, and $a \in \mathbb{R}^n$ is the vector of unknown parameters (or weights). σ^2 is the variance of the Gaussian noise. The model can be thought of as representing a noisy linear system. It is called *underdetermined* if $m \leq n$, and *overdetermined* otherwise. In the underdetermined case, there are in general many solutions, even if we did not allow for noise, and additional desired qualities of a need to be formalised. In a Bayesian framework, this is done by placing a *prior distribution* on a, concentrating its mass on parameters fulfilling the requirements.

In the applications we consider, sparsity of a is a key prior assumption: elements of a should be set to very small values whenever they are not required to describe the data well. On the other hand, few elements should be allowed to be large if necessary. Among different solutions, the ones with the largest number of very small components should be preferred *a priori*. Enforcing sparsity is a fundamental statistical regularisation principle and lies behind many well known ideas such as *selective shrinkage* or *feature selection*. It is discussed in more detail in Section 2.1. Many sparsity-favouring priors have been suggested in statistics. In this paper, we concentrate on independent *Laplace* (or *double exponential*) distribution priors of the form

$$P(a) = \prod_i P(a_i), \quad P(a_i) = \frac{\tilde{\tau}}{2} e^{-\tilde{\tau}|a_i|}, \quad \tilde{\tau} = \tau/\sigma. \tag{2}$$

1. Robustness is an issue which is often overlooked when comparing machine learning methods, yet it is quite essential in experimental design, where many decisions have to be done based on small posterior changes, and where non-robust methods often lead to undesired, erratic high-variance behaviour. In experimental design, robustness can be more important than high posterior approximation accuracy.
2. We report empirical observations here at the moment. We are not aware of strong theoretical results about this aspect.

A key advantage of this choice over others is log-concavity, which implies important computational advantages (see Section 2.1, Section 3.5). We refer to the linear model with Laplace prior as *sparse linear model*.[3]

It is important to note that our method here is different from most of the classical treatments of experimental design for the linear model, which entirely focus on Gaussian prior distributions. The difference to these approaches lies in our use of non-Gaussian sparsity priors. Bayesian inference for the linear model with Gaussian prior is analytically tractable (see O'Hagan, 1994, Chapter 9), and most of the algorithmic complications we address in the following, do not arise there. On the other hand, comparative results in some of our experiments show very significant benefits of using experimental design with sparsity priors rather than Gaussian ones. Our findings point out the need to theoretically analyse and understand experimental design with non-Gaussian priors, although in the absence of analytically tractable formulae for inference, such studies would have to be done conditioned on particular inference approximations.

Once the linear model is endowed with sparsity priors which are not Gaussian, Bayesian inference in general is not analytically tractable anymore and has to be approximated. In this paper, we employ the expectation propagation (EP) algorithm (Minka, 2001b; Opper and Winther, 2000) for approximate Bayesian inference in the sparse linear model. Our motivation runs contrary to most machine learning applications of the sparse linear model considered so far (where maximally sparse solutions for a given fixed problem are estimated and good uncertainty representations seem unimportant), mainly because Bayesian experimental design is fundamentally driven by such uncertainty representations. While Bayesian inference can also be performed using Markov chain Monte Carlo (MCMC) (Park and Casella, 2005), our approach is much more efficient, especially in the context of sequential design, and can be applied to large-scale problems of interest in machine learning. Moreover, experimental design requires the robust estimation in posterior changes across many candidates, starting from a well-defined current distribution, which seems difficult to do with MCMC. The application of EP to the sparse linear model is numerically challenging, and some novel techniques are introduced here in order to obtain a robust algorithm. In this context, the role of log-concavity for numerical stability of EP is clarified. Moreover, a variant known as fractional EP (or Power EP) (Minka, 2004) is shown to essentially overcome stability problems in the context of underdetermined models, while standard EP seems inherently unworkable in these cases. This observation about fractional EP is novel to our knowledge.

We apply our method to the problem of identifying gene regulatory networks from data obtained through active experiments, disturbing the system in a controlled manner. Since such experiments are expensive and time-consuming, a sequentially designed approach is clearly beneficial. Indeed, our experiments on synthetic data, simulated using realistic setups, show clear advantages in using Bayesian experimental design and sparsity priors over traditional approaches.

We also address the problem of sparse linear coding of natural images, optimising the codebook by empirical Bayesian marginal likelihood maximisation. Since current hypotheses about the development of early visual neurons in the brain are equivalent to a Bayesian sparse linear model setup (Lewicki and Olshausen, 1999), our method is useful to test and further refine these.

There has been a lot of recent interest in signal processing in the problem of compressive sensing (Candès et al., 2006; Donoho, 2006; Ji and Carin, 2007). We show how our framework directly

3. The reader may be puzzled about the parameterisation in terms of $\tilde{\tau} = \tau/\sigma$. One reason for this is that it renders τ scale-free: it does not depend on the scale of the response u. A more important reason is given in Section 3.5.

addresses the key issues there, which are in fact optimal design problems, and we motivate applications.

The structure of this paper is as follows. In Section 2, the statistical notion of sparsity is explained and contrasted with notions currently dominant in machine learning. Furthermore, some key applications of the sparse linear model are described. In Section 3, we show how to do approximate inference using the expectation propagation method. Optimal design is discussed in Section 4, and an approximation to the marginal likelihood is given in Section 5. We show how to address large-scale problems in Section 6. Experimental results are presented in Section 7. Our framework is directly related to other approximate inference techniques in Section 8. The paper closes with a discussion in Section 9.

Efficient and extendible code for the sparse linear model will be put into the public domain, as part of the LHOTSE toolbox for adaptive statistical models.[4]

2. The Role of Sparsity. Applications

In this section, we clarify the statistical role of sparsity and motivate the Laplace prior (2) towards this end. We also introduce the applications of interest in our work here: identification of gene networks, and sparse coding of natural images, and we give remarks about applications to compressive sensing, which are subject to work in progress (Seeger and Nickisch, 2008). The importance of optimal design and hyperparameter estimation are motivated using these examples.

2.1 The Role of Sparsity Priors

In order to obtain flexible inference methods, it often makes sense in statistics to employ models with many more degrees of freedom than could uniquely be adapted given finite data. The resulting under-determinedness (sometimes referred to as "ill-posedness", "curse", or other equally negative terms) is broken by making additional assumptions, leading to the fact that some solutions are preferred over others, although both fit the data equally well. The mechanics of this comes in different variants, such as adding a penalty term (or regulariser) to a data-fit functional, or placing a prior distribution over hypotheses. The underlying principles are, however, the same.

A fundamental regularisation idea is *sparsity*. For example, suppose a prediction function is a linear combination of features. If knowledge of good (or optimal) features for a task is vague, it makes sense to allow for a large number of candidates, then let the data decide which are relevant. A sparsity prior (or regulariser) on the coefficients, for example in the sparse linear model (1) with Laplace prior (2), leads to just that. It is important to contrast this with the different, frequently used idea of forcing components to be uniformly small in size, so that the final predictor is a sum of many (or all) features, with each giving a small but non-zero contribution. An example of the latter is the linear model (1) with a *Gaussian* prior $P(a)$, which due to conjugacy allows for a simple analytical treatment (see O'Hagan, 1994, Chapter 9). Such a prior does not encode sparsity. The Laplace distribution puts much more weight close to zero than the Gaussian, while still having higher probabilities for large values. The implications are depicted in Figure 1, see also Tipping (2001).

A sparsity prior embodies the bi-separation characteristic: such parameters a with many very small components at the expense of few large ones are favoured over a whose components are

4. See www.kyb.tuebingen.mpg.de/bs/people/seeger/lhotse/.

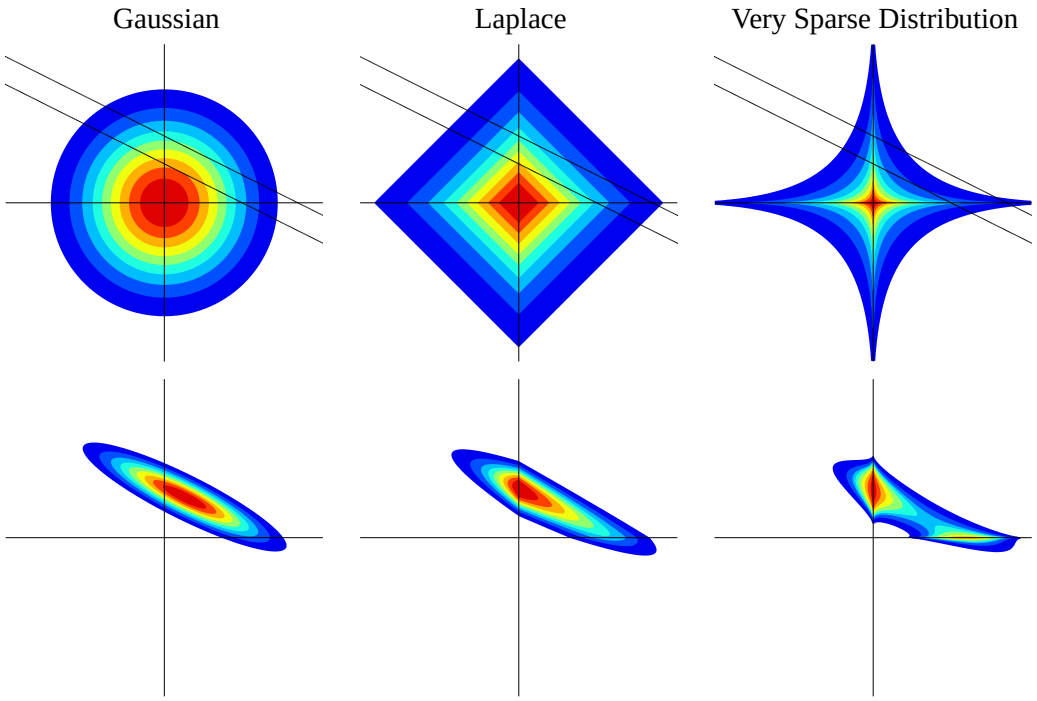


Figure 1: The entries of the parameter a can be given different prior distributions. Shown above are three candidates, plotted jointly over the values of two entries: a Gaussian, a Laplace, and a "very sparse" distribution ($P(a_i) \propto \exp(-\tau|a_i|^{0.4})$). We show contour plots of density functions, where areas of a specific color contain the same probability mass for each of the distributions. The upper row shows prior distributions of unit variance, together with the likelihood for a single measurement (a single linear constraint with Gaussian uncertainty). The lower row shows the corresponding posterior distributions. Whereas the Gaussian prior is spherically distributed, the other two shift probability mass towards the axes, so that more mass is given to sparse tuples (with one entry close to zero). This effect is clearly visible in the posterior distributions, being the normalised product of prior and likelihood. For the Gaussian prior, the areas close to the axes have rather low mass. In comparison, the posterior for the Laplace prior is skewed, so that more mass is concentrated close to the vertical axis. Both posteriors are log-concave and unimodal. The posterior for the "very sparse" prior shows shrinkage towards the axes even more strongly, and in terms of enforcing sparsity, this prior is preferable to the Laplacian. However, the posterior is bimodal now, suggesting two different interpretations for the single observation. The number of posterior modes can increase exponentially with the number of dimensions, so that sampling from or even representing this distribution has combinatorial complexity in general. *Figure by Florian Steinke*.

uniformly small throughout, but sizes are distributed regularly over this "range of smallness". Under the prior, most mass concentrates close to zero, but the tails are also comparatively heavy, allowing for occasional large values. In fact, heavy tails are an essential feature of a sparsity prior, since suppressing many components while still maintaining a flexible range of hypotheses is possible only if some components are allowed to take dominant values. The opposite is true for traditional Gaussian priors. Ishwaran and Rao (2005) call this bi-separation effect *selective shrinkage*, in that

parameters are shrunk towards zero selectively, while a Gaussian prior leads to a more uniform shrinkage. This characteristic is embodied even more strongly in sparsity priors other than the Laplace, such as "spike-and-slab" (mixture of narrow and very wide Gaussian), Student's t, or distributions $\propto \exp(-|\cdot|^\alpha)$, $\alpha < 1$, see also Figure 1. Among these, only the Laplace distribution is *log-concave*, leading to a posterior whose log density is a concave function, thus has a single local maximum. This simplifies and robustifies accurate inference computations significantly (see Section 3.5). For a non-log-concave prior, posteriors tend to be multi-modal, spreading their mass among many bumps, and accurate approximate inference can be a very hard problem. Furthermore, existing variational inference methods are more prone to non-robust unstable behaviour if applied to such models, and convergence or approximation errors can be hard to assess. Since we aim our method to be robust and easy to use by non-experts, we concentrate on log-concave Laplace sparsity priors in the sequel. The importance of log-concavity has been recognised in statistics and Markov chain sampling (Pratt, 1981; Gilks and Wild, 1992; Park and Casella, 2005; Lovász and Vempala, 2003; Paninski, 2005), but has not received much attention so far in work on variational approximate inference.

Our decision to prefer the Laplace sparsity prior over the conventional Gaussian choice, at the expense of having to approximate inference and of introducing significant complications, is ultimately validated by our experimental findings, where the Laplace prior yields large improvements over the Gaussian setting (see Section 7.1). However, apart from failing to encode a sparsity bi-separation, the Gaussian prior leads to other serious artifacts in the context of experimental design with the linear model. For example, suppose we are interested in sequentially designing covariates x for which responses u are queried (this is related to, but not the same setting we use here, see Section 4), say by choosing a "location" t in a feature map $x(t)$. It is well known and easily established that the Bayesian optimal design is *independent* of the response measurements we obtain along the way, it can in fact be computed beforehand. This fact seems absurd for many design problems, including ours here, pointing out a shortcoming of the model-prior combination. In the gene network identification problem (see Section 2.2 for notation), if we were to use a Gaussian prior, the posterior covariances would be identical for all rows of A. This means that no matter what disturbance experiments are done, the uncertainty in how gene i is influenced directly by the others, is the same for all i! Since design decisions mainly hinge on these uncertainty estimates, such artifacts due to a bad prior choice can lead to very suboptimal outcomes (see Section 7.1).

It is important to contrast our approach, and more generally the Bayesian statistical notion of sparsity, with what some *maximum a posteriori* (MAP) treatments of the sparse linear model are aiming to do. In the latter approach, which is very prominent in machine learning (Tibshirani, 1996; Chen et al., 1999; Peeters and Westra, 2004), the mode $\hat{a}$ of the posterior $P(a|X,u)$ is found through convex optimisation (recall that the log posterior is concave), and $\hat{a}$ is treated as posterior estimate of a. $\hat{a}$ has the property that many components are exactly zero:[5] the vector is sparse as such. This is useful for applications which aim for such exact sparsity, say for reasons of algorithmic efficiency. In contrast, in the Bayesian case, the posterior mass of all exactly sparse a (at least one component exactly zero) is zero, because the posterior has a density w.r.t. Lebesgue measure.[6] Not even commonly used Bayesian estimates of a, such as posterior mean or median, are exactly sparse in general. From a Bayesian viewpoint this makes sense, since in the presence of finite data,

5. One can easily show that as $\sigma^2 \to 0$, no more than m components of $\hat{a}$ can be non-zero.
6. Spike-and-slab sparsity priors have been used which place point masses on zero. However, approximate inference for such a setting is very challenging. Such priors are certainly not log-concave distributions.

one should always have some remaining uncertainty in exact values of parameters. The role of a sparsity prior in our situation is not to force many parameter values exactly to zero, but rather to enforce a clear partition into a large set of parameters which are close to zero with high (posterior) probability, and a small set which have significant mass on large values. Interestingly, following this probabilistic notion of sparsity sometimes allows to uncover sparsity in parameters of higher order that are of real interest, which is missed by MAP approaches. Our findings in Section 7.2 are a nice example of this effect.

2.2 Gene Network Identification

Measuring m-RNA expression levels for many genes in parallel is affordable and widely done today using DNA micro-arrays (DeRisi et al., 1997). One goal of such efforts is to recover regulatory networks. For example, some genes may code for transcription factor proteins, which up-/down-regulate the expression of other genes. In an *active* approach to network recovery, the evolution of expression levels of n genes is modeled by a system of ordinary differential equations, which is linearised at its steady state:

$$\dot{x}(t) = Ax(t) - u(t) + \varepsilon(t), \tag{3}$$

where $x(t)$ is the deviation in expression from steady state, and $\varepsilon(t)$ is white noise. A is the system matrix, whose non-zero entries represent the edges of the network. $u(t)$ is an external control, allowing the active user to probe the unknown A. It is generally assumed that $u(t)$ is small enough not to drive the system out of its linearity region. Due to the noisy environment, it is typical to restrict controls to be constant, $u(t) \equiv u$, and to measure the new steady state $\lim_{t\to\infty} x(t)$ (Tegnér et al., 2003). Such disturbances may be implemented biologically using gene switches (Gardner et al., 2000), which puts further restrictions on allowable u.

The linear model of (1) captures this setup as follows. Suppose that m observations $D = \{x_i, u_i\}$ have been made, where u_i is an external control, and x_i is the corresponding difference between steady state expression levels of the perturbed and the unperturbed system. We write $U = (u_i)^T \in \mathbb{R}^{m,n}$, $X = (x_i)^T \in \mathbb{R}^{m,n}$. We have that $u_i \sim N(Ax_i, \sigma^2 I)$. If a_i is the transpose of the i-th row of A, this Gaussian likelihood decomposes into n factors, one for each a_i. If the coefficients of A are assumed to be independent Laplacian *a priori*, the posterior factorises accordingly:

$$P(A|D) = \prod_j P(a_j|D), \quad P(a_j|D) \propto N(U_{\cdot,j}|Xa_j, \sigma^2 I) \prod_i P(a_{j,i}).$$

Thus, we have n independent sparse linear models, on which inference is done separately.

Since biological experiments involving gene switches are expensive and time-consuming, a key requirement is to perform with as few data as possible, which is possible if biological prior knowledge is encoded in $P(A)$. Importantly, regulatory networks are observed to be sparsely connected, that is, plausible A are sparse, a property which is directly represented in the sparse linear model. A principled way of saving on the number of expensive experiments is *optimal design*, which in a special case of interest here boils down to the question: given the current posterior belief and a set of candidate controls u_*, which of these experiments renders most new information about A? Thus, a "value of information" is sought which can be computed for each candidate u_* *without* doing the corresponding experiment. Optimal design is well developed in classical and Bayesian statistics (Fedorov, 1972; Chaloner and Verdinelli, 1995; MacKay, 1991), and access to this methodology is a key motivation for developing a good inference approximation here.

2.3 Coding of Natural Images

A second application of the sparse linear model is concerned with linear coding of natural images (Olshausen and Field, 1997; Lewicki and Olshausen, 1999), with the aim of understanding properties of visual neurons in the brain. Before we describe the setup, it is important to point out what our motivation is here, since it deviates significantly from what is usually done in machine learning. One approach in theoretical neuroscience is to formulate principles which can be described reasonably simply in mathematical terms, so that certain phenomena observed in experiments emerge if *only these principles are followed*. Once such principles are established, one can think about neural mechanisms implementing them. Also, if different principles lead to the same observed phenomena, one can plan experiments to further discriminate between them. In machine learning, the problems are known, and methods are compared with the aim of finding the best one, using an evaluation score and methodology independent of the set of methods to ensure a fair comparison. If results are not much different across methods, the most efficient one is usually preferred. In theoretical neuroscience,[7] the outcomes are known, and simple "universal" principles to explain them are sought. Once a principle is suggested, the aim is to devise a method following that principle as closely as possible. If such a method can then successfully reproduce observed phenomena, the principle can be established. In the context here, we are interested in testing a hypothesis put forward by Lewicki and Olshausen (1999), which is formulated in Bayesian terms. We are not interested here in coding images in the best possible way, and certainly not in how to do this with the highest computational efficiency.

An image $u \in \mathbb{R}^m$ is modeled as $u = Xa + \varepsilon$, where the columns of X are codebook vectors, $a \in \mathbb{R}^n$ are basis coefficients, and $\varepsilon \sim N(0, \sigma^2 I)$ independently. Note that codebook vectors are also referred to as filters, or basis functions. A central assumption on a is sparsity, which is especially important in the underdetermined (or overcomplete) regime: $m < n$. The Bayesian approach via the sparse linear model (1) has been suggested by Lewicki and Olshausen (1999), where the average coding cost of images under the model is put forward as criterion for ranking different code matrices X. Their work aims to give a probabilistic interpretation to the findings of Olshausen and Field (1997). In a Bayesian nomenclature, the average coding cost is the negative log marginal likelihood $-\log P(D)$, where $P(D) = \prod_j P(u_j)$, $P(u_j) = \int P(u_j|a_j)P(a_j)\,da_j$, and differences of these for different X are log Bayes factors. In Section 5, we show how to obtain a good approximation to $-\log P(D)$ through EP, which can be minimised w.r.t. the code matrix X in a gradient-based way. This general idea is proposed by Lewicki and Olshausen (1999) as well, but they use a second-order (Laplace) approximation to $-\log P(D)$, which is not suitable in case of a Laplace prior.[8] In the earlier approach of Olshausen and Field (1997), the learning of X is driven by point estimates (or maximum a posteriori decoding), and a criterion different from the average coding cost is optimised. This ignores posterior uncertainty in the decodings, and requires additional renormalisation heuristics in order to learn a good code. Our approximation here implements the probabilistic hypothesis of Lewicki and Olshausen (1999) fairly accurately, and can therefore be used to analyse more closely which of the features found by Olshausen and Field (1997) are due to the minimisation of average coding cost, versus which may rather be caused by particular characteristics of their learning method. Note that maximisation of the marginal likelihood is an important empirical

7. Or, in fact, in most natural sciences, with the exception of Engineering and Computer Science.
8. The problem is that $\log P(a_j)$ is not differentiable at the posterior mode $\hat{a}_j$, so that the matrix B in Lewicki and Olshausen (1999) is not well-defined. See comments in Section 3.

Bayesian way of estimating free hyperparameters, and Bayes factors are routinely used to compare model setups, so our approximation will be useful in other applications of the sparse linear model as well.

One of the key questions in natural image modelling is: under which conditions do basis vectors emerge which are spatially oriented and localised, thus show properties which have been established for the receptive fields of certain visual neurons? Given that the hypothesis of Lewicki and Olshausen (1999) is taken for granted, the sparsity hypothesis can be tested using the sparse linear model. Interestingly, other conditions brought forward (such as non-negativity) can also be dealt with in principle using the linear model, with different priors on a. Technically, non-negativity can be implemented by "cutting off" (and renormalising) a given prior density, which amounts to replacing $P(a_i)$ by $2P(a_i)\mathrm{I}_{\{a_i \geq 0\}}$. Importantly, if $P(a_i)$ is log-concave, so is this modification, because $\log \mathrm{I}_{\{a_i \geq 0\}}$ is (generalised) concave. For example, "cutting off" the Laplace distribution results in the exponential distribution,[9] which has been used in the context of image modelling by Hojen-Sorensen et al. (2002). While exponential priors encode non-negativity and sparsity at the same time, a cut-off Gaussian $P(a_i) = 2N(a_i|0, \tilde{\tau}^{-2})\mathrm{I}_{\{a_i \geq 0\}}$ could be used to represent non-negativity alone.

2.4 Bayesian Compressive Sensing

There has been a lot of recent interest in signal processing in the problem of compressive sensing (Candès et al., 2006; Donoho, 2006). The idea is appealingly simple. Suppose a signal is measured and then transferred over some channel or stored on some media. The second step almost always includes lossy compression in practice, especially with signals such as images or sound, where the loss may not be perceivable. Many of today's codes are *sparse*: the signal is transformed one-to-one, after which many coefficients are close to zero. These coefficients are then set to zero, and are not transmitted or stored. The first sensing (or sampling) step is traditionally done in a way which does not lead to loss of information, say by relying on the Nyquist/Shannon sampling theorem. The question of compressive sensing is whether one can sample a signal in a more efficient, but lossy way, so that the loss is part of that one encountered through subsequent compression anyway. The main attractiveness is that if a lossy compression is used, compressive sensing does not add further losses.

Although maybe not phrased in that way by much of the existing work, this is a classical problem of experimental design. An approximate Bayesian variant of compressive sensing has been proposed by Ji and Carin (2007), using sparse Bayesian learning (Tipping, 2001) to approximate the inference. Most practical codes today are linear, in that $y = \Phi a$, where y is the signal (say, an image), Φ is the code matrix (say, a Wavelet transform), and a are the coding coefficients. The code is designed such that a is approximately sparse, in much the same sense as elaborated in Section 2.1. Typically, Φ is one-to-one, even unitary. We then measure the signal linearly, that is, obtain $u = Py + \varepsilon$, where P is a measurement matrix, u are the responses, and ε is noise due to measurement errors. Here, $P \in \mathbb{R}^{m,n}$ with $m < n$ (the savings promised by compressive sensing). If $X = P\Phi$, this is exactly the setup of the linear model (1). Furthermore, the sparsity of a is encoded via a Laplace prior, motivating the sparse linear model for compressive sensing.

The measurement matrix P can be designed at will, where we are possibly limited to certain parametric families, due to constraints from the measurement architecture or (for very large n)

9. For this reason, the Laplace distribution is sometimes called double exponential distribution.

computational tractability (see Section 6). Anyway, we can design P row by row through an instance of standard sequential experimental design described in Section 4. This has been proposed in Ji and Carin (2007). Moreover, we can try to optimise P *a priori* over a large database of signals from the domain of the application, in what turns out to be an interesting variant of the image coding problem of Section 2.3. Here, the image code Φ is fixed, but P is to be learned.

Another point in which our approach differs from much of the existing work on compressive sensing, has to do with the sparsity prior we employ. Namely, many theoretical results have been obtained under the assumption that the signal y can be *exactly* sparsely coded, in that most coefficients in the corresponding a are exactly zero. However, in many real-world applications, this may be too strict an assumption. For example, the Wavelet transform of an image is virtually never exactly sparse, but rather features the bi-separation characteristic discussed in Section 2.1: many coefficients are very close to zero, and a subsequent quantisation leads to an image visually indistinguishable from y. Our sparsity prior concentrates on the bi-separation characteristic, without enforcing exact sparseness, thus may be better suited to many compressive sensing applications than the requirement of exact sparsity.

Results from experiments with different variants of compressive sensing are in preparation (joint work with Hannes Nickisch) and will be presented in a later paper (Seeger and Nickisch, 2008).

3. Expectation Propagation for the Linear Model

Exact Bayesian inference is not analytically tractable for the sparse linear model. In this section, we show how to apply the recently proposed *expectation propagation* (EP) method (Minka, 2001b; Opper and Winther, 2000) to this problem, circumventing some caveats we have not seen being addressed before. We begin with a high-level description, filling in the details further below. In the case of EP for the sparse linear model, it turns out that some details concerning robustness are essential for obtaining a practically useful method.

In EP, we compute a Gaussian approximation $Q(a)$ to the posterior

$$P(a|D) \propto N(u|Xa, \sigma^2 I) P(a).$$

Here, the likelihood $N(u|Xa, \sigma^2 I)$ is Gaussian, and it is the non-Gaussian prior $P(a)$ which forces us to approximate Bayesian inference. Our restriction to Gaussian $Q(a)$ is primarily done for pragmatic reasons, since Bayesian computations such as marginalisation and conditioning can be done analytically in this family, using standard matrix operations which can be computed robustly and efficiently. However, in our case, the Gaussian approximation can be argued for more strongly than in many others. Namely, recall that $\log P(a)$ is concave (2). Since the likelihood is a Gaussian function of a, the true log posterior $\log P(a|D)$ is concave as well, thus has a single mode only.

If $P^{(0)}(a) := N(u|Xa, \sigma^2 I)$ is the Gaussian likelihood (1), the true posterior is

$$P(a|D) \propto P^{(0)}(a) \prod_i t_i(a_i), \quad t_i(a_i) = \frac{\tilde{\tau}}{2} e^{-\tilde{\tau}|a_i|}.$$

We refer to the t_i as *sites*, and to $P^{(0)}$ as *base measure*. Note that the latter is not in general normalisable.

In order to motivate EP, note that an optimal Gaussian posterior approximation $Q(a)$ (at least in our context here) would be obtained by setting its mean and covariance to the true posterior

statistics. However, this would require a n-dimensional non-Gaussian integration, which cannot at present be done tractably. However, we are able to compute *one-dimensional* integrals involving a single non-Gaussian site $t_i(a_i)$. EP makes use of this capability in an iterative fashion, in order to approximate the desired joint posterior moments. The EP posterior approximation has the form

$$Q(a) \propto P^{(0)}(a) \prod_i \tilde{t}_i(a_i),$$

where $\tilde{t}_i(a_i|b_i,\pi_i)$ are Gaussian factors. Formally, one gets from the intractable $P(a|D)$ to its Gaussian approximation $Q(a)$ by replacing each non-Gaussian $t_i(a_i)$ by a Gaussian counterpart $\tilde{t}_i(a_i)$. This formal replacement introduces *site parameters* $b, \pi \in \mathbb{R}^n$, and the EP algorithm is an iterative method for adjusting these in turn.

In a single EP update, b_i, π_i are adjusted, while leaving all other site parameters the same. Starting from the current Gaussian approximation Q, we compute the Gaussian *cavity distribution* $Q^{\backslash i} \propto Q\tilde{t}_i^{-1}$ by dividing out the *site approximation* $\tilde{t}_i(a_i)$, then the non-Gaussian *tilted distribution* $\hat{P}_i \propto Q^{\backslash i} t_i$ by multiplying in the true site $t_i(a_i)$ instead, finally we update b_i, π_i such that the new Q' has the same mean and covariance as $\hat{P}_i$. These single updates are iterated in some random ordering over the sites until convergence.[10] Thus, EP is inherently based on the idea of *moment matching*. In other words, Q' is chosen by minimising the relative entropy $\mathrm{D}[\hat{P}_i \,\|\, \cdot]$ over all Gaussians.

From an algorithmic viewpoint, several questions have to be addressed. First, how can we *represent* the Gaussian $Q(a)$, so that single EP updates are served well in terms of efficiency and robustness? We will see that a good representation has to allow for the rapid "random-access" extraction of marginals $Q(a_i)$, and we have to be able to efficiently and robustly update it after a change of b_i, π_i. Second, how can the mean and variance of the non-Gaussian $\hat{P}_i(a_i)$ be computed accurately? To address these questions, we need to introduce some notation and details.

Denote the family of unnormalised Gaussian measures by

$$N^U(z|b,P) := \exp\left(-\frac{1}{2} z^T P z + b^T z\right),$$

P being positive semidefinite. Then, $P^{(0)}(a) = N^U(a|\sigma^{-2}b^{(0)}, \sigma^{-2}\Pi^{(0)})$ with $\Pi^{(0)} = X^T X$, $b^{(0)} = X^T u$. The site approximations are $\tilde{t}_i(a_i) = N^U(a_i|\sigma^{-2}b_i, \sigma^{-2}\pi_i)$, so that Q is a Gaussian. In general applications of EP, the π_i can become negative, but this does not happen in the cases discussed in this paper. We will show in Section 3.5 that for *log-concave sites* t_i, all π_i remain nonnegative throughout the course of the EP algorithm.

Moreover, the reader may wonder why we restrict ourselves to $\tilde{t}_i(a_i)$, instead of allowing for general site approximations $\tilde{t}_i(a)$. Also, a careful reader may have noted that we are only concerned about marginal distributions $Q(a_i)$ and $\hat{P}_i(a_i)$ during an EP update at t_i. Importantly, all this does not come with a loss of generality, as is shown in Section 3.1.

We initialise the algorithm with $b = 0$ and $\pi = \varepsilon 1$, $\varepsilon > 0$. A useful heuristic is $\varepsilon = \tau^2/2$, making sure that $t_i(a_i)$ and $\tilde{t}_i(a_i)$ have the same variance initially. In the case of the sparse linear model, the implementation of EP is complicated in a fundamental way. If $m < n$ (underdetermined case), the base measure $P^{(0)}(a)$ is not normalisable, because $\Pi^{(0)} = X^T X$ is singular. It is easily seen that

10. To our knowledge, little is known in general about convergence properties of EP, even with log-concave sites. Empirically, we have never observed failure of convergence in the log-concave case, except for reasons of numerical instability (see Section 3.3.1). Obtaining a formal convergence proof in this case remains a very important point for future research.

if any of the $\pi_i = 0$, the resulting $Q(a)$ is (in general) not a proper Gaussian either, so we have to ensure that $\pi_i > 0$ at all times. If $m \ll n$, we would like to represent the posterior Q in a way which scales with m rather than n. We address these issues below in this section.

It is important to note that EP is not merely a local approximation, in that $\tilde{t}_i$ is somehow fitted to t_i locally. This would not be useful at all,[11] because posterior mean and covariance are shaped *jointly* by the non-Gaussian t_i and the coupled Gaussian base measure. Loosely speaking, the likelihood couples coefficients a_i, so that the intentions of the prior factors $t_i(a_i)$, namely to force their respective arguments towards zero, have to be weighted against each other in a very non-local procedure.[12] After each EP update, although only a single site approximation is modified, its influence propagates to all other sites, because they are coupled through the base measure. In fact, non-locality is a central aspect of Bayesian inference which makes it so hard to compute, and inference is particularly hard to do in models where strong long-range posterior dependencies are present which cannot easily be predicted from local interactions only.

Finally, would it not be much simpler and more efficient to locate the true posterior mode through convex optimisation (recall that the posterior is log-concave), then do a Laplace approximation there, which amounts to expanding the log posterior density to second order around the mode? Indeed, finding the mode can be done efficiently by solving a quadratic program (Tibshirani, 1996). General problems with this approach include that the curvature around the mode may not be characteristic of the target density, and that the mode may not be a good place to center a Gaussian approximation at. In the case of the sparse linear model, the Laplace approximation is not even a valid option, since it is not well-defined in the presence of a Laplace prior.[13] Namely, $\log P(a_i)$ does not have a curvature at $a_i = 0$. The posterior mode is guaranteed to contain at least some zero components, so the curvature there is not defined. EP does not require $P(a_i)$ or $\log P(a_i)$ to be differentiable. On models where both methods can be applied, EP tends to improve upon a Laplace approximation significantly, but is also typically more expensive (Minka, 2001a; Kuss and Rasmussen, 2005).

3.1 Overview of Algorithm

In this section, we provide a schematic overview of the EP algorithm, filling in details in the sections to come. Recall that EP iterates site updates at $i \in \{1,\ldots,n\}$, computing $Q^{\backslash i} \propto Q\tilde{t}_i^{-1}$ and $\hat{P}_i \propto Q^{\backslash i} t_i$, then adjusting $Q \to Q'$ such that Q' has the same mean and covariance as $\hat{P}_i$. Since t_i depends on a_i only, $\hat{P}_i(a_{\backslash i}|a_i) = Q^{\backslash i}(a_{\backslash i}|a_i)$, where $a_{\backslash i} := (a_j)_{j\neq i}$, thus $Q'(a_{\backslash i}|a_i) = Q^{\backslash i}(a_{\backslash i}|a_i)$. Therefore, an EP update automatically results in the site approximation $\tilde{t}_i$ being a (Gaussian) function of a_i only. It also implies that in order to drive the EP update, all we need is the *marginal* distribution $Q(a_i)$. Just as most other variational "message-passing" approximate inference methods, EP can be seen as an iterative algorithm, improving estimates of the marginals $Q(a_i)$, $i = 1,\ldots,n$ until convergence. An EP update is *local*, in that its input is a marginal $Q(a_i)$ and it affects single site parameters b_i, π_i only. However, this *globally* affects all other marginals, which have to be updated through Gaussian propagation.

In common variational algorithms applied to discrete structured graphical models, such corrections of marginal estimates are performed by passing messages along the graph. In our case, the

11. Our experiments comparing Laplace and Gaussian priors in Section 7.1 illustrate this fact very nicely.
12. Our arguments about locality assume that a neighborhood structure can be imposed on a, say neighboring pixels in an image.
13. It is not known whether P. S. Laplace thought about this problem or even fixed it.

Algorithm 1 Expectation propagation algorithm for sparse linear model.

Require: X, u, τ, σ^2, η
  $b = 0$. $\pi = \varepsilon 1$. Compute initial representation of Q
  repeat
    for $i \in \{1, \ldots, n\}$ (random order) **do**
      Compute marginal $Q(a_i) = N(a_i | h_i, \sigma^2 \rho_i)$ from representation
      Do (fractional) EP update: $(b_i, \pi_i) \to (b_i', \pi_i')$
      Update representation of Q
    end for
    Refresh representation
  until marginal estimates $\{Q(a_i)\}$ converged

fully coupled Gaussian factor $P^{(0)}$ plays the role of the graph, and the messages are replaced by a *posterior representation* of $Q(a) = N(a|h, \sigma^2\Sigma)$. Just as with messages, the purpose of a representation is twofold: first, it needs to deliver mean and variance of an arbitrary marginal $Q(a_i)$ rapidly. Second, we need to be able to update it efficiently after each EP update. Our representations are given in Section 3.2, together with efficient update rules. Numerical errors can accumulate after many updates, so the representation is *refreshed* (i.e., recomputed from scratch) after each $O(n)$ EP updates. An iteration of EP updates over all (or most of the) sites is referred to as *sweep*. The structure of the EP approximate inference algorithm is given in Algorithm 1.

We close this section by remarking on the stopping rule we use in our EP implementation. One could stop once the site parameters do not change significantly anymore. However, we are really interested in the marginal means and variances, which in some cases are only weakly dependent on certain site parameters. For example, a large π_i means in general that the corresponding marginal mean is nailed down with a small variance, and increasing π_i further may have no large effect on the marginal distribution. Let $d(a,b) := |a-b| / \max\{|a|, |b|, 10^{-3}\}$ and $\Delta_i = \max\{d(h_i', h_i), \sigma d(\sqrt{\rho_i'}, \sqrt{\rho_i})\}$, where $Q(u_i) = N(h_i, \sigma^2\rho_i)$ and $Q'(u_i) = N(h_i', \sigma^2\rho_i')$ are the posterior marginals before and after an update at site i. We stop once $\max_i \Delta_i$ for a sweep over all sites is below some threshold.

3.2 Posterior Representation

In this section, we develop a representation of the posterior approximation $Q(a) = N(h, \sigma^2\Sigma)$ which allows efficient access to entries of h, $\operatorname{diag}\Sigma$ (marginal moments), and which can be updated robustly and efficiently for single site parameter changes (after EP updates). In fact, we propose two different representations: a degenerate and a non-degenerate one. The former is only useful in the underdetermined case ($m < n$), its updates are less numerically stable and more complicated, but it scales as $O(m^2)$, while the non-degenerate one scales as $O(n^2)$. If $m \ll n$, the degenerate representation leads to large computational savings.

We begin with the simpler non-degenerate representation:

$$\Sigma^{-1} = X^T X + \Pi = LL^T, \quad \gamma := L^{-1}(b^{(0)} + b),$$

where $\Pi := \operatorname{diag}\pi$ here and elsewhere. $L \in \mathbb{R}^{n,n}$ is the lower-triangular Cholesky factor (Horn and Johnson, 1985). Recall that $b^{(0)} = X^T u$. Note that $h = L^{-T}\gamma$. The marginal $Q(a_i) = N(h_i, \sigma^2\rho_i)$

is determined as $h_i = v^T\gamma$, $\rho_i = \|v\|^2$, where $v = L^{-1}\delta_i$. Here, δ_i is the Dirac unit vector with 1 at position i, and 0 elsewhere. This costs $O(n^2)$ (single back-substitution). After an EP update $b_i \to b_i'$, $\pi_i \to \pi_i'$, we have that

$$L'(L')^T = LL^T + (\pi_i' - \pi_i)\delta_i\delta_i^T, \quad L'\gamma' = L\gamma + (b_i' - b_i)\delta_i.$$

L', γ' are computed from L, γ using a Cholesky rank one update (downdate) for positive (negative) $\pi_i' - \pi_i$. This can be done in $O(n^2)$, we use a modification of the `LINPACK` routines `dchud`, `dchdd` (Dongarra et al., 1979), see Seeger (2004) for details. The update (downdate) is not done if $|\pi_i' - \pi_i|$ is too small. The reader may wonder why we do not represent and update Σ directly, using the Woodbury formula (see below). However, this would be numerically less stable than the Cholesky representation suggested here, and the operation count is the about the same.

In the underdetermined case $m < n$, another *degenerate* representation can be used, which leads to large savings if $m \ll n$. We noted in Section 3 above that Q is well-defined only if all $\pi_i > 0$. For numerical stability (with the degenerate representation), we require that $\pi_i \geq \kappa$ at all times, where $\kappa > 0$ is a small constant (we use $\kappa = 10^{-8}$ presently). This constraint is enforced in all EP updates. We can use the Woodbury formula (Henderson and Searle, 1981) in order to write

$$\begin{aligned}\Sigma &= \left(X^T X + \Pi\right)^{-1} \\ &= \Pi^{-1} - \Pi^{-1}X^T\left(I + X\Pi^{-1}X^T\right)^{-1}X\Pi^{-1}.\end{aligned}$$

We represent this via the lower-triangular Cholesky factor L in

$$LL^T = I + X\Pi^{-1}X^T.$$

Furthermore, let $\gamma := L^{-1}X\Pi^{-1}(b^{(0)} + b)$, whence

$$h = \Sigma(b^{(0)} + b) = \Pi^{-1}\left(b^{(0)} + b - X^T L^{-T}\gamma\right),$$

thus both h and Σ are represented by L, γ. For not too small κ, this representation is numerically stable. The marginal $Q(a_i)$ is obtained as $\rho_i = \pi_i^{-1}(1 - \pi_i^{-1}\|v\|^2)$, $h_i = \pi_i^{-1}(b_i^{(0)} + b_i - v^T\gamma)$, where $v := L^{-1}x$ with $x = X_{\cdot,i}$. After an EP update $b_i \to b_i'$, $\pi_i \to \pi_i'$, the representation is modified as follows. Let $\Delta_1 := (b_i^{(0)} + b_i')/\pi_i' - (b_i^{(0)} + b_i)/\pi_i$, $\Delta_2 := (\pi_i')^{-1} - \pi_i^{-1}$. We have that

$$L'(L')^T = LL^T + \Delta_2 xx^T, \quad L'\gamma' = L\gamma + \Delta_1 x.$$

Just as above, L', γ' can be computed from L, γ as a Cholesky rank one update/downdate, at the cost of $O(m^2)$. We do not modify π_i and the representation if $|\Delta_2|$ falls below some small threshold.

All in all, we can use a representation of Q whose size, as well as cost of a single site update, is quadratic in the smaller of n and m. Beware that L, γ have different definitions in the two cases. Note that we can also use the non-degenerate representation in the case $m < n$. In general, the non-degenerate representation leads to more numerically stable computations (supposedly because the Woodbury formula is not used), which are in fact more efficient in practice once $m \approx n/2$. We recommend to use the degenerate representation only if significant computational savings are observed in practice.

In some experimental design applications, such as gene network identification considered here, $m \ll n$ initially, but m grows up to $n/2$ eventually. In such cases, one could be tempted to use the degenerate representation initially, then switch to the non-degenerate one. In general, this does not make sense, since the majority of the computational effort is spent in the later stages anyway, and the non-degenerate representation should be used throughout.

3.3 The EP Update

An EP update works by matching moments between a tilted and the new posterior distribution. For an update at site i, we require the marginal $Q(a_i) = N(h_i, \sigma^2\rho_i)$ only, which is obtained from the Q representation. The moment matching requires the computation of Gaussian expectations with $t_i(a_i)$, a univariate quadrature which in general is not an analytical computation.

If $Q^{\backslash i}(a_i) = N(h_{\backslash i}, \sigma^2\rho_{\backslash i})$, we have that

$$\rho_{\backslash i} = \frac{\rho_i}{1-\rho_i\pi_i}, \quad h_{\backslash i} = \frac{h_i - \rho_i b_i}{1-\rho_i\pi_i}.$$

If the degenerate representation is used, it is more stable to compute the cavity marginal directly. Namely, if $v := L^{-1}X_{\cdot,i}$, then $\rho_{\backslash i} = \|v\|^{-2} - \pi_i^{-1}$ and $h_{\backslash i} = (b_i^{(0)} - v^T\gamma)/\|v\|^2 + b_i/\pi_i$.

Next, we need to compute mean and variance of $\hat{P}_i(a_i) = Z_i^{-1}Q^{\backslash i}(a_i)t_i(a_i)$, which we do as described in Seeger (2003), Appendix C.1.3. Note that $Z_i = \mathrm{E}_{Q^{\backslash i}}[t_i(a_i)]$, and define $\beta_i := (d\log Z_i)/(dh_{\backslash i})$, $\nu_i := -(d^2\log Z_i)/(dh_{\backslash i}^2)$. The concrete computation of β_i, ν_i (or equivalently, of the first and second moment of $\hat{P}_i(a_i)$) can be done analytically for Laplace sites, but is not straightforward due to issues of numerical stability, it is described in Appendix A. Then, the new site parameters are given by

$$\pi_i' = \frac{\sigma^2\nu_i}{1-\sigma^2\nu_i\rho_{\backslash i}}, \quad b_i' = \frac{\sigma^2(\beta_i + h_{\backslash i}\nu_i)}{1-\sigma^2\nu_i\rho_{\backslash i}}.$$

We show in Section 3.5 that $\nu_i \geq 0$, thus $\pi_i' \geq 0$, due to the log-concavity of t_i. If $\pi_i' < \kappa$ and the degenerate representation is used, we set $\pi_i' = \kappa$.

The numerical difficulties with the EP update for Laplace sites are remarkable, given that no such problems occur in several other EP applications, for example Gaussian process classification (GPC) with probit or logit noise (Minka, 2001b; Opper and Winther, 2000; Lawrence et al., 2003), where less careful implementations still work fine, and even approximate Gaussian quadrature can be used. Several early attempts of ours led to complete failure of the algorithm on realistic data (in the underdetermined case), motivating the fairly elaborate solution in Appendix A. While we cannot offer a firm explanation for this yet, our intuition is that the effect of Laplace prior sites on the posterior is much stronger, trying to emulate the essentially discrete feature selection process in a "unimodal" manner. Our findings also shed some sceptical light on proposals to implement a generic toolbox for EP, applying Gaussian quadrature[14] to do EP updates for general sites (Zoeter and Heskes, 2005). In the gene network identification application, we ran into problems of numerical instability coming from the combination of Laplace sites with very underdetermined coupling factors $P^{(0)}$. We suspect these problems are inherent, and in our case could be handled only by considering a modification of EP, as discussed just below.

3.3.1 FRACTIONAL EP UPDATES

We just mentioned the numerical difficulty of doing EP updates with Laplace sites in the strongly underdetermined case $m < n$. A frequent cause of numerical problems with EP is sloppiness in

14. Gaussian quadrature would fail completely for sites like the Laplace, which are not smooth functions. A central assumption with virtually all quadrature methods today is that the integrand up to a predefined weight function can be closely approximated by a low-order polynomial. Note that Monte Carlo integration is usually not considered useful for (low-dimensional) quadrature, due to its poor relative accuracy.

the implementation. For example, representation updates based on the Woodbury formula are a frequent source of accumulation of round-off and cancellation errors (see Section 3.2). The EP update with Laplace sites is quite difficult to do in a stable way (see Appendix A).[15] However, even using all these careful measures did not allow us to run standard EP on many of the gene network identification problems of Section 7.1 or on a fraction of the image coding problems of Section 7.2. We think that these stability problems of EP are inherent for some tasks, giving some motivation below. Fortunately, EP can be modified to use fractional updates, which in fact counter exactly the numerical problems we face. While fractional EP has been suggested as alternative to standard EP (Minka, 2004), its role for circumventing stability problems has not been noted so far to our knowledge.

Recall from Section 3 that if we set all or most of the $\pi_i = 0$ in the underdetermined case, the variance of most marginals $Q(a_i)$ is infinite. We face this problem by ensuring that $\pi_i \geq \kappa$ at all times. Still, at least for some updates, the cavity marginal variance of $Q^{\setminus i}(a_i)$ is huge. This is because we divide through the site approximation $\tilde{t}_i(a_i)$, whose $\pi_i \geq \kappa$ keeps the variance small. The variance is not infinite due to the effect of the other $\pi_j \geq \kappa$ and the coupling through $P^{(0)}$, but in many underdetermined situations, this coupling is weak. We then try to do an EP update based on a very wide cavity distribution $Q^{\setminus i}(a_i)$ and a quite narrow site $t_i(a_i)$ (enforcing a strong sparsity constraint requires a rather large τ). This is inherently difficult to do.

It would be better to make $Q^{\setminus i}(a_i)$ narrower and $t_i(a_i)$ wider, which is exactly what happens in *fractional EP updates*. Here, we obtain $Q^{\setminus i}(a_i)$ by dividing out only a fraction of $\tilde{t}_i(a_i)$, and $\hat{P}_i(a_i)$ by multiplying with only a fraction of $t_i(a_i)$. This idea is fairly natural, simply imagine the sites being replicated q times, then taken to the power of $\eta = 1/q$ to obtain the original setup. The only difference to standard EP is that we tie the parameters of the corresponding fractional site approximation replicas. Of course, the idea is not limited to rational fractions. Some extensions and theory of this method are discussed by Minka (2004). Another view on fractional EP is that projections from standard EP's $\hat{P}_i$ to Q' are done based not on the relative entropy (see Section 3), but on an α-divergence depending on the fraction.

For the fraction parameter $\eta \in (0,1]$, let $Q^{\setminus i} \propto Q\tilde{t}_i^{-\eta}$ and $\hat{P}_i \propto Q^{\setminus i} t_i^{\eta}$. We choose the new site parameters b_i', π_i' such that the moments of $\hat{P}_i$ and Q' match. This can be incorporated into the derivations above by setting $\tilde{b}_i = \eta b_i$, $\tilde{\pi}_i = \eta\pi_i$, and $\tilde{\tau} = \eta\tau$. The cavity moments are computed as

$$\rho_{\setminus i} = \frac{\rho_i}{1 - \rho_i \eta \pi_i}, \quad h_{\setminus i} = \frac{h_i - \rho_i \eta b_i}{1 - \rho_i \eta \pi_i}.$$

For the degenerate representation, a direct computation may be more stable:

$$\rho_{\setminus i} = \pi_i^{-1} \frac{R}{1 - \eta R}, \quad h_{\setminus i} = \pi_i^{-1} \left(\frac{b_i^{(0)} - v^T \gamma}{1 - \eta R} + b_i \right), \quad R = 1 - \pi_i^{-1} \|v\|^2.$$

We then compute $\tilde{b}_i', \tilde{\pi}_i'$ as above, using $\tilde{\tau} = \eta\tau$ instead of τ in the Laplace site, so that $\hat{P}_i$ and $\propto Q^{\setminus i}\tilde{t}_i(\cdot|\tilde{b}_i', \tilde{\pi}_i')$ have the same moments. Fractional updates are easily implemented for sites $t_i(a_i|\tau)$ with some hyperparameter τ, such that $t_i(a_i|\tau)^{\eta} = t_i(a_i|\eta\tau)$. The Laplace site is of this kind, if the normalisation constant of $\tau/(2\sigma)$ is dropped (it does not affect mean or variance of $\hat{P}_i$). Note that in

15. It is even harder to do for certain non-log-concave sites. For example, the sparse linear model with Student's t prior would be very hard to address with standard EP (Malte Kuss, pers. comm.).

general, t_i^η is log-concave if t_i is. Finally, the site parameters are updated as

$$b_i' = (1-\eta)b_i + \tilde{b}_i', \quad \pi_i' = (1-\eta)\pi_i + \tilde{\pi}_i',$$

upon which $\hat{P}_i$ and the new Q' have the same moments.

Another idea of making EP run smoother on hard problems is *damping* (Minka, 2001a). There, the full standard EP update is computed, but the site parameters are updated to a convex combination of old and proposed new values. This addresses a quite contrary problem to ours here. Damping is useful if EP update computations are stable, but lead to an improper new posterior, or the propagation of the updated information fails. If EP is viewed as finding a saddle point of a free energy approximation (Opper and Winther, 2005), damping can be understood as a step-size rule within this process. It slows down convergence in general in situations where EP without damping works fine, but the fixed points are not altered. Our problem is not solved by damping, since proposed new values for the site parameters cannot even be computed.

Finally, the reader may wonder whether the problems with standard EP are due to a bad initialisation of the site parameters. While we have not analysed it in all details, we think the problem is inherent. For example, we tried to run fractional EP to convergence, then start standard EP (with $\eta = 1$) from the fractional fixed point. On critical cases, this fails about as fast as if started in the usual way, often in the first sweep of standard EP.

3.4 Inclusion of a New Point

Suppose we would like to operate inference in the sparse linear model in a sequential manner, in that new data points (x_*, u_*) become available over time. This is the case in sequential design applications, since single experiments result in new measurements. In this section, we show how the EP posterior representation is updated once a new point (x_*, u_*) is added to the current data set D. The inclusion of (x_*, u_*) works in two stages. First, the Gaussian base measure is modified in order to incorporate the new point. Second, EP updates are done until convergence. The mechanics of the latter have been described above, so we can concentrate on the first stage here.

For the non-degenerate representation, let $v := L^{-1}x_*$. The change of $b^{(0)}$ results in $\tilde{\gamma} = \gamma + u_* v$. Since $L'(L')^T = LL^T + x_* x_*^T$, L', γ' is obtained from $L, \tilde{\gamma}$ by a rank one Cholesky update (see Section 3.2). The cost is $O(n^2)$.

For the degenerate representation, let $X' = (X^T, x_*)^T \in \mathbb{R}^{m+1,n}$ and $u' = (u^T, u_*)^T \in \mathbb{R}^{m+1}$. Since $b^{(0)} = X^T u$, we have that $b^{(0)\prime} = b^{(0)} + u_* x_*$. Let $l := L^{-1} X \Pi^{-1} x_*$. Then, $\tilde{\gamma} = \gamma + u_* l$ incorporates the update of $b^{(0)}$. Next, LL^T grows by a row/column $((Ll)^T, 1 + x_*^T \Pi^{-1} x_*)^T$. Therefore, L', γ' are obtained from $L, \tilde{\gamma}$ by a Cholesky extension, as described in Seeger (2004). The cost of the inclusion is $O(m^2)$.

3.5 Some Consequences of Log-concavity

A nonnegative function $f(x)$ is *log-concave* if

$$f(\lambda x_1 + (1-\lambda)x_2) \geq f(x_1)^\lambda f(x_2)^{1-\lambda}$$

for all x_1, x_2, and $\lambda \in [0,1]$. $f(x)$ is log-concave iff $\log f(x)$ is concave as a generalised function, which can take on the value $-\infty$, see Boyd and Vandenberghe (2002), Sect. 3.5. We call a distribution log-concave, if its density exists and is log-concave. In this section, we show some implications of log-concave sites for the numerical stability of EP.

Gaussians are clearly log-concave, so models of the sort considered here are log-concave if the sites are (products of log-concave functions are log-concave). For example, Laplace sites $t_i(a_i)$ are log-concave, while Student's t sites are not. A direct consequence is that for log-concave sites, the posterior is log-concave, so its unique mode can be found by convex optimisation. Log-concavity is stronger than unimodality though. For example, all upper level sets (areas enclosed by contours) of the posterior are convex sets. Intuitively, log-concave distributions are "simple", although strong consequences of this fact for variational approximate inference methods are not known to our knowledge.[16] Our main result is the following theorem.

Theorem 1 *Let EP be applied to a model with true posterior of the form*

$$P(a|D) \propto P^{(0)}(a) \prod_i t_i(a_i),$$

where $P^{(0)}(a)$ is a joint unnormalised Gaussian factor, and the sites $t_i(a_i)$ are log-concave. Suppose the site parameters π_i are initialised to non-negative values. Then, all EP updates are computable (in exact arithmetic), and all π_i remain non-negative throughout.

The proof is given in Appendix A.1. The theorem holds just as well for general sites $t_i(a)$ with corresponding site approximations $\tilde{t}_i(a) = N^U(\sigma^{-2}b_i, \sigma^{-2}\Pi_i)$, if "$\pi_i \geq 0$" is replaced by "$\Pi_i$ positive semidefinite". It hinges on a fundamental marginalisation theorem for log-concave functions due to Prékopa, see Bogachev (1998). Namely, suppose that $f(x,y)$ is jointly log-concave in (x,y), $x \in \mathbb{R}^p$, $y \in \mathbb{R}^q$. Then $\int f(x,y)\,dy$ is log-concave in x. Theorem 1 implies that EP can be implemented in a numerically stable way. Namely, the non-negativity of all π_i ensures that the representations introduced in Section 3.2 can be updated in a stable manner. The situation for some applications with non-log-concave sites is much less satisfactory. It is usually not possible to keep all π_i positive anymore, without making significant approximation errors (Minka, 2001a). Full EP updates lead to erratic behaviour or cannot even be done, and damping has to be used, leading to slower convergence. Negative entries π_i can lead to very ill-conditioned Cholesky factors in the representations, resulting in large errors at each update.

Our theorem also implies that for applications where EP is started with $\pi = 0$, for example Gaussian process classification, we have that the entropy $\mathrm{H}[Q]$ of the posterior decreases monotonically during the first sweep. Namely, the entropy is $\log|\Sigma|$ up to constants, which is decreasing in every single π_i. Minka (2001a) notes that the first sweep of EP is equivalent to a method called *assumed density filtering* (Kushner and Budhiraja, 2000), so our theorem has implications for this method as well.[17]

Another interesting consequence of log-concavity holds for the sparse linear model, independent of whether EP is used for approximate inference or not. It serves to motivate the parameterisation of the Laplace sites (2) in terms of $\tilde{\tau} = \tau/\sigma$. Up to additive constants, $\log P(u,a)$ has the form

$$(2m+n)\log\sigma^{-1} - \frac{1}{2}\|u/\sigma - Xa/\sigma\|^2 - \tau\sum_i |a_i/\sigma|,$$

16. In contrast, MCMC sampling from log-concave distributions has been proven to be computationally efficient (Lovász and Vempala, 2003).

17. The entropy $\mathrm{H}[Q]$ can increase in later sweeps of EP (this happens regularly, not only in special cases). This is why we need to consider Cholesky *downdates* in Section 3.2, and shows that $\mathrm{H}[Q]$ alone cannot be used to prove convergence of EP.

which is jointly concave in (ϕ, σ^{-1}), where $\phi := a/\sigma$. This fact has been noted in Park and Casella (2005). In fact, even $P(u,a)\sigma$ is log-concave in (ϕ, σ^{-1}), since $2m+n \geq 1$. The marginal likelihood $P(u)$ is a crucial criterion when it comes to hyperparameter optimisation or Bayesian tests (see Section 5). Now,

$$P(u) = \int P(u,a)\, da = \int P(u,a)\sigma\, d\phi,$$

and by the marginalisation theorem, $P(u)$ is log-concave in σ^{-1}. This implies that if all other hyperparameters are fixed, the empirical Bayesian maximisation of $\log P(u)$ w.r.t. the noise variance σ^2 is in fact a convex problem with a unique solution. Unfortunately, this property does not extend to other hyperparameters such as τ or X. On a practitioner's level, it is interesting to relate this fact to a scheme mapping out the entire regularisation path of Lasso (or, equivalently, an SVM) (Hastie et al., 2004). In either case, adjusting one hyperparameter trading off prior and likelihood given all others is shown to be simple. Here, as there, this gives some reassurance if σ^2 is adapted along with other parameters (see Section 7.2).

We close this section by some technical side comments for readers interested in details, all others may skip this paragraph. We require results from Section 5. We just showed that the exact $\log P(u)$ is concave in σ^{-1}, but how about the EP approximation of this quantity, called L in Section 5? To answer this question, we first have to establish that L is well-defined and continuous as a function of σ^2 in the first place. Now, L is defined in terms of the site parameters at convergence, and the EP algorithm has not been proven to always converge uniquely. Opper and Winther (2005) show that L is a proper approximate free energy as function of b, π, but the site parameters at convergence are only a saddle point thereof. Using tools such as the implicit function theorem, one can argue that $L(\sigma^2)$ is well-defined across some range, but does this hold globally across all σ^2? If the dependence of the site parameters on σ^2 is ignored locally, then L is log-concave in σ^{-1}, following similar arguments as above. We know from Section 5 that for computing the first derivative w.r.t. σ^2, the site parameters can be assumed constant, but this is not true in general for the second derivative (which would characterise concavity). Clearly, there is more work needed to gain a better understanding of such properties of the implicitly defined EP approximate free energy L.

4. Sequential Optimal Design

The role of sequential optimal design[18] for saving on expensive experiments has already been motivated in Section 2. The topic is well-researched in classical and Bayesian statistics (Fedorov, 1972; Chaloner and Verdinelli, 1995). A variant is known in machine learning as *active learning*[19] (Seung et al., 1992). We follow MacKay (1991) here, whose setting is closest to ours.

In the sparse linear model, a typical design problem can be formulated as follows. Given a set of candidate points x_*, at which of these should a corresponding target value u_* be sampled in order to obtain as much new information about the unknown a as possible? Assuming (for the moment) that u_* is known for a x_*, natural scores quantify the decrease in posterior uncertainty or gain in

18. *Optimal design* is a fixed term in statistics for a methodology, in which designs are optimised. We have no intention of claiming that any of the methods presented here solve problems in an optimal way, in fact they usually do not. In the context of this paper, *optimal design* and *experimental design* mean the same thing.
19. Confusingly, *active learning* is also used for the related, but not identical setup, where data comes in sequentially, and the method has to decide which cases to incorporate versus which to ignore. We are not interested here in this latter setting.

information from the current posterior Q to the novel Q' which is obtained by including (x_*, u_*) into the data set D. In this paper, we concentrate on the information gain $\mathrm{D}[Q' \,\|\, Q] = \mathrm{E}_{Q'}[\log Q' - \log Q]$. A large information gain means that Q' is different from Q, thus much novel information is gained from (x_*, u_*). Now, u_* is *not* known for the candidates x_*. Bayesian methodology dictates that u_* is averaged over its current posterior $Q(u_*|x_*, D) = \int P(u_*|x_*, a) Q(a|D)\, da$. Since $Q(a|D)$ is Gaussian for our approximation, the posterior over u_* is Gaussian as well. A natural score for x_* is the expected information gain $\mathrm{E}_{Q(u_*|x_*,D)}[\mathrm{D}[Q' \,\|\, Q]]$. This one-dimensional integral can easily be approximated using Gaussian quadrature.

However, optimal design for the gene network application of Section 2 does not fall into this standard category and requires some additional thoughts. The goal is to score the utility of inclusion of candidate controls u_*, given current data D (and posterior Q). Among a list of candidates, the highest-scoring u_* is then subjected to a new experiment in order to obtain x_*, whence (u_*, x_*) are included to form $D' = D \cup \{(u_*, x_*)\}$ and a new posterior Q'. Here, the posterior is a product of independent factors for the rows of A, so that for given (x_*, u_*), the information gain is the sum of $\mathrm{D}[Q' \,\|\, Q]$ over the posterior factors, where $(x_*, u_{*,j})$ is appended to D for the j-th factor.

More importantly, it is x_* which is unknown, rather than u_* in the standard setup. While $Q(u_*|x_*, D)$ is a Gaussian in our setup, $Q(x_*|u_*, D) = \int P(x_*|u_*, A) Q(A|D)\, dA$ is not a simple distribution. However, we can easily sample from it by first drawing $A \sim Q(A|D)$, then[20] $x_* = A^{-1}(u_* - \varepsilon)$, $\varepsilon \sim N(0, \sigma^2 I)$. Sampling from $Q(A|D)$ is discussed in Appendix B.2. Our *information gain* score in the gene network application is

$$S(u_*; D) = \mathrm{E}_{Q(x_*|u_*,D)} \left[\mathrm{D}[Q' \,\|\, Q] \right],$$

where the expectation is approximated by using a number of independent samples x_*.

Going back to the standard setup, for fixed (u_*, x_*), Q' is obtained from the current Q by first modifying the base measure $P^{(0)}$ corresponding to the inclusion, then updating the site parameters b, π. The problem with this is that the EP updates are expensive, so only few candidates could be scored for each inclusion.[21] A simpler and much cheaper alternative is to *approximate* the information gain by modifying $P^{(0)}$ only, but keeping the old site parameters, when defining Q' in $\mathrm{D}[Q' \,\|\, Q]$. In other words, for the purpose of scoring, we treat the model as purely linear-Gaussian, with Q as "effective Gaussian prior". This simple score can be computed very efficiently and reliably, so many candidates can be scored. Details are given in Section 4.1. Recall that the score for the gene network setup is the sum of information gains for the posteriors of each row of A.

Once a candidate is chosen for inclusion, a true experiment is done in order to obtain a complete new data point. In the standard setup, this means drawing u_*, given x_*, but in the gene network setting, we determine x_* for given control u_*. The new information is then included by a posterior update, as described in Section 3.4, and the site parameters are driven to new convergence by the EP algorithm.

Note that the fact that we approximate the true posterior $P(a|D)$ by a Gaussian $Q(a)$, as well as use Q' with the same site parameters as Q, means that we merely approximate the information gain, and at present we cannot give useful approximation guarantees, beyond our empirical demonstrations that good designs are usually found. However, our use of Gaussian Q and a simple update

20. We use a LU decomposition of A. The cost of $O(n^3)$ may be prohibitive for large n (although the same A sample can be used to score *all* candidates), in which case we would recommend sparsifying A and using a sparse matrix solver.
21. One idea would be to update few sites only after each inclusion. The extension described in Section 6.3 could be used to implement this, which is however not done here.

$Q \to Q'$ means that our information gain approximation can be computed robustly, which, as noted in Section 1, is often as important for experimental design as is high approximation accuracy.

4.1 Simple Information Gain

The simple information gain score is $\mathrm{D}[Q' \,\|\, Q]$, where Q is the current posterior, and Q' is obtained from Q by including (x_*, u_*) into the base measure $P^{(0)}$ as discussed in Section 3.4, but leaving the site parameters b, π at their old value. The relative entropy between Gaussians is well known:

$$\begin{aligned}\mathrm{D}[N(h', \sigma^2\Sigma') \,\|\, N(h, \sigma^2\Sigma)] = &\frac{1}{2}\log|M| + \frac{1}{2}\mathrm{tr}\left(M^{-1} - I\right) \\ &+ \frac{1}{2}\sigma^{-2}(h'-h)^T\Sigma^{-1}(h'-h), \quad M := (\Sigma')^{-1}\Sigma.\end{aligned} \tag{4}$$

Importantly, in our case we have that $(\Sigma')^{-1} = \Sigma^{-1} + x_* x_*^T$, so that $M = I + x_* x_*^T\Sigma$ has a simple form. This allows us to compute the simple information gain very efficiently. Details are given in Appendix B.1.

4.2 Marginal Criteria

The simple information gain scored discussed in the previous section measures the distance between the *joint* distributions Q and Q' (before and after inclusion). However, inference schemes such as expectation propagation (and other variational ones) are designed to approximate the posterior marginals well. EP applied to models with Gaussian base distribution results in a full joint posterior approximation Q, which can of course be used to make decisions or to compute information scores, but very little is known about the quality of Q beyond its marginals. A careful approach would therefore base experimental design scores on the marginals of Q only. On the other hand, criteria based on the full joint posterior can be more powerful in order to distinguish between many candidates.

For such marginal scores, we need to know how the marginals change after an update of $P^{(0)}$. Let $h, \mathrm{diag}\,\Sigma$ be the current marginal moments. We need to compute $h', \mathrm{diag}\,\Sigma'$ after inclusion of (x_*, u_*) (we only deal with the "simple" variant here, where no EP updates are done after the inclusion). Let $\alpha = x_*^T\Sigma x_*$, $z_* = \Sigma x_*$. By the Woodbury formula, we have that

$$\Sigma' = \Sigma - (1+\alpha)^{-1} z_* z_*^T.$$

The new h' is given in Appendix B.1, where we also show how to compute z_*, α efficiently. The marginal moments $h, \mathrm{diag}\,\Sigma$ have to be computed from the representation before each scoring round, although another idea is developed in Section 6.

Interestingly, in initial gene network identification experiments, employing the sum of marginal information gains worked less well than using the simple joint information gain of Section 4.1. In this case, the latter seems to carry more useful information about the candidates. In other words, the posterior correlations estimated by EP seem good enough to be useful here. Results with marginal scores are not reported in this paper.

5. The Marginal Likelihood

Bayesian methodology requires that unobserved variables are marginalised over in order to do predictions or to make optimal decisions. However, in many situations this is not practically feasible

for some variables. In the case of the sparse linear model, we can approximately integrate out the parameters a using EP, but likelihood and prior depend on other *hyperparameters* still, namely σ^2, τ, and (parameters of) X. A surrogate widely accepted amongst Bayesian practitioners is to *estimate* good values for the hyperparameters by maximising the *marginal likelihood* of the observed data.[22]

An approximation to the marginal likelihood may be obtained from EP. Details about this approximation can be found in Seeger (2005). As shown there, it is the same as the approximate free energy proposed in Opper and Winther (2005). We give the derivation for fractional EP in general (see Section 3.3.1), $\eta \in (0,1]$. Standard EP is obtained for $\eta = 1$. We have that

$$P(D) = P(u) = \int \prod_{i=1}^{n} t_i(a_i) P^{(0)}(a)\, da. \tag{5}$$

Recall that in EP, the sites t_i are replaced by approximations $\tilde{t}_i$ of Gaussian form. Earlier on, we did not bother with the normalisation constants of these approximations, but now we have to make them explicit: $t_i(a_i) \to C_i \tilde{t}_i(a_i)$, $\tilde{t}_i(a_i) = N^U(a_i|\sigma^{-2}b_i, \sigma^{-2}\pi_i)$. Roughly speaking, EP works by making the first and second order moments of the posterior marginals $Q(a_i)$ and the tilted distributions $\hat{P}_i(a_i)$ equal for all i. In this line, we fix the C_i such that the normalisation constants are the same as well:

$$\log C_i = \eta^{-1}(\log Z_i - \log \tilde{Z}_i), \quad Z_i = \mathrm{E}_{Q^{\setminus i}}\left[t_i(a_i)^\eta\right],\ \tilde{Z}_i = \mathrm{E}_{Q^{\setminus i}}\left[\tilde{t}_i(a_i)^\eta\right].$$

Here, $Q^{\setminus i} \propto Q\tilde{t}_i(a_i)^{-\eta}$. The EP approximation $L \approx \log P(u)$ is then obtained by replacing t_i by $C_i\tilde{t}_i$ in (5). This results in

$$L = \sum_{i=1}^{n} \log C_i + \frac{1}{2}\left(\log|\Sigma| + \sigma^{-2}h^T(b^{(0)} + b) - \sigma^{-2}\|u\|^2 + (n-m)\log(2\pi\sigma^2)\right). \tag{6}$$

In order to maximise L, we require its gradient w.r.t. hyperparameters, which can be computed exactly if EP is run to convergence, such that the moments of all $Q(a_i)$ and $\hat{P}_i(a_i)$ coincide. In Seeger (2005), the following is shown:

$$\nabla_{\theta^{(0)}} L = \mathrm{E}_Q[\nabla_{\theta^{(0)}} \log P^{(0)}(a)], \tag{7}$$

where $\theta^{(0)}$ are the natural parameters of $P^{(0)}$. Furthermore, if α is a parameter of the site t_i independent of $P^{(0)}$, then

$$\frac{\partial L}{\partial \alpha} = \frac{\partial \log Z_i}{\partial \alpha} = \mathrm{E}_{\hat{P}_i}\left[\frac{\partial}{\partial \alpha}\log t_i(a_i)\right]. \tag{8}$$

Note that this holds for fractional EP in general, if $\eta \in (0,1]$. The specialisations to our case are given in Appendix C.

Note that the EP approximation L of $\log P(u)$ has an important consistency property. It is well known that $\nabla_{\theta^{(0)}} \log P(u) = \mathrm{E}_{P(a|D)}[\nabla_{\theta^{(0)}} \log P^{(0)}(a)]$, from which (7) is obtained by replacing the true posterior by the EP approximation $Q(a)$: the true gradient of the approximate criterion is the approximate gradient of the true criterion. Another way to view this is to note that L depends on hyperparameters directly as well as through the EP site parameters b, π, thus the gradient has direct as well as indirect contributions. Importantly, the stationary conditions of EP at convergence

22. In work in progress, we show how the noise variance σ^2 can be integrated out along with a using EP. This, however, leads to a significantly more complicated and somewhat less robust algorithm. Details will be given in a later paper.

imply that the latter contributions do vanish. In Seeger (2005), it is shown that $(\partial L)/(\partial \sigma^{-2} b_i) = (\partial L)/(\partial \sigma^{-2} \pi_i) = 0$ at EP stationary points, which directly implies the simple formulae (7), (8). This fact becomes clear if L is seen as approximate free energy (Opper and Winther, 2005) (although only the case $\eta = 1$ is discussed there).

Note that the fraction η is a parameter of the approximation method, not a statistical variable or hyperparameter. It is natural to ask for which η one would obtain the best approximation of $\log P(u)$. Since L is not a bound on $\log P(u)$, we cannot directly optimise η. Useful theoretical insights about fractional EP variational free energies for different η are not known to us. We will address this point as part of an empirical comparative study, which is subject to future work. However, note that EP cannot be run at all in a stable way for certain setups if η is (very close to) one (see Section 7.1).

6. Large-Scale Applications

A naive implementation of the EP algorithm for the sparse linear model requires $O(n^3)$ time for each sweep and $O(n^2)$ memory, if the non-degenerate representation is used. While this is acceptable for moderate sizes of n, like in the gene network identification application, it is certainly not feasible for large n. In this section, we propose some ideas in order to apply our framework in such situations.

The dominant computation within our framework, both for experimental design and marginal likelihood maximisation, is spent performing a sweep of EP updates. Naively, this is a loop over all n sites (in random ordering). For each site i, the marginal $Q(a_i)$ has to be determined, and the representation for doing so has to be updated afterwards. Time can be saved by doing less than n updates per sweep, and by speeding up the marginal extraction or representation update. In a sequential context such as experimental design, it is sensible to assume that many EP updates will not lead to much change in Q, especially during later stages. A key problem is how to efficiently detect the sites whose update would change the current posterior the most.

Furthermore, a large-scale application will normally not be generic, but comes with a lot of structure already. For example, the design matrix $X \in \mathbb{R}^{m,n}$ is often given implicitly, since its storage alone would be too costly, and matrix-vector multiplications (MVMs) with X are often much more efficient than $O(nm)$. A key step in the direction of a large-scale implementation is to make sure that such special structure is used optimally.

A motivating example for a large-scale application comes from compressive sensing (see Section 2.4). Recall that $X = P\Phi$, where Φ is a fixed coding matrix, and P is a measurement matrix we can design. If the task deals with full images, n can be a million, and neither Φ nor P can be stored as dense matrices, but have to be defined implicitly as linear mappings. For example, Φ could be an orthonormal Wavelet code, and P could consist of m selected rows from the discrete cosine transform (DCT) matrix. An MVM with Φ costs $O(n)$, an MVM with P is $O(n\,log\,n)$, both much faster than a naive $O(nm)$ MVM for large m. Experiments with this setup are in preparation.

6.1 Matrix-free Updates

We already noted that storing X explicitly should not be necessary, if we have an efficient method to compute MVMs with X and X^T. For example, X could be of special structure, or it could be sparse (most entries exactly zero). Suppose that $m \ll n$ with very large n. In this case, the degenerate representation of $Q(a)$ is used, requiring $O(m^2)$ storage. Each EP update requires $O(m^2)$ and the extraction of a column of X, in other words $X\delta_i$, a single MVM (see Section 3.3). It is necessary to refresh the representation now and then, which requires the computation of $X\Pi^{-1}X^T$ for arbitrary

positive π_i. This can be reduced to computing $X\Pi^{-1}X^T\delta_i$, $i = 1,\ldots,m$, corresponding to m MVMs with X and X^T each.

6.2 No Representation At All

It is in general not feasible to compute all required marginals on demand, just storing π, b, and the data. A representation is used in order to do this feasibly, using the fact that each update leads to a rank one modification only. Time is traded for memory, as explained in Section 3.2.

However, suppose that MVMs with X and X^T can be done very efficiently. For an update at site i, we require $Q(a_i) = N(h_i, \sigma^2\rho_i)$. Recall that $\Sigma^{-1} = X^TX + \Pi$ and $h = \Sigma(b^{(0)} + b)$. The quadratic criterion

$$q(v) := \delta_i^T v - (1/2)v^T(X^TX + \Pi)v$$

can be minimised using the linear conjugate gradients (LCG) algorithm (Saad, 1996), requiring a MVM with $X^TX + \Pi$ per iteration, thus MVMs with X, X^T, and $O(n)$. At the minimum, we have $v_* = \Sigma\delta_i$ and $q(v_*) = \rho_i/2$, whence $h_i = v_*^T(b^{(0)} + b)$.

We can also start from the degenerate representation and formulate the marginal computation as quadratic minimisation over vectors of size m, where the system matrix is $I + X\Pi^{-1}X^T$. However, both variants have the same cost of two MVMs per iteration, and their convergence behaviour should be similar.

This method has the advantage of not requiring any representation at all. Apart from the architecture for computing X and X^T MVMs, we need $O(n)$ memory only. However, it is useful only if LCG converges to satisfying accuracy rapidly (after many fewer than n iterations), and if single MVMs can be done much faster than $O(nm)$. Another drawback is that the marginal computations are approximate only, and the error may well depend on the current π. Therefore, it is maybe most sensible to combine it with a way of reducing the number of EP updates required, as is discussed just below.

6.3 Keeping Marginal Moments Up-to-date

The suggestions so far try to speed up single EP update computations. However, if n is very large, a major problem is that we cannot update all n sites in a sweep. For example, in the context of experimental design, it is not affordable to update each site after each new data point inclusion. Updates have to be done selectively on a subset. In this section, we indicate how this can be done. See Seeger and Nickisch (2008) for a demonstration in practice.

Given the marginal $Q(a_i)$, an EP update at i is $O(1)$, so its effect on $Q(a_i)$ can be measured cheaply. The costly part (in the formulation used so far) is to extract the marginal from the representation, and to update the latter. It is reasonable to assume that a small impact of an EP update on the marginal $Q(a_i)$ implies that the whole posterior Q changes little, so site i need not be updated at the moment.

In order to direct EP updates towards sites with maximum marginal impact, it is necessary to keep *all* marginals $Q(a_i)$ up-to-date at all times. In other words, $Q(a_i)$ must be computable in $O(1)$ from the representation, for any i. With the representations of Section 3.2, this costs $O((\min\{n,m\})^2)$. We concentrate on the degenerate representation, which is more important in the large-scale context (the non-degenerate case is also simpler). Let $V := X^TL^{-T}$, and define

$e_1 = \mathrm{diag} VV^T$, $e_2 = V\gamma$. Given the latter vectors, each marginal can be computed in $O(1)$:

$$\rho = \Pi^{-1}(1 - \Pi^{-1} e_1), \quad h = \Pi^{-1}(b^{(0)} + b - e_2).$$

We show how e_1, e_2 can be updated along with the representation. Recall Section 3.2, $x = X_{\cdot,i}$. We can compute Δ_1, Δ_2 and $Q'(a_i)$ in $O(1)$, without knowing $v = L^{-1}x = (V_{i,\cdot})^T$. If $|\Delta_2|$ or $\mathrm{D}[Q'(a_i) \,\|\, Q(a_i)]$ is too small, the update is rejected. Otherwise, we compute v and $w := L^{-T}v$, $r := X^T w$. First, $\tilde{\gamma} = \gamma + \Delta_1 v$. The Woodbury formula gives

$$\left(L'(L')^T\right)^{-1} = \left(LL^T\right)^{-1} - (\Delta_2^{-1} + e_{1,i})^{-1} ww^T,$$

so that $e_1' = e_1 - (\Delta_2^{-1} + e_{1,i})^{-1} r \circ r$, and

$$e_2' = V'(L')^{-1}(L\tilde{\gamma}) = e_2 + (\Delta_1 - (\Delta_2^{-1} + e_{1,i})^{-1} v^T \tilde{\gamma}) r.$$

Now, $v^T\tilde{\gamma} = e_{2,i} + \Delta_1 \|v\|^2 = e_{2,i} + \Delta_1 e_{1,i}$, so that

$$e_2' = e_2 + \frac{\Delta_1 - \Delta_2 e_{2,i}}{1 + \Delta_2 e_{1,i}} r.$$

Finally, L', γ' are obtained from $L, \tilde{\gamma}$ by a rank one Cholesky update as before. The cost is increased by the computation of w and r, the latter requires a single MVM with X^T.

The representation has to be recomputed now and then. Here, the computation of e_1 is most challenging, but can be reduced to doing m MVMs with X^T (with the columns of L^{-T}, obtained by back-substitutions).

In an experimental design context, the representation has to be updated once new data points (x_*, u_*) are included, as discussed in Section 3.4. Using the notation there, V is transformed to V' by appending the column $v := l_*^{-1}(x_* - X^T L^{-T} l)$, where (l^T, l_*) is the new row of L'. Therefore, $e_1' = e_1 + v \circ v$. Moreover, $\gamma' = ((\gamma + u_* l)^T, g_*)^T$ for a scalar g_*, so that

$$e_2' = V(\gamma + u_* l) + g_* v = e_2 + u_* x_* + (g_* - u_* l_*) v.$$

The computation of v requires one MVM with X^T.

Once every marginal is available in $O(1)$ at all times, we can actively select which one to update next. For a set of candidates i, we compute score values S_i, selecting site $\mathrm{argmax}_i S_i$ for the next update. A possible score is $S_i = \mathrm{D}[Q'(a_i) \,\|\, Q(a_i)]$. Scoring all sites for each update is $O(n)$, thus prohibitive, so a set of scoring candidates J should be maintained and evolved. A simple rule, which has been used in the context of sparse Gaussian process methods (Lawrence et al., 2003), works as follows. Before each update, all sites in J are scored. The winner is chosen for the update, and is removed from J, along with a fraction (say, $1/2$) of the worst-scored ones. J is then filled up again by drawing at random from $\{1, \ldots, n\} \setminus J$.

Finally, we note that it is possible in principle to maintain e_1, e_2, therefore the marginal moments, without storing a representation of size $O(m^2)$ at all. For example, the update after a change of π_i, b_i is in terms of $r = X^T L^{-T} v = VV^T \delta_i$. Since $VV^T = \Pi - \Pi\Sigma\Pi$, we have that $r = \pi_i(\delta_i - \Pi\Sigma\delta_i)$. We have shown above how to approximate $\Sigma\delta_i$ by the LCG algorithm. Equivalently, $VV^T = X^T(I + X\Pi^{-1}X^T)^{-1}X$, so we can also compute r by LCG on a system of size m. In principle, such a representation-free method can be used to address problems with large m. However, when working without a representation, we have no efficient possibility anymore to "refresh"

e_1, e_2 now and then. Moreover, using LCG is an additional source of approximation errors. The danger is that (e_1, e_2) and (π, b) drift away from the relation that binds them with exact computations. Experiments assessing the usefulness of such a representation-free treatment in comparison to a $O(m^2)$ representation are in preparation.

7. Experiments

In this section, we present experiments for gene regulatory network identification, for sparse coding of natural images, and for compressive sensing.

7.1 Regulatory Network Identification

Our application of experimental design to gene network identification, using the sparse linear model, has been described in Section 2.2. The material presented here is extracted from Steinke et al. (2007), where all details omitted here can be found. The experiments were done by Florian Steinke. Note that `Matlab` code is available[23] for scientific use. The results given here can be reproduced with this code.

In order to evaluate our method, we simulate the whole network identification process. First, we generate a biologically inspired ground-truth network together with parameters for a numerical simulator of nonlinear dynamics, respecting the network. We feed our method with a number of candidate perturbations $\{u_*\}$, among which it can choose the experiments to be done. If (say) u_* is chosen, a corresponding x_* is drawn from the simulator, and (u_*, x_*) is included into the posterior $Q(A)$ as new observation. We score the predictions from the current posterior against the true network after each inclusion.

Our generator samples networks with a scale-free edge distribution, using $n = 50$ nodes with in-degrees (excluding self-edges) in $\{0, \ldots, 6\}$. An edge is activating with probability $1/2$, inhibitory otherwise. For a given network structure, we sample plausible interaction dynamics, using noisy Hill-type kinetics inspired by the model of Kholodenko et al. (2002). Here, systems without a stable fixed point are rejected.

The disturbance candidates u_* were restricted to have a small number r of non-zero entries, since a tightly controlled excitation or inhibition for many genes at the same time is unreasonably expensive in practice. All non-zero elements have the same size, but a random sign, so that all u_* have the same norm. We use a pool of 200 randomly generated candidates in general.

All results are averaged over 100 runs with independently drawn networks and systems. In the comparative plots presented below, the different methods all run on the same data.

Our evaluation score measures the quality of the ranking of candidate edges, computed from the posterior according to the probabilities $Q(\{|a_{ij}| > 0.1\})$. We modify a standard ROC curve (true positive rate (TPR) as function of false positive rate (FPR)) by computing the area under the ROC curve (AUC) only up to a number of false positives equal to the number of edges in the true network. Namely, since true networks are sparse, there are many more non-edges than edges, and only very small FPRs are acceptable at all. We denote our score as *iAUC*, it is normalised to lie in $[0, 1]$. For $n = 50$, the trivial method which outputs a random permutation as ranking, has expected iAUC of 0.02. Furthermore, on average about 25% of the true edges are "undetectable"

23. See `www.kyb.tuebingen.mpg.de/sparselinearmodel/`. The code is joint work with Florian Steinke and Koji Tsuda. If you use it as part of a scientific publication, please cite Steinke et al. (2007) (details are on the web site).

after linearisation: their entries a_{ij} are very close to zero, so they do not contribute to the dynamics within the linearisation region. Such edges were excluded from the computation of iAUC.

Our method comes with two hyperparameters: the noise variance σ^2, and the scale τ of the Laplace prior. Given sufficient data, they could be estimated by the method described in Section 5, but this is hard to do in an experimental design setting, where we start with very few observations.[24] It is reasonable to assume that a good value for σ^2 does not change too much between networks with similar biological attributes, so that we can transfer it from a system whose dynamics are known, or for which sufficiently many observations are already available. This transfer was simulated in our experiments by generating 50 networks with data as mentioned above, then estimating σ^2 from the size of the ε residuals. The prior parameter τ was set by a simple heuristic described in Steinke et al. (2007).

We used fractional EP with $\eta = 1/2$. Standard EP (i.e., $\eta = 1$) does not converge for the majority of the inference tasks required. This problem is discussed in Section 3.3.1.

In Figure 2, we present reconstruction curves for our method versus competing techniques, which lack novelties of our approach (experimental design, Laplace prior). Very clearly, experimental; design helps to save on costly experiments. The effect is more pronounced for the Laplace than for the Gaussian prior. The former is a better prior for the task, and it is usually observed that improvements of designed over random experiments scale with the appropriateness of the model. In this case, the iAUC level 0.9 is attained after 36 experiments with designed disturbances, yet only after 50 measurements with randomly chosen ones, thus saving 30% of the experiments. In fact, our results indicate that experimental design only realises its full potential together with the non-Gaussian sparsity prior (see also Section 2.1).

In general, the model with Laplace prior does significantly better than with a Gaussian one. Of course, τ for the Laplace and the variance for the Gaussian prior were selected independently, specific to the prior. The difference is most pronounced at times when significantly less than n experiments have been done and the linear system (3) is strongly underdetermined. This confirms our arguments in favour of the Laplace prior (see Section 2.1).

The under-performance of the most direct variant LD of our method, up to about $n/2$ observations, is not yet completely understood. However, it has been repeatedly observed that aggressive experimental design based on very little knowledge can perform worse than random data sampling, if the model does not perfectly reflect the truth. On the other hand, it is important to note that LD recovers completely from the initial under-performance, and from $m = 25$ onwards significantly outperforms the random variant LR, so the initial design choices are not just plain wrong. We also tested a hybrid strategy LM of starting with random, then switching to designed experiments. In this particular application, starting from no knowledge about the network, an initial random exploration seems to lead to most useful results early on, while not hurting a subsequent sequential design.

7.2 Sparse Coding of Natural Images

The application of the sparse linear model to image coding (Olshausen and Field, 1997; Lewicki and Olshausen, 1999) is motivated in Section 2.3. Here, we present results of a study along the lines of work reported by Olshausen and Field (1997).[25] As is argued at length in Section 2.3, our goal here

24. One may be able to correct initial estimates of σ^2, as more observations are made, and a method for doing so is subject to future work.

25. Data and code used there was obtained from `http://redwood.berkeley.edu/bruno/sparsenet/`.

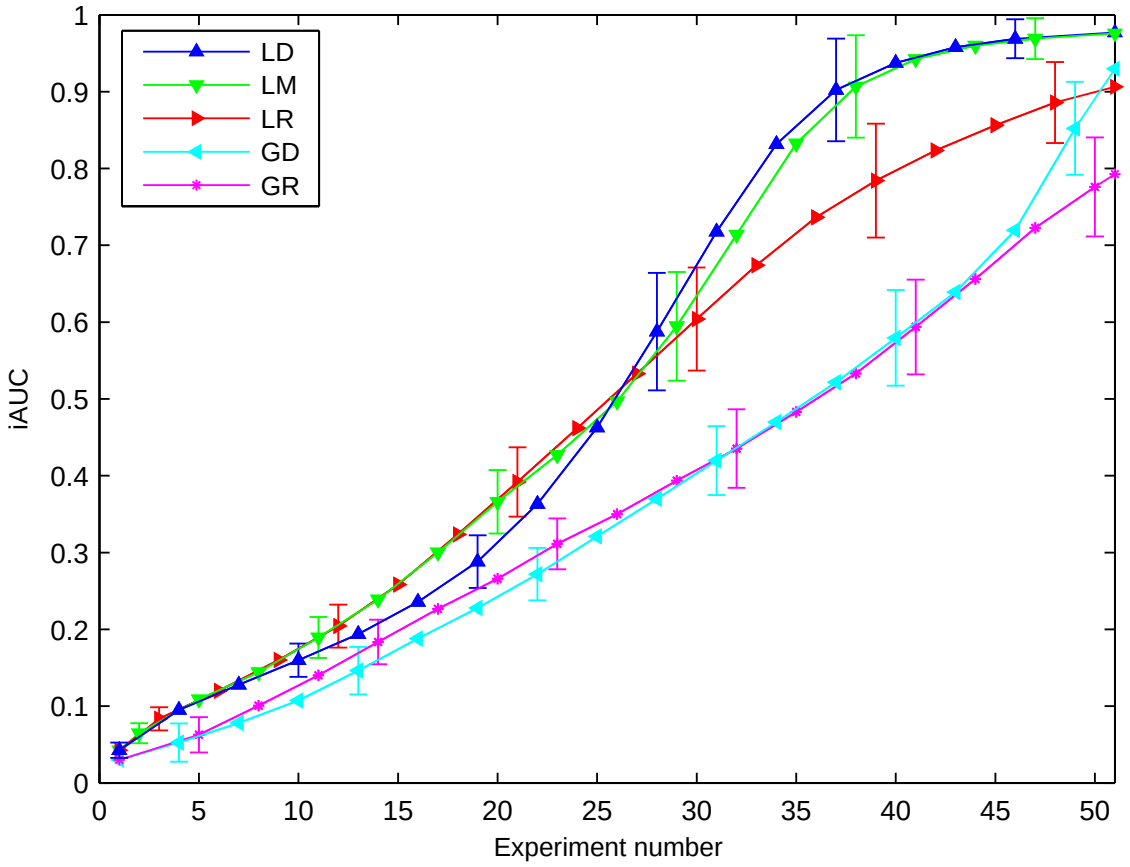


Figure 2: Reconstruction curves for experiments (gene expression changes of 1%, SNR 100, $r = 3$ non-zeros per u). **LD**: Laplace prior, experimental design. **LR**: Laplace prior, random experiments. **GD**: Gaussian prior, experimental design. **GR**: Gaussian prior, random experiments. **LM**: Laplace prior, mixed selections (first 20 random, then designed). Error bars show one standard deviation over runs. All visually discernible differences in mean curves of different methods are significant under the t-test at level 1%.

is not to compare a range of models to find out which can code images better or learn codes more efficiently, but rather to test the hypothesis put forward in Lewicki and Olshausen (1999), which does not call for such a comparison. We extracted two data sets of $r = 50000$ image patches of size 12×12 by subsampling the 10 whitened natural scenes, using their Matlab code. One is for training, the other for evaluation.[26] We allow for a twice overcomplete basis, therefore $m = 144$, $n = 288$. We also drew a random subsample of size 1000 from the test set, which was used in order to produce curves over many codes. Recall that a code in our model is given by the matrix X, whose columns are referred to as codebook vectors or filters.

26. The sets are not guaranteed to be completely distinct, although the extraction of the same patch during the different sampling runs is unlikely.

Before we describe the results, we should stress that the main aim at this point is to demonstrate the efficiency and usefulness of our method on a learning problem of large scale. 41473 hyperparameters are learned here on a set of 50000 cases. Each update step requires approximate inference on 100 models, each of which comes with 288 latent parameters. A more careful study will explore the implications of our findings to neuroscience (early vision) and image coding. This will entail a more refined learning schedule for our method, while our choices here are ad hoc and did not receive much tuning. Moreover, we base our evaluation entirely on the EP marginal likelihood approximation on a test set. An independent MCMC evaluation of this criterion, using for example annealed importance sampling together with hybrid Monte Carlo (which seems commonly accepted, but is very expensive to run), is clearly needed in order to draw any scientific conclusions (which we refrain from doing here). Such a study is subject to future work, and is not in the scope of this paper.

As noted above, the approach of Olshausen and Field (1997) is to approximate inference by maximum a posteriori (MAP): $P(a_j|u_j,X) \approx \delta_{\hat{a}_j}(a_j)$, where $\hat{a}_j$ is the posterior mode. Since the log posterior is concave, this mode is unique and can be found efficiently. In order to learn the code X, they propose to impute their estimates in order to obtain a complete data set $\{(u_j,\hat{a}_j)\}$, then to do maximum likelihood training. Since the estimate $\hat{a}_j$ depends on the current X, this is an iterative process. Their method will be called OF in the sequel. In contrast to this, we follow the hypothesis of Lewicki and Olshausen (1999) and learn X by maximising the EP approximation to the log marginal likelihood $\log P(D|X)$ (see Section 5). While Lewicki and Olshausen (1999) argue that the method of Olshausen and Field (1997) can be seen as optimising a (different) surrogate to $\log P(D|X)$ as well, ours is a much better approximation in general.[27]. Our method will be called EP here.

We had to modify their code in a minimal way, in order for it to run automatically on a given fixed training set. Our changes are detailed in Appendix E. Just as with our own method, we did not attempt to refine parameters for their code here.

The code of Olshausen and Field (1997) performs stochastic gradient descent on batches of size $|B| = 100$. We use a similar approach, which works as follows. The criterion $-\log P(D|X)$ is a sum of independent parts, one for each image. Let $\phi := -\sum_{j\in B} L_j$ be the EP approximation to this criterion, evaluated over a batch of size $|B|$ (here, L_j is the EP log marginal likelihood approximation on image j). The update rule for X is

$$X' = X - D', \quad D' = 0.85D + \xi X X^T \nabla_X \phi,$$

where $\xi > 0$ is the learning rate. The pre-multiplication of the gradient by XX^T is advantageous for this application, as has been argued in the context of "natural gradient" learning. As opposed to Olshausen and Field (1997) and Lewicki and Olshausen (1999), we adjust the noise variance σ^2 in the same way by minimising ϕ. If $l := \log\sigma^2$, a simple update rule is

$$l' = l - d', \quad d' = 0.85d + \xi_l \nabla_l \phi,$$

where $\xi_l > 0$ is a learning rate different from ξ. The learning rates are decreased in a reasonably slow way,

$$\xi(t) = \frac{A}{B+t}, \quad \xi_l(t) = \frac{A_l}{B_l+t},$$

27. MCMC experiments to strengthen this claim are subject to future work.

where t is the number of updates so far, and A, B, A_l, and B_l are free parameters (Bottou, 1998).

We can compare our update rule of X with the one used in Olshausen and Field (1997). It is easy to see that the gradient of the exact log marginal likelihood is

$$\nabla_X \log P(u_j|X) = \sigma^{-2} \mathrm{E}_{P(a_j|u_j,X)}[e_j a_j^T] = \sigma^{-2} \left((u - X\,\mathrm{E}[a_j])\mathrm{E}[a_j]^T - X\,\mathrm{Cov}[a_j] \right),$$

where $e_j := u_j - X a_j$. If OF is seen as optimising an approximation thereof, then the expectation over $P(a_j|u_j,X)$ is replaced by plugging in the mode $\hat{a}_j$ (which is what we mean by "imputation" above). In other words, the posterior mean $\mathrm{E}[a_j]$ is replaced with the mode, and the second term depending on the posterior covariance $\mathrm{Cov}[a_j]$ is neglected altogether. Since the Laplace sparsity prior leads to a posterior which is significantly skewed (towards coordinate axes, see Figure 1), mean and mode tend to be quite different.[28] In EP, the posterior expectations are replaced by $\mathrm{E}_Q[\cdot]$, where $Q = N(h, \sigma^2\Sigma)$ is the EP posterior approximation (see Appendix C). In fact, running OF precisely with the learning rule just stated does not work well in practice, and the neglectance of posterior uncertainty in the learning rule is put forward as a reason for this in Lewicki and Olshausen (1999). Olshausen and Field (1997) propose a heuristic renormalisation of the columns of X towards some "desired variance" as remedy, and this seems an important feature in their code. This heuristic comes with a number of parameters, which are fixed in their code to some values presumably optimised for their data by hand. In contrast, EP comes with τ, σ^2 only, and the latter can be adjusted automatically as well,[29] as is demonstrated here. Note that Lewicki and Olshausen (1999) suggest to approximate $\mathrm{Cov}[a_j]$ by the Laplace method in order to improve on the OF learning rule. However, as noted in Section 3, this method is not well-defined in case of the sparse linear model. Code implementing the proposal of Lewicki and Olshausen (1999) is not publicly available.

We can draw an analogy between the difference of learning X in OF and EP to current practices in speech recognition (Rabiner and Juang, 2003). Given a trained system, the recognition (or decoding) is done by searching for the most likely sequence, in what is called Viterbi decoding. However, training the system should be done by expectation Maximisation (EM), where the latent sequence is integrated out using inference. This is about what EP does here, with the difference that inference in hidden Markov models used for speech is analytically tractable, but has to be approximated here (by EP). However, since EM training is still computationally demanding, most speech recognition systems use Viterbi training today, where just as in OF the most likely (MAP) sequence is imputed instead of doing inference. While EM training is known to produce better recognisers on the same data, MAP training is still preferred for reasons of computational efficiency.

Our setup is as follows. We ran all methods on the same training data set, starting from the same initial code (drawn at random). For OF, learning rate and renormalisation heuristic parameters were left unchanged in their code. We used the values $0.1, 0.2, 0.4, 0.6962$ for τ, and $0.006, 0.01$ for σ^2. The values $\tau = 0.6962$, $\sigma^2 = 0.01$ come from the OF code, while $\sigma^2 = 0.006$ is closer to values ultimately preferred by the EP runs. All methods were run for 10000 batch updates, thus 20 sweeps over all images (in random ordering, different for each sweep). OF was run[30] separately for each of the eight (τ, σ^2) variants. On the other hand, for the EP runs, σ^2 was adjusted along with X as described above, and only τ was provided (the initial value for σ^2 was 0.002). The following

28. As discussed in Section 2.1, the mode $\hat{a}_j$ has many components which are exactly zero, which does not hold for the mean.
29. We could adjust τ in the same way with EP, but this is not done here. As noted in Section 3.5, the optimisation of σ^2 should behave better.
30. We also ran OF with $\tau = 0.05$, which gave bad results not reported here.

	$\tau = 0.1$		$\tau = 0.2$		$\tau = 0.4$		$\tau = 0.69$	
σ^2	OF	EP	OF	EP	OF	EP	OF	EP
0.006	-27.09	-80.04	-68.17	-80.04	-35.91	-80.05	92.73	-80.04
0.01	-2.329	-63.37	-53.53	-63.47	-43.69	-63.63	60.44	-63.80

Table 1: EP negative log marginal likelihood (EP average coding cost) per image, evaluated on the full test set (50000 cases), for different methods after 10000 batch updates of learning X. OF: Olshausen/Field; EP: our method.

learning rate schedule parameters[31] were used: $A = 0.79$, $B = 0.79 \cdot 10^3$, $A_l = 0.79 \cdot 10^{-5}$, $B_l = B$. This means that ξ decreases from 10^{-3} to $7.32 \cdot 10^{-5}$, and ξ_l from 10^{-8} to $7.32 \cdot 10^{-10}$.

We compare methods in general by evaluating the EP negative log marginal likelihood approximation on the test set, normalised by the number of images. This is equivalent to the EP approximation of the average coding cost per image (Lewicki and Olshausen, 1999) (smaller is better). For all but the final codes (after 10000 updates), we do this evaluation on the subset of size 1000. In order to evaluate test scores or to learn X (EP variants only), we need to perform EP inference on each image separately. To this end, we intended to use standard EP initially ($\eta = 1$, see Section 3.3.1), but ran into severe numerical problems on a significant number of images,[32] as described in Section 3.3.1. This led us to use fractional EP with $\eta = 0.9$ instead, which is the basis for all learning and test score evaluation results presented in this section.

learning curves along 10000 batch updates are shown in Figure 3 (using the test subset), and final EP negative log marginal likelihoods per image on the full test are given in Table 1. To recapitulate, the figures show average coding costs per image under the codes learned by the different methods, where τ and σ^2 are fixed for the evaluation. While OF was provided with τ, σ^2, EP only received τ during learning and had to adjust σ^2 alongside the code matrix X. We see from Figure 3 (upper left) that for the learning rate schedule used here, EP grows σ^2 smoothly from 0.002 to about 0.005.

Note that in the Bayesian viewpoint of image coding, all hyperparameters of a model work together in order to represent a data distribution (of image patches u) well, that is the code X, but also τ and the noise variance σ^2. In other words, code and noise variance are dependent. The EP runs settle at around $\sigma^2 = 0.005$, so the codes X found by them do better at $\sigma^2 = 0.006$ than at $\sigma^2 = 0.01$. The results for EP seem to not much depend on τ, but the situation is quite different for OF. At $\tau = 0.2$, the OF codes do well in comparison to the EP ones, and the lower $\sigma^2 = 0.006$ is preferred as well. For $\tau = 0.1$ or $\tau = 0.4$, they do significantly worse, and the preferred value is

31. These values were chosen after few initial runs, but not optimised over. Only for $\xi_l(0)$ did we compare runs, looking at learning curves on the training set. For $\xi_l(0) = 10^{-9}$, σ^2 hardly changed at all, while for $\xi_l(0) = 10^{-7}$, σ^2 increased sharply to above 0.02, then descended slowly towards 0.01.

32. None of these problems happened with fractional EP, $\eta = 0.9$. However, there is a pattern to these failures, indicating that further analysis would be valuable. In general, during learning X, EP convergence was harder to attain when the code was already optimised, with structural features emerging in the filters. While X could still be learned with $\tau = 1$, the test set log marginal likelihood evaluations for these codes could not be computed for many patches (using EP with $\eta = 1$). We evaluated these scores using EP with $\eta = 0.9$, finding very similar results (not shown here) than with the codes learned using $\eta = 0.9$. The reason for not simply abandoning standard EP for more robust fractional variants in general is based on arguments concerning alpha-divergences (Minka, 2004) (no hard theory is available to settle this issue, to our knowledge), apart from the somewhat more appealing motivation that can be given for standard EP (Opper and Winther, 2000).

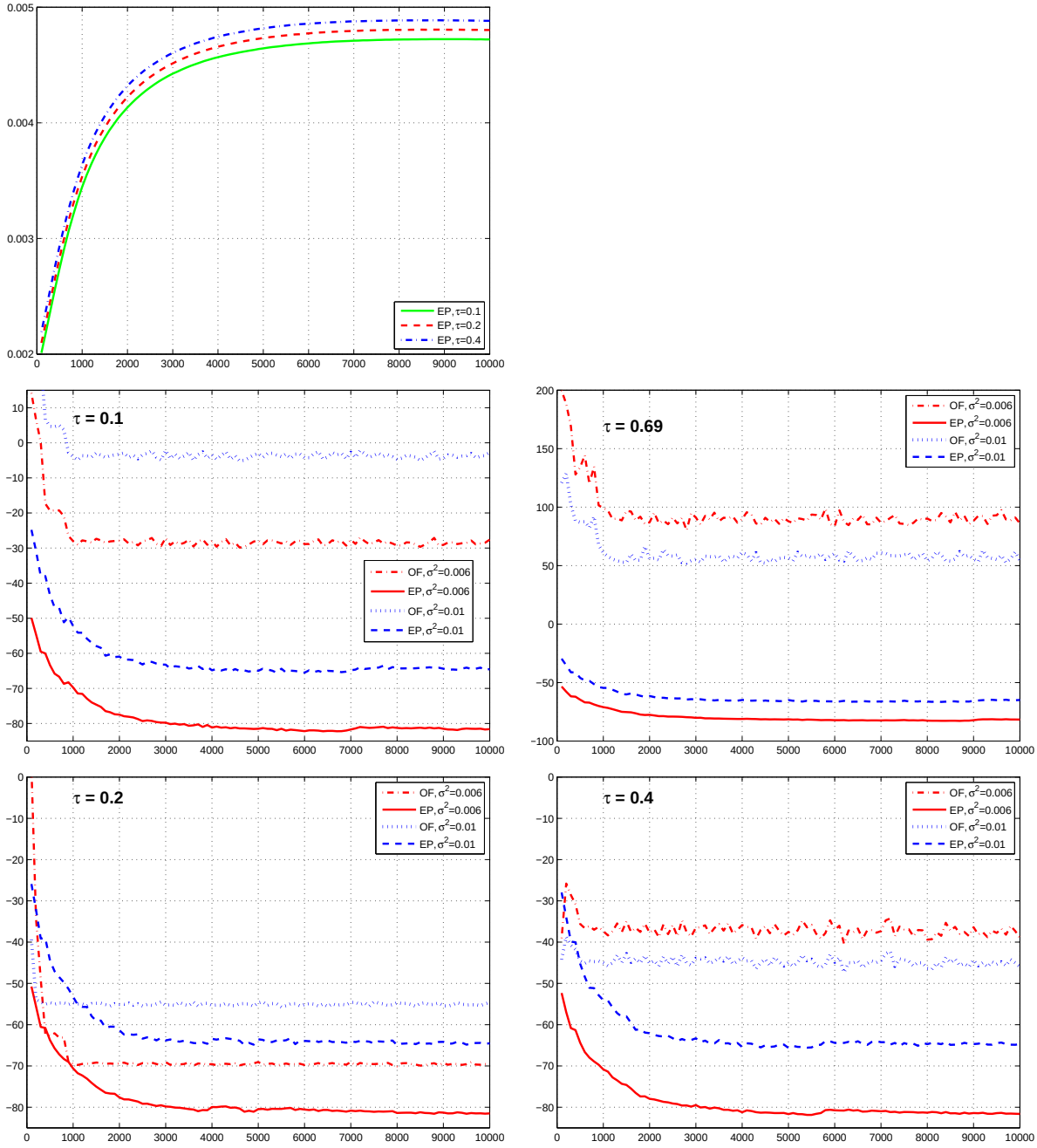


Figure 3: learning curves along 10000 batch updates. Upper left: Noise variance σ^2 for EP (different prior scales τ). Others: EP $-\log P(D)/r$ on test subset ($r = 1000$); $\tau = 0.1$ (middle left), $\tau = 0.6962$ (middle right), $\tau = 0.2$ (lower left), $\tau = 0.4$ (lower right).

$\sigma^2 = 0.01$ for $\tau = 0.4$. Finally, poor results[33] are obtained by OF with $\tau = 0.6962$, as well as with

33. We re-ran this case several times, in the way described in Appendix E, always obtaining the same poor results.

$\tau = 0.05$ (not shown here). The learning curve behaviour of the EP runs is much smoother than for OF, suggesting that the former optimisation problem is better behaved.

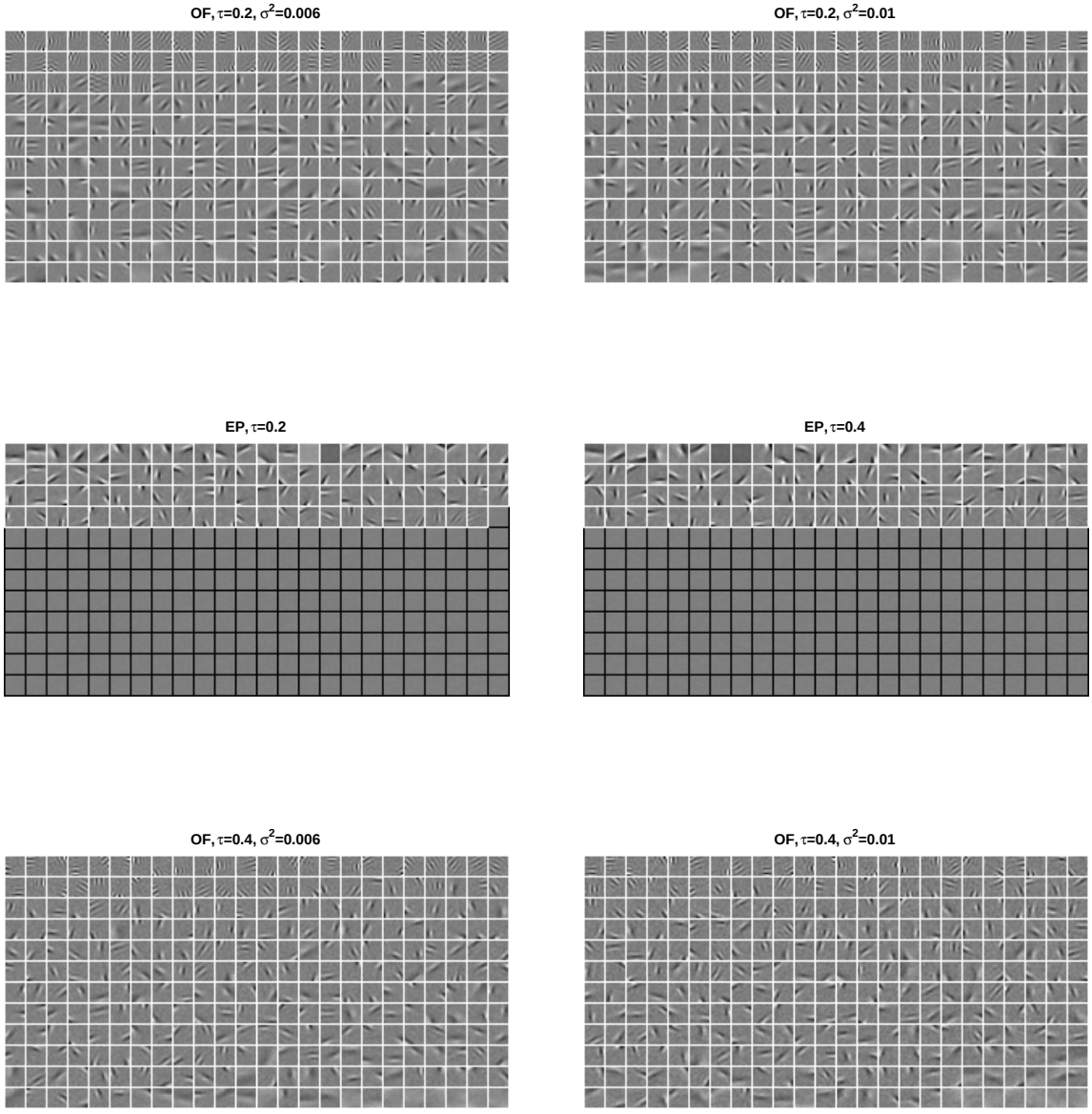


Figure 4: Final codes (after 10000 batch updates). Filters are ordered by descending $\|X_{\cdot,i}\|$, row-major ordering. Filters with $\|X_{\cdot,i}\| > (3\%) \max_j \|X_{\cdot,j}\|$ have white frame, black otherwise.

	$\tau = 0.1$		$\tau = 0.2$		$\tau = 0.4$		$\tau = 0.69$	
σ^2	EP[96]	EP[288]	EP[95]	EP[288]	EP[96]	EP[288]	EP[94]	EP[288]
0.006	-81.41	-80.04	-81.41	-80.04	-81.41	-80.05	-81.41	-80.04
0.01	-64.74	-63.37	-64.84	-63.47	-65.18	-63.63	-65.18	-63.80

Table 2: EP negative log marginal likelihood (EP average coding cost) per image, evaluated on the full test set (50000 cases), where X has been learned by EP. EP[288]: All X columns (copied from Table 1); EP[94–96]: Only X columns of significant size.

The final codes for different setups are given in Figure 4. The most distinctive difference between EP and OF codes is that for the latter, the filters do not differ much in size,[34] while there is a clear size signature in the codes found by EP: about 96 filters have significant sizes, while the remaining ones are about two orders of magnitude smaller. In the panels of Figure 4, filters with $\|X_{\cdot,i}\|$ larger than three percent of the maximum value are surrounded by white frames. For EP, these are 94–96 of 288 columns of X. In Table 2, we show EP average coding costs for the full test set, given that only the filters of significant size are used. These are even slightly lower than for the respective models using all columns of X.

We see that for the given task, codes attaining the lowest average cost are in fact *undercomplete*. A Bayesian method (such as EP here) removes unnecessary dimensions by default, through what has been called automatic relevance determination (see also Section 8.1). This does not happen for the OF method, which is not Bayesian and ignores covariances when learning X. We also note that in the codes found by OF, the filters of largest size are non-localised gratings, while all filters of significant size found by EP are localised and oriented. Both the smaller number of filters required and the strict localisation properties can be explained by noting that each image patch is explained probabilistically in EP, following Lewicki and Olshausen (1999), while in OF, this has to be done using a deterministic sparse encoding. In an update of X, each image only affects a small number of filters. It is then not too surprising that additional non-localised filters emerge in OF. If the hypothesis of Lewicki and Olshausen (1999) is taken for granted, these filters should be interpreted as artifacts of its improper implementation.

Note that the OF method runs much faster than EP. Finding $\hat{a}_j$ is a quadratic program (Tibshirani, 1996), which can be solved efficiently. Our EP code for the experiments here is "naive", in that all sites are visited in random ordering, no further efforts (such as the ones described in Section 6.3) are done. However, the arguments in Olshausen and Field (1997) and Lewicki and Olshausen (1999) do not call for a method which can be run very efficiently on a digital computer. A model is suggested which, in simple terms such as independence, linearity, and sparsity, could account for the formation of early visual neuron's receptive fields. The hypothesis of Lewicki and Olshausen (1999) is equivalent to a Bayesian perspective, where inference is a core requirement for improving the code, in much the same way as in EM for speech recognition, or graphical model learning in general. For both OF and EP, filters of significant size are localised, oriented gratings. However, our EP method more accurately implements the hypothesis of Lewicki and Olshausen (1999) than the algorithm of Olshausen and Field (1997), and leads to *qualitatively different* find-

34. Their renormalisation heuristic keeps them at similar, yet not at equal sizes.

ings: the data calls for a significantly undercomplete, therefore rather compact code, a fact that is not picked up by the OF method at all (Berkes et al., 2008, report similar findings with an approximate Bayesian method). Moreover, OF uses a significant number of non-localised filters, which are not present among the vectors of significant size found by EP. A more careful study based on our framework will shed light on what relevant properties in the codes can be explained by the probabilistic hypothesis of Lewicki and Olshausen (1999), versus which findings should rather be attributed to their particular computational method (maximum a posteriori, winner-takes-all X updates, variance renormalisation heuristic, etc.).

Apart from learning codes with EP, we can also use the log marginal likelihood approximation of EP in order to compare codes obtained by other methods. In Bayesian terms, such a comparison is done by computing Bayes factors, which is comparable to hypothesis testing. Moreover, for a fixed code X and data $\{u_j\}$, the noise variance σ^2 can be optimised by EP, in what is suggested to be a robust process in Section 3.5.

7.3 Compressive Sensing

In this section, we present results for a compressive sensing toy example. The motivation behind this application was given in Section 2.4. Results from a larger set of experiments, including some large-scale applications (see Section 6), are given in a later paper (Seeger and Nickisch, 2008). The experiments have been done by Hannes Nickisch.

In our toy experiment, the signal $y \in \mathbb{R}^n$ is sparse itself, so the coding matrix Φ is the identity. We have $n = 512$. Measurements are taken as $u = Py + \varepsilon$, where P (or X here) is the measurement matrix, and ε is Gaussian noise with standard deviation $\sigma = 0.005$. We compare methods where the measurement projections P are optimised in a sequential row-by-row manner, with methods where P is drawn uniformly at random on the unit hypersphere. In any case, the projections (i.e., rows of P) are constrained to have unit norm. The signal y is created by drawing $k = 20$ non-zero positions at random. The non-zero y_i are drawn at random from $\{-1,+1\}$ (uniform spikes), or according to a density[35] with support $\mathbb{R} \setminus (-0.21, 0.21)$ (non-uniform spikes). Examples for such signals are shown in Figure 5.

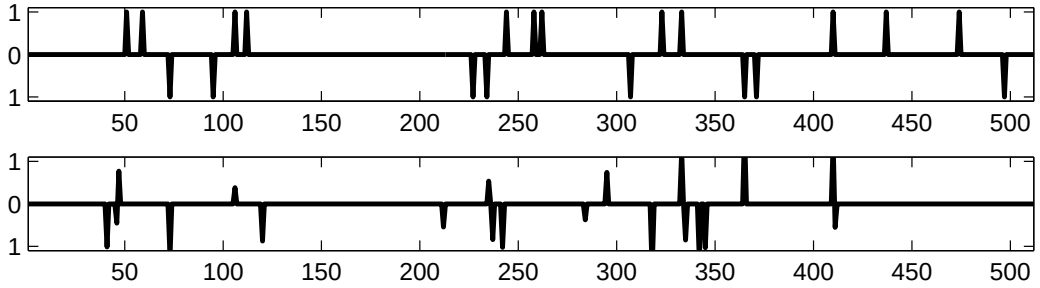


Figure 5: Examples for signal y. Top: uniform spikes. Bottom: non-uniform spikes.

The sequential experimental design of rows of P is a special case of the standard design setup of Section 4. Namely, among all p_j of unit norm, select the one which leads to minimum expected entropy $\mathrm{E}[\mathrm{H}[Q']]$, where Q' is the posterior after inclusion of p_j, and the expectation is w.r.t. $Q(u_j)$,

35. Namely, $y_i = \alpha(r + 0.25\,\mathrm{sgn}\, r)$, $r \sim N(0,1)$, where $\alpha = (5/4 + \sqrt{2/\pi} - 2/\pi)^{-1/2} \approx 0.84$, so that $\mathrm{Var}[y_i] = 1$.

u_j being the new measurement. Note that $\mathrm{H}[Q']$ is in fact independent of u_j, since Q' is Gaussian. Moreover, one can show that this criterion is equivalent to the expected information gain in this case (MacKay, 1991). A simple argument shows that the eigenvector for the largest eigenvalue of Σ solves this problem, where $Q = N(h, \sigma^2\Sigma)$ is the current posterior. This eigenvector can be found by the power method.

We compare the following methods. Our design approach based on EP is called *EP opt*. The method suggested by Ji and Carin (2007) is called *RVM opt* (P designed) or *RVM rand* (P random). They select P in the same way as we do, but making use of their approximate posterior, which they obtain as a variant of sparse Bayesian learning (SBL) (Tipping, 2001) (RVM refers to the most commonly used variant of SBL). The method most frequently used in compressive sensing applications so far is basis pursuit (Chen et al., 1999), where y is estimated by minimising $\|y\|_1 = \sum_i |y_i|$, subject to $Xy = u$. Note that this corresponds to MAP estimation in the sparse linear model if $\sigma^2 \to 0$ (noiseless case). This can be formulated as a linear program. *L1* and *BP* here use two different implementations.[36] For all methods, the first 40 rows of P are drawn at random. If $\hat{y}$ denotes the best prediction of y from the measurements u (the mean of Q for our method and the RVM variants), the error is measured as $\|\hat{y} - y\|/\|y\|$, where $\|\cdot\|$ is the Euclidean norm. Results are shown in Figure 6.

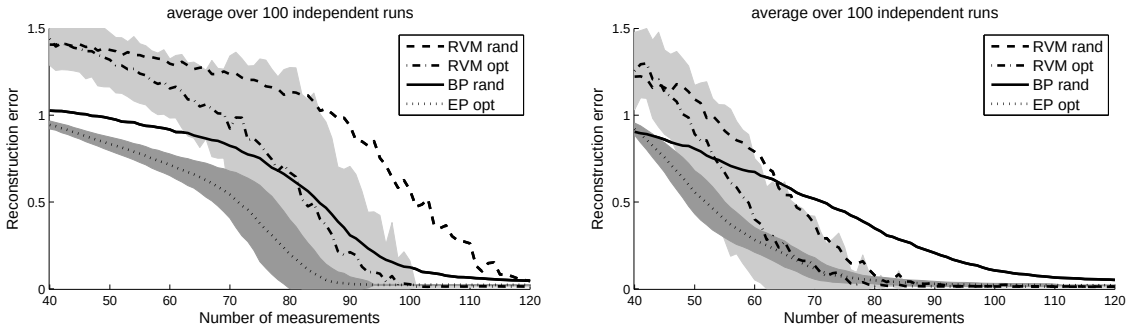


Figure 6: Results for compressive sensing toy example comparison. Left: uniform spikes. Right: non-uniform spikes. Averaged over 100 runs, shown are means and standard deviations (latter only for EP and RVM). See text for details.

Further experiments are required in order to draw definite conclusions, such are in preparation. We note that our EP method outperforms all others, and that random methods in general perform worse than the ones using experimental design. Moreover, our method clearly performs much better than the method of Ji and Carin (2007), while theirs is somewhat faster. Notably, our method also performs more robustly across runs than theirs. Moreover, the methods trying to approximate Bayesian inference in general perform better than basis pursuit on this task. The latter is certainly significantly faster than any of the other methods here, but its suboptimal performance on the same fixed data, and more importantly the lack of an experimental design framework, clearly motivates considering approximate Bayesian inference for compressive sensing as well.

36. *L1* is `l1-magic` from `www.acm.caltech.edu/l1magic/`, and *BP* is from SparseLab `sparselab.stanford.edu/`.

8. Related Approximate Inference Methods

In this section, we put EP for the sparse linear model into perspective by directly comparing it to other proposed methods of approximate inference. We rely on Palmer et al. (2006), who provide a somewhat more general discussion, but EP is not mentioned there. Before we start, we remind the reader that by "approximate inference" method, we mean a technique which delivers a useful approximation of marginal (or even joint) posteriors, and ideally can also be used in order to approximate the marginal likelihood. This is important in the context of the sparse linear model, where many methods rather try to find a maximally sparse solution of the noisy system (1), without addressing the former points. A study, comparing the methods discussed here in terms of approximation quality of marginals and of the marginal likelihood, is subject to future work. Note that none of these methods has been applied to the experimental design problem we address here (to our knowledge), with the exception of SBL (Ji and Carin, 2007).

8.1 Sparse Bayesian Learning

The idea of automatic relevance determination (ARD) has been proposed by Neal (1996). It is a variant of empirical Bayesian marginal likelihood maximisation (see Section 5). In the context of the sparse linear model, only a few components of a are typically relevant for describing the data, all others could be set to zero. ARD works by placing a prior $N(a_i|0,\sigma^2\pi_i^{-1})$ on a_i, where π_i is a scale parameter, then maximising the marginal likelihood $P(u,\pi)$ w.r.t. π. Here, π_i can be given a heavy-tailed hyperprior. The Occam's razor effect embedded in empirical Bayes (MacKay, 1992) leads to π_i becoming large for irrelevant components a_i: a model with few relevant components is simpler than one with many, and if both describe the data well, the former is preferred under ARD.

ARD has been applied to the sparse linear model by Tipping (2001), where the method was called *sparse Bayesian learning* (SBL). The derivation there makes use of *scale mixture* decompositions (Gneiting, 1997; Palmer et al., 2006) for the non-Gaussian prior sites. Namely, many univariate symmetric distributions can be represented in the form $P(a_i) = \mathrm{E}[N(a_i|0,\sigma^2\pi_i^{-1})]$, with some distribution over π_i. Tipping uses Student's t sparsity priors $P(a_i)$, for which π_i has a Gamma distribution. However, a direct comparison with the sparse linear model used here requires Laplace priors.

The Laplace density has the following scale mixture decomposition (Park and Casella, 2005; Gneiting, 1997):

$$\frac{\tilde{\tau}}{2}e^{-\tilde{\tau}|a_i|} = \mathrm{E}[N(a_i|0,\sigma^2\pi_i^{-1})], \quad \pi_i \sim \lambda\pi_i^{-2}e^{-\lambda/\pi_i}\mathrm{I}_{\{\pi_i>0\}} = \mathrm{IG}(1,\lambda),\ \lambda = \frac{\tau^2}{2}. \tag{9}$$

Note that the scale distribution of π_i does not have mean or variance. With $\Pi = \mathrm{diag}\,\pi$, we have

$$P(u,\pi) = \int P^{(0)}(a)N(a|0,\sigma^2\Pi^{-1})\,da|\Pi|^{-2}\lambda^n e^{-\lambda 1^T(\pi^{-1})},$$

which has the same form as in our framework. Here, $b_i = 0$, and the π_i have a different interpretation as scale hyperparameters. The marginal likelihood $P(u)$ is obtained by integrating out π, which cannot be done tractably. Instead, a *maximum a posteriori* (MAP) approximation is done in SBL: we find a maximiser $\hat{\pi}$ of $P(u,\pi)$, then approximate $P(u) \approx P(u,\hat{\pi})$. This is a joint non-convex optimisation problem, so all we can hope for is a local maximum. Faul and Tipping (2002) propose

the simple sequential technique of maximising $P(u,\pi)$ one π_i at a time. This results in the following update rule, as is shown in Appendix D.1:

$$\pi_i' = \frac{\sqrt{9+4\tau^2\beta}-3}{2\beta}, \quad \beta = \rho_i + \sigma^{-2}h_i^2 = \sigma^{-2}\mathrm{E}_Q[a_i^2]. \tag{10}$$

Confusingly, this is in fact not the method used in the experiments of Tipping (2001). This point is clarified at the end of this section.

Comparing SBL to our EP method, we note that the former does not require quadrature, but merely simple analytical updates. The π_i remain positive, and the method is numerically stable. SBL can be implemented using the same representation as ours. While $b = 0$ here, this does not lead to simplifications in representations or updates. In fact, both methods can share much of the same code, they differ only in how $\pi_i \to \pi_i'$ is computed for each site i. The marginal likelihood approximation resulting from SBL is $P(u,\hat{\pi})$. Just as for EP, this is not a bound on $P(u)$ (see Section 5).

Note that a variant of SBL with the Laplace prior has been proposed by Figueiredo (2003). However, they were interested in the MAP solution $\mathrm{argmax}_a P(a|D)$ rather than in an approximation to the posterior, which allowed them to integrate out the π_i by EM. Note also that SBL for the linear model with Student's t prior has been applied to gene network identification by Rogers and Girolami (2005), although they did not consider experimental design.

While a direct comparison is subject to future work, we note that SBL is certainly simpler to implement for the sparse linear model. Some safeguards required to make EP run in a numerically robust way, are not needed with SBL. On the other hand, EP is of course more general, since SBL is limited to non-Gaussian sites with a scale mixture decomposition. For example, non-symmetrical distributions such as classification likelihoods cannot be used.

There is at least the following worrying fact about SBL as approximation to Bayesian inference. We have used the scale mixture decomposition of the Laplace density (9) in terms of π_i, but we could just as well have chosen the one based on $s_i = \pi_i^{-1}$, with an exponential distribution on s_i. Doing so, we obtain an entirely different method, which did much worse than the variant derived here in initial experiments and in fact fails badly as approximation to Bayesian inference, since predictive variances are orders of magnitude too small. Furthermore, this "variant" of SBL converges exceedingly slowly in the s_i, while the method given here runs quite fast. Nevertheless, *both* variants are motivated in the same way: scale mixture decomposition, followed by a MAP approximation. The problem is that the latter, much in contrast to exact Bayesian inference (or, in fact, to expectation propagation), is not invariant to reparameterisations. The one chosen by Tipping (2001) certainly works well, at least in terms of delivering sparse solutions, but with others, SBL can fail badly. This important ambiguity has been noted by Wipf et al. (2004).

Finally, we note that there is some confusion about what exactly constitutes SBL, started in part by somewhat unclear formulations in Tipping (2001). In the paper, a method for finding maximally sparse solutions to the noisy linear system (1) is proposed. While the motivation is clearly Bayesian, the fact that a Student's t sparsity prior is used, is mentioned only in order to explain the favourable results. In fact, the "prior" actually used for π_i is $\propto 1/\pi_i$, resulting in $P(a_i) \propto 1/|a_i|$. Both are not normalisable as distributions. Our interest is in approximate Bayesian inference, with an eye towards experimental design, so we cannot consider such uninformative priors. We take the freedom here to interpret SBL as introducing scale mixture parameters π_i, followed by a MAP approximation w.r.t. π, at the expense of actually not covering the algorithm used in the experiments of Tipping

(2001). Wipf et al. (2004) show that the latter algorithm can in fact be interpreted as an instance of direct site bounding (see Section 8.2), which at first sight has little to do with scale mixtures, but see Palmer et al. (2006).

We would not stress this point if there was little difference in practice between (what we refer to as) SBL and direct site bounding (or other variants of the theme). However, initial comparative experiments with the sparse linear model show very significant differences in approximation quality. All these techniques find local maxima of $P(u|\pi)f(\pi)$. Contrary to what seems to be widely believed among practitioners, the form of f really matters. From our experience, the quality of approximate inference as well as the speed of convergence of sequential optimisation depend strongly on f. Wipf et al. (2007) show that the capability of the method estimating the correct relevant subset also hinges dominantly on the choice of f. Beyond that, the dependence on f of the quality of the covariance estimate, centrally important for experimental design, has not been analysed at all to our knowledge.

8.2 Direct Site Bounding. Variational Mean Field Bayes

A direct approach for obtaining an easily computable lower bound on the log marginal likelihood $\log P(u)$ works by lower-bounding the sites $t_i(a_i)$ by terms of Gaussian form. A powerful way of obtaining global lower bounds of simple form is exploiting convexity (Jaakkola, 1997). We can apply this approach to the sparse linear model with Laplace prior, which results in a method proposed by Girolami (2001). The general idea in the context of non-Gaussian linear models is noted in Palmer et al. (2006).

For the Laplace (2), we have that $\log t_i(a_i) = -\tilde{\tau}\sqrt{a_i^2} + \log(\tilde{\tau}/2)$, which is convex in a_i^2. A global tight lower bound is obtained using Legendre-Fenchel duality (Boyd and Vandenberghe, 2002), resulting in

$$e^{-\tilde{\tau}|a_i|} = \sup_{\pi_i>0} N^U(a_i|0,\sigma^{-2}\pi_i)e^{-(\tau^2/2)\pi_i^{-1}}.$$

We can plug in the r.h.s. for $t_i(a_i)$, then integrate out a in order to obtain a lower bound on $\log P(u)$. The outcome is quite similar to SBL, where t_i is replaced by the same term times $(2\pi)^{-1/2}\tau\pi_i^{-3/2}$. Since the ratio does not depend on a, we have that

$$P(u) \geq P_{Giro}(u;\pi) = (2\pi)^{-n/2}\tau^n|\Pi|^{-3/2}P_{SBL}(u,\pi),$$

Following Appendix D.1, it is clear that the update of π_i, keeping all others fixed, results in a quadratic equation with the positive solution

$$\pi_i' = \frac{\tau}{\sqrt{\beta}}, \quad \beta = \rho_i + \sigma^{-2}h_i^2 = \sigma^{-2}\mathrm{E}_Q[a_i^2].$$

While SBL does not render a bound on $\log P(u)$, Girolami's method does so by construction. Note that SBL and direct site bounding lead to quite similar replacements for t_i, if applied to the linear model with Laplace prior. The same is true if a Student's t prior is used, as has been observed by Wipf et al. (2004). Somewhat ironically, the modification in the latter case is precisely the result of the "uninformative limit" taken in Tipping (2001), which also seems to work best in practice. This point is discussed at the end of Section 8.1. Palmer et al. (2006) give the precise relationship between SBL and direct site bounding (called "integral case" and "convex case" there), showing that if t_i admits a scale mixture decomposition, it can also be bounded via Legendre duality.

Note that for the same (β, τ), π'_i is smaller for the SBL update than for the direct site bounding one. Namely,

$$\pi'_{SBL,i} = \pi'_{Giro,i} \times \left(\sqrt{1+\alpha} - \sqrt{\alpha}\right), \quad \alpha = 9/(4\tau^2\beta).$$

The ratio is smallest for small β, so Girolami's method chooses much larger π_i for the components which are "switched off", in that $\mathrm{E}_Q[a_i^2/\sigma^2] \approx 0$. It is thus more aggressively aiming for sparse solutions. In initial comparative experiments, Girolami's method outperformed SBL on the sparse linear model significantly in terms of the quality of inference approximation. Especially, the SBL marginal likelihood approximation turned out to be poor. A larger comparative study, from which conclusions can be drawn, is subject to future work.

A comparison between approximate inference techniques would be incomplete without including *variational mean field Bayes* (VMFB) (Attias, 2000; Ghahramani and Beal, 2001), maybe the most well known variational technique in the moment. It is also simply known as "variational Bayes" (see `www.variational-bayes.org`), although we understand this term as encompassing other variational methods for Bayesian inference as well, such as EP, SBL, direct site bounding, and others more. The distinctive feature of VMFB, previously known as "structured mean field", is the use of the generic mean field lower bound, as reviewed in Appendix D.2. VMFB for the sparse linear model is equivalent to direct site bounding, as has been shown in Palmer et al. (2006), and as is discussed in more detail in Appendix D.2. This equivalence holds as well for linear models with many other symmetric priors, for example Student's t.

8.3 Markov Chain Monte Carlo

While variational approximations are fairly established in machine learning, the dominant methods for approximating Bayesian inference in statistics are Markov chain Monte Carlo (MCMC) simulations (Neal, 1993; Gilks et al., 1996). In these techniques, a Markov chain over latent variables of interest (and possibly additional auxiliary ones) is simulated, whose stationary distribution is the desired posterior.

A simple MCMC method for the sparse linear model with Laplace prior has been proposed by Park and Casella (2005). They employ the scale mixture representation (9), introducing the scale parameters π as auxiliary variables alongside a. Their method is an instance of block Gibbs sampling, in that a is resampled given π, and vice versa. For simplicity, we denote the true posterior $P(\ldots|D)$ by Q in this section only. Now, the full conditional distribution $Q(a|\pi)$ is simply $N(a|h, \sigma^2\Sigma)$, with h, Σ defined as usual in terms of π (as in SBL above, $b = 0$ here), a Gaussian we can sample from easily (see Appendix B.2).

Next, the π_i are independent under $Q(\pi|a)$, with

$$Q(\pi_i|a) \propto \pi_i^{-3/2} \exp\left(\frac{-a_i^2\sigma^{-2}(\sqrt{2\lambda\sigma^2}/|a_i| - \pi_i)^2}{2\pi_i}\right) \propto \pi_i^{-3/2} \exp\left(\frac{-\tilde{\lambda}(\pi_i - \tilde{\mu})^2}{2\tilde{\mu}^2\pi_i}\right),$$

with $\tilde{\mu} = \sqrt{2\lambda\sigma^2}/|a_i|$, $\tilde{\lambda} = 2\lambda = \tau^2$. This density is the inverse Gaussian, which can be sampled from easily (see Chhikara and Folks, 1989, Section 4.5). The normalisation constant is $(\lambda/\pi)^{1/2}$.

Note that, just as with SBL and direct site bounding, we can use our existing EP code in order to implement this method as well. The π_i are resampled, instead of being updated deterministically. While they could be updated sequentially, Park and Casella (2005) consider joint updates which

tend to have better mixing properties. The representation, now maintaining the true $Q(a|\pi)$, has to be recomputed from scratch after each π update, so that each step costs $O(n \min\{m,n\}^2)$.

This sampler is certainly very simple to implement, especially with our representation code in place. Park and Casella (2005) give some arguments about the favourable role of log-concavity of $Q(a)$ for the sampler.[37] These are empirical, and even if good theoretical properties of MCMC samplers for log-concave posteriors have been established (Lovász and Vempala, 2003), these are different from the method considered here. Initial experiments with the sampler gave good results, although some erratic jumps in π components can be observed. The main cause of failure of block Gibbs samplers is the presence of strong dependencies between a and π. A more definite statement would require a comparison between this method and another sampler not based on scale variables.

The main advantage of MCMC over variational approximations is that it has no approximation bias in principle, if the chain is run for an unbounded amount of steps. In contrast, variational methods such as EP do have such a bias, which cannot be diminished by simply running them for longer.[38] It is also the case that simple variants of MCMC are typically fairly easy to implement, for example there are hardly ever problems with numerical stability. A main drawback of MCMC applied to problems of the sort considered here is that significantly more running time is required in order to obtain solutions of similar accuracy. Another major disadvantage is that a lot of expertise is required in order to run MCMC in a proper way. There are no convergence diagnostics which are easy to use or, in fact, are generally widely accepted. Most machine learning applications, such as the ones considered here, require methods which can be run robustly by users without extensive training in diagnosing Markov chain convergence. This problem becomes severe in the context of experiment design, where new decisions have to be done continuously, and even an expert would be hard pressed trying to diagnose proper convergence for all MCMC runs in between. Another drawback of MCMC is that while samples of the posterior are obtained, these cannot be used in a simple way in order to obtain a good estimate of the log marginal likelihood $\log P(u)$ (see Section 5). While the method of Chib (1995) proposes just that, it failed catastrophically in toy experiments of rather small scale with the sampler considered here, even if an excessive number of steps was used. This failure is interesting, given that the posterior is a log-concave (unimodal) distribution.

9. Discussion

We have shown how to perform accurate approximate Bayesian inference in the linear model with Laplace prior efficiently, by means of expectation propagation, and how this can be used to address tasks such as optimal design and hyperparameter estimation. The importance of numerical stability is raised for EP, and several means of improving robustness are proposed. Some implications of log-concavity for EP, and for approximate inference in general, have been shown.

The optimal design capability has been demonstrated for the application of gene regulatory network identification, where the sparsity prior was found to be essential in order to realise very significant gains. It is also motivated by preliminary experiments in the area of compressive sensing. Marginal likelihood optimisation has been used in order to optimise sparse codes for natural images,

37. They sample jointly over (a,σ^2), noting that $Q(a,\sigma^2)$ is log-concave in the transformation of (a,σ^2) described in Section 3.5.

38. Many variational methods allow for the choice of approximation families of varying complexity. For example, EP can be run with exponential families beyond the Gaussian, and even the case of Gaussian Q can potentially be improved by considering joint updates of blocks of sites. This requires the computation of multivariate non-Gaussian integrals, which is hard to do accurately, and is not done here.

in what constitutes an application of approximate inference on a large scale. Our experiments have been driven by a robust, efficient, and general implementation, which will be made available for scientific use.

9.1 Related Work

The sparse linear model (1) is of high practical relevance in statistics and machine learning, and has received a lot of attention. Some approximate inference techniques related to ours have been reviewed in Section 8. It is noted there that the computational representations and their robust update rules developed here, are required for these just as well.

The idea of L_1 regularisation of least squares has been used in very many contexts. The maximum a posteriori (MAP) treatment of the sparse linear model has been proposed as *Lasso* (Tibshirani, 1996) and as *basis pursuit* (Chen et al., 1999) (the latter for $\sigma^2 \to 0$). While the Lasso results in a quadratic program, basis pursuit is a linear programming problem. The prime advantage of an MAP treatment is that fitting to fixed data can be done very efficiently, in fact significantly faster than running EP until convergence. Very recently, several strong properties of the Lasso, basis pursuit, or other convex programming formulations of sparse estimation have been established, showing that in certain regimes they perfectly reconstruct very sparse signals in a minimax sense (Donoho and Elad, 2003; Candès et al., 2006; Wainwright, 2006). On the other hand, MAP as an approximation to Bayesian inference is fairly poor in this case. As noted in Section 3, a direct Laplace approximation is not well-defined for the sparse linear model. Even if this obvious problem was not present, the fact that there are many more variables than observations, renders the usual justification for Laplace's method obsolete. We have demonstrated a few advantages of going the full Bayesian way properly in this paper, such as optimal design based on uncertainty estimates, or marginal likelihood hyperparameter estimation. The MAP approximation for the sparse linear model has been applied to the gene network identification problem by Peeters and Westra (2004), but they did not address the problem of optimal design.

A general framework for EP on a class of hybrid models has been proposed by Zoeter and Heskes (2005). EP updates are done generically using Gaussian quadrature. Based on our findings here, EP for the sparse linear model with Laplace prior is very sensitive to the accuracy of EP updates, and the Gauss-Hermite rule would not lead to a working solution here. The generic proposal of converting between natural and moment parameterisation stated there is known to be unstable even in purely Gaussian models such as the Kalman filter, while our representation updates are essentially stable for log-concave sites. Also, the generality is quite restricted, in that they assume a fully factorised distribution family $\mathcal{F}$, which would not include joint Gaussians Q we consider here. Thus, while the prospect of a generic EP implementation is intriguing, important special cases such as Laplace or other sparsity prior sites, or joint Gaussian factors, would have to be treated as special cases. It remains to be seen whether the techniques to improve EP's numerical properties proposed here, are useful in this more general context as well.

Technically, our framework is quite related to the Independent Component Analysis method of Hojen-Sorensen et al. (2002), using Adaptive TAP (Opper and Winther, 2000) in order to estimate mean and covariance of the sources.[39] In fact, EP can be seen as particularly efficient way of searching for an ADATAP fixed point. They address the sparse image coding problem with the sparse linear model, but do not consider optimal design applications. Our approach is different to

39. What is meant is the *posterior* covariance of the sources, since in ICA, they are assumed to be independent *a priori*.

theirs in several important points. Their paper approaches a larger range of problems. On the other hand, they do not employ the natural EP marginal likelihood approximation we use here, but rather a variational bound. The study of Kuss and Rasmussen (2005) has indicated the superior quality of the EP approximation in a different, but related situation. Second, their image coding experiments are fairly small in scale, and they do not report any of the numerical problems we encountered, or in fact propose special measures to deal with such. Their paper treats numerically benign cases such as classification with logistic or probit likelihood alongside challenging (Laplace) or (in our opinion) highly problematic ones (Student's t; exponential power with exponent < 1), essentially recommending the same generic computations (which do not work, to the best of our knowledge and experience, in the situations we were interested in here).

9.2 Future Work

We have commented in Section 2.3 on the application of our method to the problem of learning and analysing image codes (Olshausen and Field, 1997; Lewicki and Olshausen, 1999), with the aim of understanding properties of visual neurons in the brain. In this context, the sparse linear model has been proposed as a useful setup, in which codes can be learned by maximising the marginal likelihood. The marginal likelihood approximation of Section 5 is more accurate than the one used by Lewicki and Olshausen (1999), and it will be interesting to test their hypothesis using our framework. A study with similar aims is given in Berkes et al. (2008), using variational mean field Bayes to approximate inference.

Other interesting applications lie in the area of compressive sensing. Some potential ones have been motivated in Section 2.4, and results will be reported in a later paper (Seeger and Nickisch, 2008). In this context, the large scale techniques motivated in Section 6 will be explored. Our preliminary findings in Section 7.3 indicate that approximate Bayesian inference and experimental design hold significant promises for compressive sensing, where so far approaches based on L_1-penalised estimation and random designs seem to predominate.

As detailed in Section 8, our EP framework is closely related to several other established methods of approximate inference. We plan to do a large, comparative study on several different tasks and data sets, where ground truth computations will be done via computationally intensive MCMC. We are not aware of existing comparative studies encompassing several approximate inference techniques for the sparse linear model.

Our experiences with the sparse linear model on the image coding problem (or with very underdetermined gene network identification settings) suggest that in some relevant cases, numerical stability issues seem to be inherently present in EP (i.e., are not just due to a bad implementation). These need to be understood much better, before we can seriously talk about generic EP solutions (Zoeter and Heskes, 2005), comparable to BUGS (Spiegelhalter et al., 1995) for Gibbs sampling or VIBES (Bishop and Winn, 2003) for variational mean field Bayes, both of which do not pose big problems of numerical stability. The sparse linear model seems a good test bed for such studies, different to the Gaussian process classification problem, which is numerically rather harmless. Since log-concavity helps in the important special case of fully Gaussian posterior approximations, its role needs to be understood better. Again, the Laplace prior of the sparse linear model will be important there, being "just about log-concave". Also, "cut-off" sites enforcing non-negativity, or more generally linear constraints (see Section 2.3), will play an important role there, not even being supported on all of $\mathbb{R}$.

In the MCMC approach of Park and Casella (2005), the noise variance σ^2 is integrated out along with the parameters a. This is possible (approximately) with EP as well, by choosing Q from an exponential family over (a, σ^2), which is not purely Gaussian. We have already done initial experiments with this extension, which will be reported in a later paper. In comparison to the method given here, the extension treats σ^2 as nuisance variable in a proper Bayesian fashion. It does not have to be chosen by other means, such as marginal likelihood maximisation. Much of the treatment of a, such as the representation of $Q(a|\sigma^2)$, or the analytical EP update w.r.t. a_i, is inherited from the framework given here. As an extension of EP beyond the case of fully Gaussian approximations, the extension is important as test bed for theoretical analyses. On the other hand, the extension is somewhat more complicated to implement, furthermore some of the numerical robustness of our method here is lost. For example, Theorem 1 does not hold for non-Gaussian Q. Moreover, the integration over σ^2 required by the EP updates cannot be done analytically, and approximate Gauss-Laguerre quadrature has to be used.

The Bayesian sparse linear model may have many other applications, given that its MAP variants (Lasso, basis pursuit) are very widely used. EP has also been applied to approximate inference in generalised linear models, where the likelihood is not Gaussian anymore, but comes from another exponential family. An application of this sparse generalised linear model to analysing neuronal spiking data is given in Seeger et al. (2007a) and Gerwinn et al. (2008), see also Qi et al. (2004). In this context, efficient online optimisation of experimental stimuli is an important task as well (Lewi et al., 2007).

The recent empirical success of EP in many different applications renders it important to gain a firm understanding of this technique. Some of the many relevant open questions are: For which models does the (single loop) EP algorithm provably converge? For which models is there no more than a single fixed point? How good is an EP approximation in terms of the marginals, and beyond that in terms of the covariance estimate? Numerical stability is an important issue for EP, which does not arise with most other approximate inference techniques. For which models can we expect numerical difficulties, and why? The step towards fractional EP may improve numerical properties of the method in general, but how do fractional EP approximations compare to the standard EP fixed points? Finally, how can EP fixed points be found for very large n, when the current practice of visiting each site in turn becomes unpractical?

Acknowledgments

The gene network identification application has been done in joint work with Florian Steinke and Koji Tsuda. The preliminary experiments with compressive sensing have been done in joint work with Hannes Nickisch. For the image coding application, we would like to acknowledge discussions with Matthias Bethge. We would like to thank Manfred Opper for interesting discussions about EP, and the anonymous referees for helpful comments. Supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778.

Appendix A. Details for the EP Update

In this section, we collect details concerning the EP update described in Section 3.3.

The Laplace site is $t_i(a) = \exp(-\tilde{\tau}|a|)$, $\tilde{\tau} = \tau/\sigma > 0$. Note that elsewhere, a prefactor of $\tilde{\tau}/2$ is used, which means that the value for Z_i to be computed here has to be multiplied post hoc.[40] We need to compute moments $I_k = \mathrm{E}_{N(h,\rho)}[a^k t_i(a)]$, $k = 0,1,2$, where we write $a = a_i$, $h = h_{\setminus i}$, $\rho = \rho_{\setminus i}$ for simplicity. W.l.o.g., we can assume that $\tilde{\tau} = 1$. Then, $I_0 = \tilde{I}_0(h) + \tilde{I}_0(-h)$, where some algebra gives

$$\tilde{I}_0(h) := \mathrm{E}\left[\mathrm{I}_{\{a\geq 0\}} e^{-a}\right] = \exp(\rho/2 - h)(1 - \Phi(\rho^{1/2} - h\rho^{-1/2})),$$

where Φ denotes the cumulative distribution function of $N(0,1)$. Now, from the definition of $\tilde{I}_0$, it is easy to see that $\tilde{I}_0(|h|) \geq \tilde{I}_0(-|h|)$, so that

$$\log I_0 = \log \tilde{I}_0(|h|) + \log\left(1 + \frac{\tilde{I}_0(-|h|)}{\tilde{I}_0(|h|)}\right)$$

can be computed in a stable manner. In the following, we make use of the well known asymptotic expansion

$$1 - \Phi(x) \sim N(x)x^{-1}\left(1 - 1x^{-2}\left(1 - 3x^{-2}\left(1 - 5x^{-2}\left(1 - 7x^{-2}(\ldots)\right)\right)\right)\right).$$

If $F(x) := \log(1 - \Phi(x))$, we use the asymptotic expansion up to $1 - 7x^{-2}$ for $x > 5$, while computing $F(x)$ exactly otherwise.[41] It is interesting to note that the simpler approximation $1 - \Phi(x) \approx N(x)/x$ is insufficient and leads to complete failure of EP on most tasks.

With this in mind, we have that $\log \tilde{I}_0(|h|) = \rho/2 - |h| + F(\rho^{1/2} - |h|\rho^{-1/2})$, and

$$R := \frac{\tilde{I}_0(-|h|)}{\tilde{I}_0(|h|)} = \exp\left(2|h| + F(\rho^{1/2} + |h|\rho^{-1/2}) - F(\rho^{1/2} - |h|\rho^{-1/2})\right).$$

For example, if $\rho^{1/2} - |h|\rho^{-1/2} > 5$, we use the tail approximation for both F terms. Namely, if the approximation is $1 - \Phi(x) \approx N(x)x^{-1}g(x)$, we end up with

$$R = \frac{(\rho - |h|)g(\rho^{1/2} + |h|\rho^{-1/2})}{(\rho + |h|)g(\rho^{1/2} - |h|\rho^{-1/2})}.$$

Note that in general, $R \in (0,1]$.

Next, $I_1 = \tilde{I}_1(h) - \tilde{I}_1(-h)$, with

$$\tilde{I}_1(h) := \mathrm{E}\left[\mathrm{I}_{\{a\geq 0\}} a e^{-a}\right] = (h - \rho)\tilde{I}_0(h) + \rho^{1/2}\exp(\rho/2 - h)\mathrm{E}\left[\mathrm{I}_{\{s\geq \rho^{1/2} - h\rho^{-1/2}\}} s\right],$$

where $s \sim N(0,1)$. Using $(dN(s))/ds = -sN(s)$, we have that $\mathrm{E}[\mathrm{I}_{\{s\geq s_0\}} s] = N(s_0)$. Furthermore,

$$\exp(\rho/2 \pm h)N(\rho^{1/2} \pm h\rho^{-1/2}) = N(h\rho^{-1/2}),$$

which does not depend on $\operatorname{sgn} h$. Therefore, the mean of $\hat{P}_i(a)$ is

$$\begin{aligned}\hat{h} = \frac{I_1}{I_0} &= \frac{(h - \rho)\tilde{I}_0(h) - (-h - \rho)\tilde{I}_0(-h)}{I_0} = h + \rho\frac{\tilde{I}_0(-h) - \tilde{I}_0(h)}{I_0} \\ &= h + \rho(\operatorname{sgn} h)\left(1 - 2(1 + R)^{-1}\right).\end{aligned}$$

40. The reason for dropping this prefactor is that we want to deal with the case of fractional sites (see Section 3.3.1) t_i^η by simply replacing τ by $\eta\tau$.

41. The C math library provides `log1p`$(x) = \log(1 + x)$, which is accurate for small $|x|$.

Since $I_1/I_0 = h + \rho\beta_i$, we have that

$$\beta_i = (\mathrm{sgn}\, h)\left(1 - 2(1+R)^{-1}\right). \tag{11}$$

Next, $I_2 = \tilde{I}_2(h) + \tilde{I}_2(-h)$, where some algebra gives

$$\begin{aligned}
\tilde{I}_2(h) &:= \mathrm{E}\left[\mathrm{I}_{\{a\geq 0\}} a^2 e^{-a}\right] \\
&= (\rho^2 - h^2)\tilde{I}_0(h) + 2h\tilde{I}_1(h) - 2\rho^{3/2} N(h\rho^{-1/2}) + \rho\exp(\rho/2 - h)\mathrm{E}\left[\mathrm{I}_{\{s\geq \rho^{1/2} - h\rho^{-1/2}\}} s^2\right] \\
&= (h-\rho)^2\tilde{I}_0(h) + 2\rho^{1/2} N(h\rho^{-1/2})(h-\rho) + \rho\exp(\rho/2 - h)\mathrm{E}\left[\mathrm{I}_{\{s\geq \rho^{1/2} - h\rho^{-1/2}\}} s^2\right].
\end{aligned}$$

Using $s^2 N(s) = N(s) - (dsN(s))/ds$, we see that

$$\rho\exp(\rho/2 - h)\mathrm{E}\left[\mathrm{I}_{\{s\geq \rho^{1/2} - h\rho^{-1/2}\}} s^2\right] = \rho\tilde{I}_0(h) + \rho(\rho^{1/2} - h\rho^{-1/2}) N(h\rho^{-1/2}).$$

Together, we have

$$\tilde{I}_2(h) = (h^2 + \rho^2 + \rho - 2h\rho)\tilde{I}_0(h) + \rho^{1/2} N(h\rho^{-1/2})(h-\rho),$$

thus

$$I_2 = (h^2 + \rho^2 + \rho) I_0 - 2\rho h(\tilde{I}_0(h) - \tilde{I}_0(-h)) - 2\rho^{3/2} N(h\rho^{-1/2}).$$

Using that $\beta_i = (\tilde{I}_0(-h) - \tilde{I}_0(h))/I_0$ and $\hat{h} = h + \rho\beta_i$, some algebra gives that

$$\frac{I_2}{I_0} = \rho + \rho^2 + h^2 + 2h(\hat{h} - h) - 2\rho^{3/2} N(h\rho^{-1/2}) I_0^{-1}.$$

Therefore, the variance of $\hat{P}_i(a)$ is

$$\begin{aligned}
\hat{\rho} &= \frac{I_2}{I_0} - \hat{h}^2 = -h^2 + (2h - \hat{h})\hat{h} + \rho + \rho^2 - 2\rho^{3/2} N(h\rho^{-1/2}) I_0^{-1} \\
&= \rho + \rho^2(1 - \beta_i^2) - 2\rho^{3/2} N(h\rho^{-1/2}) I_0^{-1},
\end{aligned}$$

since $(2h - \hat{h})\hat{h} = h^2 - (\rho\beta_i)^2$. Since $\hat{\rho} = \rho(1 - \rho\nu_i)$, we have that

$$\nu_i = \beta_i^2 - 1 + (\pi\rho/2)^{-1/2} \exp\left(-\frac{h^2}{2\rho} - \log I_0\right).$$

Finally, in order to incorporate $\tilde{\tau} \neq 1$, we note that this simply means plugging in $h = \tilde{\tau} h_{\setminus i}$, $\rho = \tilde{\tau}^2 \rho_{\setminus i}$ above, and multiplying β_i by $\tilde{\tau}$, ν_i by $\tilde{\tau}^2$. Note that $Z_i = I_0 = \mathrm{E}_{Q^{\setminus i}}[t_i(a_i)]$ is not required for the EP update itself, but has to be evaluated if an approximation to the marginal likelihood $P(D)$ is sought (see Section 5; recall that Z_i as computed here has to be multiplied with the prefactor $\tilde{\tau}/2$ of t_i which we omitted).

A.1 The Role of Log-concavity

In this section, we give the proof of Theorem 1. Recall the definition of log-concavity and the marginalisation theorem of Prékopa from Section 3.5. For an update at site i, we can assume that $Q^{\setminus i}(a_i)$ is a proper Gaussian. We begin by showing that $Z_i = \mathrm{E}_{Q^{\setminus i}}[t_i(a_i)]$ is log-concave in $h_{\setminus i}$. Namely, $\log Q^{\setminus i}$ is jointly concave in $(a_i, h_{\setminus i})$ (being a negative quadratic in $a_i - h_{\setminus i}$), so that $t_i(a_i)Q^{\setminus i}(a_i|h_{\setminus i})$ is log-concave in $(a_i, h_{\setminus i})$. Then, $Z_i(h_{\setminus i})$ is log-concave by the marginalisation theorem. Therefore, $\nu_i = -(\partial^2 \log Z_i)/(\partial h_{\setminus i}^2) \geq 0$ (see Section 3.3). The variance of $\hat{P}_i$ is $\sigma^2\rho_i'$, where $\rho_i' = \rho_{\setminus i}(1 - \sigma^2\nu_i\rho_{\setminus i})$. Since t_i is bounded with support of positive measure, this variance exists and is positive, implying that $1 - \sigma^2\nu_i\rho_{\setminus i} \in (0, 1]$. But $\pi_i' = \sigma^2\nu_i/(1 - \sigma^2\nu_i\rho_{\setminus i}) \geq \sigma^2\nu_i \geq 0$, so π_i' remains nonnegative throughout.

Appendix B. Details for Sequential Design

In this section, we collect details for the sequential design application of the sparse linear model.

B.1 The Simple Information Gain Score

The simple information gain is introduced in Section 4.1. Recall the Gaussian relative entropy from (4), and the fact that $M = I + x_* x_*^T \Sigma$. Thus, if $\alpha := 1 + x_*^T \Sigma x_*$, then $\log|M| = \log\alpha$, using the relation $|I + VW^T| = |I + W^T V|$. Furthermore, the Woodbury formula (Henderson and Searle, 1981) gives $M^{-1} = I - \alpha^{-1} x_* x_*^T \Sigma$, so that $\mathrm{tr}(M^{-1} - I) = \alpha^{-1} - 1$.

Finally, $\tilde{b}' = \tilde{b} + u_* x_*$, where $\tilde{b} = b^{(0)} + b$ (see Section 3.4), so that

$$h' = \left(\Sigma - \alpha^{-1}\Sigma x_* x_*^T \Sigma\right)\left(\tilde{b} + u_* x_*\right) = h + \alpha^{-1}(u_* - x_*^T h)\Sigma x_*,$$

and

$$(h' - h)^T \Sigma^{-1}(h' - h) = (\alpha - 1)\alpha^{-2}(u_* - x_*^T h)^2.$$

Altogether, the simple information gain score is

$$S(x_*, u_*) = \frac{1}{2}\left(\log\alpha + \frac{\alpha - 1}{\alpha}\left(-1 + \alpha^{-1}\left(\frac{u_* - x_*^T h}{\sigma}\right)^2\right)\right).$$

We need to compute α and $x_*^T h$. In the degenerate case, let $v = L^{-1} X \Pi^{-1} x_*$, then $\alpha = 1 + x_*^T \Pi^{-1} x_* - \|v\|^2$, and $x_*^T h = x_*^T \Pi^{-1}(b^{(0)} + b) - v^T\gamma$. In the non-degenerate case, let $v = L^{-1} x_*$, then $\alpha = 1 + \|v\|^2$, and $x_*^T h = v^T\gamma$.

The marginal criteria of Section 4.2 require the computation of $z_* = \Sigma x_*$. In the non-degenerate case, $z_* = L^{-T} v$. In the degenerate case, $z_* = \Pi^{-1}(x_* - X^T L^{-T} v)$.

B.2 Sampling A

We need to sample from $Q(A|D)$ in order to approximate the expected information gain, as noted in Section 4. Let $Q(a)$ be the posterior over a row of A, based on the representation given in Section 3.2, and let $n \sim N(0, I)$. In the non-degenerate case, $a = L^{-T}(\sigma n + \gamma)$ is distributed according to $N(h, \sigma^2\Sigma)$.

Sampling is more difficult in the degenerate case. Let

$$I + X\Pi^{-1}X^T = UDU^T$$

be the spectral decomposition, where D is diagonal and nonnegative, and $U \in \mathbb{R}^{m,m}$ is orthonormal. We make the ansatz

$$c = \left(I - \Pi^{-1} X^T U R U^T X\right) \Pi^{-1/2} n$$

with diagonal R. $\mathrm{E}[cc^T] = \Sigma$ gives $(D-I)R^2 - 2R + D^{-1} = 0$, which is solved by $R = \mathrm{diag}(1/(\sqrt{d_i}(\sqrt{d_i}+1)))_i$. Finally, $a = \sigma c + h$.

Appendix C. The Marginal Likelihood

In this section, we derive the EP marginal likelihood approximation and its gradient w.r.t. model parameters. Recall the discussion of Section 5, the definition of L is given in (6). First, $\log C_i = \eta^{-1}(\log Z_i - \log \tilde{Z}_i)$. The computation of $\log Z_i$ is discussed in Appendix A. Some algebra (Seeger, 2005) gives

$$\log \tilde{Z}_i = \frac{1}{2}\left(\log(1-\eta\pi_i\rho_i) - \frac{\eta\pi_i h_i^2 - 2h_i\eta b_i + \rho_i(\eta b_i)^2}{\sigma^2(1-\eta\pi_i\rho_i)}\right),$$

where $Q(a_i) = N(a_i|h_i, \sigma^2\rho_i)$.

We begin with $\nabla_X L$ and $\partial L/\partial\sigma^{-2}$ (both parameters of $P^{(0)}$), using (7). Since σ^2 also features explicitly in the sites t_i, the derivative is the sum of two parts, and we deal with the second part below.

$$d\log P^{(0)}(a) = \mathrm{tr}\left(\sigma^{-2} e a^T\right)^T (dX) + \frac{1}{2}\left(m\sigma^2 - \|e\|^2\right)(d\sigma^{-2}), \quad e := u - Xa.$$

If $Q(a) = N(h, \sigma^2\Sigma)$, then

$$\mathrm{E}_Q\left[d\log P^{(0)}(a)\right] = \mathrm{tr}\left(\sigma^{-2} f h^T - X\Sigma\right)^T (dX) - \frac{1}{2}\left(\|f\|^2 + \sigma^2\,\mathrm{tr}\, X\Sigma X^T - m\sigma^2\right)(d\sigma^{-2}),$$
$$f := \mathrm{E}_Q[e] = u - Xh.$$

Now, $\mathrm{tr}\, X\Sigma X^T = \mathrm{tr}(I - \Sigma\Pi) = n - (\mathrm{diag}\,\Sigma)^T\pi$, so that

$$dL = \mathrm{tr}\left(\sigma^{-2} f h^T - X\Sigma\right)^T (dX) - \frac{1}{2}\left(\|f\|^2 - \sigma^2(\mathrm{diag}\,\Sigma)^T\pi + (n-m)\sigma^2\right)(d\sigma^{-2}).$$

The derivative w.r.t. $\tilde{\tau} = \tau/\sigma$ is computed using (8). We have that $(d/d\tilde{\tau})\log t_i(a_i) = -|a_i| + 1/\tilde{\tau}$, so we need to compute

$$\mathrm{E}_{\hat{P}_i}[-|a_i|] = -Z_i^{-1}\mathrm{E}_{Q^{\setminus i}}[|a_i| t_i(a_i)],$$

which is of similar from to I_1 in Appendix A. In the notation used there, if $\hat{I}_1 = \tilde{I}_1(h) + \tilde{I}_1(-h)$, then $\hat{I}_1/I_0 = -\rho - \beta_i h + 2\rho^{1/2} N(h\rho^{-1/2}) I_0^{-1}$. Plugging in $h = \tilde{\tau} h_{\setminus i}$, $\rho = \tilde{\tau}^2\rho_{\setminus i}$, and dividing by $\tilde{\tau}$, we have that

$$-Z_i^{-1}\mathrm{E}_{Q^{\setminus i}}[|a_i| t_i(a_i)] = \tilde{\tau}\rho_{\setminus i} + \beta_i h_{\setminus i} - 2\rho_{\setminus i}^{1/2} N(h_{\setminus i}\rho_{\setminus i}^{-1/2}) I_0^{-1},$$

where β_i is given by (11) (it is not multiplied by $\tilde{\tau}$). Finally, $d\tilde{\tau} = \sigma^{-1}(d\tau) + \frac{1}{2}\tau\sigma(d\sigma^{-2})$, whereby we can complete the derivative w.r.t. σ^{-2} as well.

As an aside, there is a subtle issue concerning the derivative w.r.t. σ^2. Seeger (2005) shows that indirect dependencies on hyperparameters through the site parameters do not have to be taken into account when computing the gradient. But if the derivative of (6) w.r.t. σ^2 is computed, keeping

b_i, π_i constant, the result is different from ours here. This is explained by our non-standard parameterisation of site parameters in the present paper. Namely, what is referred to as site parameters in Seeger (2005), are in fact the $\sigma^{-2}b_i, \sigma^{-2}\pi_i$ here, *not* b_i, π_i. If the former are kept constant, a direct differentiation of (6) renders our result here.

Appendix D. Related Approximate Inference Techniques

In this section, we give details on the approximate inference techniques discussed in Section 8.

D.1 Sparse Bayesian Learning

Recall Section 8.1. In order to compute the marginal likelihood $P(u,\pi)$, we note that

$$P^{(0)}(a)N(a|0,\sigma^2\Pi^{-1}) = (2\pi\sigma^2)^{-m/2}e^{-(\sigma^{-2}/2)\|u\|^2}(2\pi\sigma^2)^{-n/2}|\Pi|^{1/2} N^U(a|\sigma^{-2}b^{(0)},\sigma^{-2}(X^TX+\Pi)),$$

so that with $h = \Sigma b^{(0)}$, $\Sigma^{-1} = X^TX + \Pi$, some algebra gives

$$2\log P(u,\pi) = \sigma^{-2}h^Tb^{(0)} + \log|\Sigma| - \tau^2 1^T(\pi^{-1}) + C,$$

where $C = -3\log|\Pi| - m\log(2\pi\sigma^2) - \sigma^{-2}\|u\|^2 + 2n\log(\tau^2/2)$.

We need to maximise $\log P(u,\pi)$ w.r.t. π_i, keeping all other π_j fixed. Then, $d\Pi = (d\pi_i)\delta_i\delta_i^T$, and let $\log P(u,\pi) = (1/2)\psi + C$. Furthermore, $Q(a_i) = N(a_i|h_i,\sigma^2\rho_i)$, that is, $\rho_i = \Sigma_{i,i}$. Now, $d\log|\Pi + X^TX| = \rho_i(d\pi_i)$, and $d\sigma^{-2}h^Tb^{(0)} = -\sigma^{-2}h_i^2(d\pi_i)$, so that

$$d\psi = \left(-\sigma^{-2}h_i^2 - 3\pi_i^{-1} - \rho_i + \frac{\tau^2}{\pi_i^2}\right)d\pi_i.$$

Equating this to zero results in a quadratic equation for π_i, whose nonnegative solution is given by (10).

D.2 Variational Mean Field Bayes

The *variational mean field Bayesian* (VMFB) framework is a fairly generic approach to variational inference. It starts from the classical variational characterisation of inference (Wainwright and Jordan, 2003):

$$\log P(u) = \sup_Q \mathrm{E}_Q\left[\log P(u,a,\sigma^2,\pi) - \log Q(a,\sigma^2,\pi)\right], \tag{12}$$

then relaxes the problem by imposing factorisation constraints on allowable Q (the optimal unconstrained choice for Q is the true posterior).[42] The variational characterisation is also known as mean field lower bound, because it is the defining feature of (structured) mean field approximations (Jordan et al., 1997).

Once appropriate factorisation assumptions are placed on Q, the feasible set can be written analytically in terms of factors from these families, and the right hand side of (12) and its gradient

42. Here, we introduce the scale parameters π_i by employing the scale mixture representation (9). VMFB works for models which can be represented exclusively in terms of exponential family distributions, which is often possible by introducing latent variables. One could possibly choose another representation of the Laplace sites, whence the equivalence of VMFB and direct site bounding would not hold, but this is not done here.

can be computed easily. Furthermore, this expression now lower bounds $\log P(u)$, because the maximisation is over the subset of factorising Q. On the other hand, the optimisation over factorising Q is not convex in general, and usually only a local optimum is found. Moreover, even the global maximum is the minimiser of $\mathrm{D}[Q \,\|\, P(\cdot|D)]$ over factorising Q (this is also the slack in the lower bound), so that Q does not in general have the same marginal moments as $P(\cdot|D)$. The latter would be obtained by minimising $\mathrm{D}[P(\cdot|D) \,\|\, Q]$ over factorising Q, but not even local minima of the latter can be found by any tractable method currently known.

For fixed σ^2, it has been shown in Palmer et al. (2006) that VMFB is strongly equivalent to Girolami's method of Section 8.2, in that the variational parameters, their updates, and the $\log P(u)$ lower bound are the same. We make the factorisation assumption $Q(a,\pi) = Q(a)Q(\pi)$. The resulting lower bound on $\log P(u)$ is optimised by updating the factors in turn, fixing the corresponding other one. If we fix $Q(\pi)$, the maximiser is $Q(a) = N(h, \sigma^2\Sigma)$, where h, Σ are defined as usual, but plugging in $\mathrm{E}_Q[\pi]$ for π. If $Q(a)$ is kept fixed, then the maximiser is

$$Q(\pi) \propto e^{\mathrm{E}_{Q(a,\sigma^2)}[\log P(\pi|a,\sigma^2,u)]} \propto P(\pi) e^{\mathrm{E}_{Q(a,\sigma^2)}[\log P(a|\sigma^2,\pi)]},$$

which decomposes w.r.t. the π_i. The form is given in Section 8.3, namely $\log Q(\pi_i) = C + \log \pi_i^{-3/2} - (\pi_i^2 \mathrm{E}[a_i^2\sigma^{-2}] + 2\lambda)/(2\pi_i)$, which is inverse Gaussian with mean $\tilde{\mu} = \tau/\sqrt{\mathrm{E}[a_i^2\sigma^{-2}]}$ and $\tilde{\lambda} = \tau^2$. A sequential VMFB variant iterates over the sites, updating $\pi_i' = \tau/\sqrt{\mathrm{E}[a_i^2\sigma^{-2}]}$. This is algorithmically equivalent the direct site bounding method of Section 8.2.

Appendix E. Modifications of Olshausen/Field Code

We compare our method against the one proposed by Olshausen and Field (1997), using their code which can be obtained at `http://redwood.berkeley.edu/bruno/sparsenet/`. Since the code is written for interactive use, we had to modify it in order to work for our study, which compares fully automatic methods.

First, our modification accepts a fixed training set of $r = 50000$ image patches, while the original code extracts patches on the fly.[43] A sweep over the whole set consists of 500 batch updates, where batches are drawn at random without replacement. 20 sweeps are done in total.

The code comes with several parameters. A study of the code reveals that `noise_var` is our σ^2, `beta` is our $\tilde{\tau} = \tau/\sigma$, and `sigma` is set to one. There is a learning rate parameter `eta`, which the documentation recommends to set by hand, starting with $\eta = 5$, reducing it towards $\eta = 1$ (the default value in the code). We chose the following schedule: $\eta = 5, 4, 3$ for 100, $\eta = 2$ for 500, then $\eta = 1$ for the remaining 9200 updates.

References

H. Attias. A variational Bayesian framework for graphical models. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 209–215. MIT Press, 2000.

43. We used their extraction code in order to create the data set in the first place.

P. Berkes, R. Turner, and M. Sahani. On sparsity and overcompleteness in image models. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, 2008.

C. Bishop and J. Winn. Structured variational distributions in VIBES. In C. Bishop and B. Frey, editors, *Workshop on Artificial Intelligence and Statistics 9*, pages 244–251, 2003. Electronic Proceedings (ISBN 0-9727358-0-1).

V. Bogachev. *Gaussian Measures*. Mathematical Surveys and Monographs. American Mathematical Society, 1998.

L. Bottou. Online learning and stochastic approximations. In D. Saad, editor, *On-Line Learning in Neural Networks*. Cambridge University Press, 1998.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2002.

E. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52 (2):489–509, 2006.

K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3): 273–304, 1995.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

R. Chhikara and L. Folks. *The Inverse Gaussian Distribution: Theory, Methodology, and Applications*. Marcel Dekker Inc., 1989.

S. Chib. Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90(432):1313–1321, 1995.

J. DeRisi, V. Iyer, and P. Brown. Exploring the matebolic and genetic control of gene expression on a genomic scale. *Science*, 282:699–705, 1997.

J. Dongarra, C. Moler, J. Bunch, and G. Stewart. *LINPACK User's Guide*. Society for Industrial and Applied Mathematics, 1979.

D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

D. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l_1 minimization. *Proc. Natl. Acad. Sci. USA*, 100:2197–2202, 2003.

A. Faul and M. Tipping. Analysis of sparse Bayesian learning. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 383–389. MIT Press, 2002.

V. Fedorov. *Theory of Optimal Experiments*. Academic Press, 1972.

M. Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1050–1059, 2003.

T. Gardner, C. Cantor, and J. Collins. Construction of a genetic toggle switch in Escherichia coli. *Nature*, 403(6767):339–342, 2000.

S. Gerwinn, J. Macke, M. Seeger, and M. Bethge. Bayesian inference for spiking neuron models with a sparsity prior. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, 2008.

Z. Ghahramani and M. Beal. Propagation algorithms for variational Bayesian learning. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 507–513. MIT Press, 2001.

W. Gilks, S. Richardson, and D. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1st edition, 1996.

W. R. Gilks and P. Wild. Adaptive rejection sampling for Gibbs sampling. *Applied Statistics*, 41(2): 337–348, 1992.

M. Girolami. A variational method for learning sparse and overcomplete representations. *Neural Computation*, 13:2517–2532, 2001.

T. Gneiting. Normal scale mixtures and dual probability densities. *J. Statist. Comput. Simul.*, 59: 375–384, 1997.

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.

H. Henderson and S. Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23:53–60, 1981.

P. Hojen-Sorensen, O. Winther, and L. Hansen. Mean field approaches to independent component analysis. *Neural Computation*, 14:889–918, 2002.

R. Horn and C. Johnson. *Matrix Analysis*. Cambridge University Press, 1st edition, 1985.

H. Ishwaran and J. Rao. Spike and slab gene selection for multigroup microarray data. *Journal of the American Statistical Association*, 100(471):764–780, 2005.

T. Jaakkola. *Variational Methods for Inference and Estimation in Graphical Models*. PhD thesis, Massachusetts Institute of Technology, 1997.

S. Ji and L. Carin. Bayesian compressive sensing and projection optimization. In Z. Ghahramani, editor, *International Conference on Machine Learning 24*. Omni Press, 2007.

M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods in graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1997.

B. Kholodenko, A. Kiyatkin, F. Bruggeman, E. Sontag, H. Westerhoff, and J. Hoek. Untangling the wires: A strategy to trace functional interactions in signaling and gene networks. *PNAS*, 99(20): 12841–12846, 2002.

H. Kushner and A. Budhiraja. A nonlinear filtering algorithm based on an approximation of the conditional distribution. *IEEE Transactions on Automatic Control*, 45:580–585, 2000.

M. Kuss and C. Rasmussen. Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 6:1679–1704, 2005.

N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 609–616. MIT Press, 2003.

J. Lewi, R. Butera, and L. Paninski. Real-time adaptive information-theoretic optimization of neurophysiological experiments. In B. Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

M. Lewicki and B. Olshausen. Probabilistic framework for the adaption and comparison of image codes. *J. Opt. Soc. Amer. A*, 16(7):1587–1601, 1999.

L. Lovász and S. Vempala. Hit and run is fast and fun. Technical Report MSR-TR-2003-05, Microsoft Research, Redmond, January 2003.

D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):589–603, 1991.

D. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

T. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, January 2001a.

T. Minka. Expectation propagation for approximate Bayesian inference. In J. Breese and D. Koller, editors, *Uncertainty in Artificial Intelligence 17*. Morgan Kaufmann, 2001b.

T. Minka. Power EP. Technical report, Microsoft Research, Cambridge, 2004.

R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993. See `www.cs.toronto.edu/~radford`.

R. M. Neal. *Bayesian Learning for Neural Networks*. Number 118 in Lecture Notes in Statistics. Springer, 1996.

A. O'Hagan. *Bayesian Inference*, volume 2B of *Kendall's Advanced Theory of Statistics*. Arnold, London, 1994.

B. Olshausen and D. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.

M. Opper and O. Winther. Expectation consistent approximate inference. *Journal of Machine Learning Research*, 6:2177–2204, 2005.

M. Opper and O. Winther. Gaussian processes for classification: Mean field algorithms. *Neural Computation*, 12(11):2655–2684, 2000.

A. Palmer, D. Wipf, K. Kreutz-Delgado, and B. Rao. Variational EM algorithms for non-Gaussian latent variable models. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*. MIT Press, 2006.

L. Paninski. Log-concavity results on Gaussian process methods for supervised and unsupervised learning. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, 2005.

T. Park and G. Casella. The Bayesian Lasso. Technical report, University of Florida, 2005.

R. Peeters and R. Westra. On the identification of sparse gene regulatory networks. In *Proc. 16th Int. Symp. on Math. Theory of Networks*, 2004.

J. Pratt. Concavity of the log likelihood. *Journal of the American Statistical Association*, 76(373): 103–106, 1981.

Y. Qi, T. Minka, R. Picard, and Z. Ghahramani. Predictive automatic relevance determination by expectation propagation. In C. Brodley, editor, *International Conference on Machine Learning 21*. Morgan Kaufmann, 2004.

L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1st edition, 2003.

S. Rogers and M. Girolami. A Bayesian regression approach to the inference of regulatory networks from gene expression data. *Bioinformatics*, 21(14):3131–3137, 2005.

Y. Saad. *Iterative Methods for Sparse Linear Systems*. International Thomson Publishing, 1st edition, 1996.

M. Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, July 2003. See `www.kyb.tuebingen.mpg.de/bs/people/seeger`.

M. Seeger. Low rank updates for the Cholesky decomposition. Technical report, University of California at Berkeley, 2004. See `www.kyb.tuebingen.mpg.de/bs/people/seeger`.

M. Seeger. Expectation propagation for exponential families. Technical report, University of California at Berkeley, 2005. See `www.kyb.tuebingen.mpg.de/bs/people/seeger`.

M. Seeger and H. Nickisch. Compressed sensing and Bayesian experimental design. To appear at ICML, 2008.

M. Seeger, S. Gerwinn, and M. Bethge. Bayesian inference for sparse generalized linear models. In J. Kok, J. Koronacki, R. Lopez, S. Matwin, D. Mladenic, and A. Skowron, editors, *European Conference on Machine Learning 18*. Springer, 2007a.

M. Seeger, F. Steinke, and K. Tsuda. Bayesian inference and optimal design in the sparse linear model. In M. Meila and X. Shen, editors, *Workshop on Artificial Intelligence and Statistics 11*, 2007b.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Conference on Computational Learning Theory 5*, pages 287–294. Morgan Kaufmann, 1992.

D. Spiegelhalter, A. Thomas, N. Best, and W. Gilks. BUGS: Bayesian inference using Gibbs sampling. Technical report, MRC Biostatistics Unit, Cambridge University, 1995.

F. Steinke, M. Seeger, and K. Tsuda. Experimental design for efficient identification of gene regulatory networks using sparse Bayesian models. *BMC Systems Biology*, 1(51), 2007.

J. Tegnér, M. Yeung, J. Hasty, and J. Collins. Reverse engineering gene networks: Integrating genetic perturbations with dynamical modeling. *PNAS*, 100(10):5944–5949, 2003.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of Roy. Stat. Soc. B*, 58: 267–288, 1996.

M. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

M. J. Wainwright. Sharp thresholds for noisy and high-dimensional recovery of sparsity using l_1-constrained quadratic programming. Technical Report 709, UC Berkeley, Dept. of Statistics, 2006.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Dept. of Statistics, 2003.

D. Wipf, J. Palmer, and B. Rao. Perspectives on sparse Bayesian learning. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

D. Wipf, J. Palmer, B. Rao, and K. Kreutz-Delgado. Performance analysis of latent variable models with sparse priors. In *Proceedings of ICASSP 2007*, 2007.

O. Zoeter and T. Heskes. Gaussian quadrature based expectation propagation. In Z. Ghahramani and R. Cowell, editors, *Workshop on Artificial Intelligence and Statistics 10*, 2005.

Journal of Machine Learning Research 1 (2008) 815-857 Submitted 6/05; Revised 2/07; Published 5/08

Finite-Time Bounds for Fitted Value Iteration

Rémi Munos REMI.MUNOS@INRIA.FR
SequeL project, INRIA Lille - Nord Europe
40 avenue Halley
59650 Villeneuve d'Ascq, France

Csaba Szepesvári[*] SZEPESVA@CS.UALBERTA.CA
Department of Computing Science
University of Alberta
Edmonton T6G 2E8, Canada

Editor: Shie Mannor

Abstract

In this paper we develop a theoretical analysis of the performance of sampling-based fitted value iteration (FVI) to solve infinite state-space, discounted-reward Markovian decision processes (MDPs) under the assumption that a generative model of the environment is available. Our main results come in the form of finite-time bounds on the performance of two versions of sampling-based FVI. The convergence rate results obtained allow us to show that both versions of FVI are well behaving in the sense that by using a sufficiently large number of samples for a large class of MDPs, arbitrary good performance can be achieved with high probability. An important feature of our proof technique is that it permits the study of weighted L^p-norm performance bounds. As a result, our technique applies to a large class of function-approximation methods (e.g., neural networks, adaptive regression trees, kernel machines, locally weighted learning), and our bounds scale well with the effective horizon of the MDP. The bounds show a dependence on the stochastic stability properties of the MDP: they scale with the discounted-average concentrability of the future-state distributions. They also depend on a new measure of the approximation power of the function space, the inherent Bellman residual, which reflects how well the function space is "aligned" with the dynamics and rewards of the MDP. The conditions of the main result, as well as the concepts introduced in the analysis, are extensively discussed and compared to previous theoretical results. Numerical experiments are used to substantiate the theoretical findings.

Keywords: fitted value iteration, discounted Markovian decision processes, generative model, reinforcement learning, supervised learning, regression, Pollard's inequality, statistical learning theory, optimal control

1. Introduction

During the last decade, reinforcement learning (RL) algorithms have been successfully applied to a number of difficult control problems, such as job-shop scheduling (Zhang and Dietterich, 1995), backgammon (Tesauro, 1995), elevator control (Crites and Barto, 1997), machine maintenance (Mahadevan et al., 1997), dynamic channel allocation (Singh and Bertsekas, 1997), or airline seat allocation (Gosavi, 2004). The set of possible states in these problems is very large, and so only

∗. Most of this work was done while the author was with the Computer and Automation Research Inst. of the Hungarian Academy of Sciences, Kende u. 13-17, Budapest 1111, Hungary.

algorithms that can successfully generalize to unseen states are expected to achieve non-trivial performance. The approach in the above mentioned works is to learn an approximation to the optimal value function that assigns to each state the best possible expected long-term cumulative reward resulting from starting the control from the selected state. The knowledge of the optimal value function is sufficient to achieve optimal control (Bertsekas and Tsitsiklis, 1996), and in many cases an approximation to the optimal value function is already sufficient to achieve a good control performance. In large state-space problems, a function approximation method is used to represent the learnt value function. In all the above successful applications this is the approach used. Yet, the interaction of RL algorithms and function approximation methods is still not very well understood.

Our goal in this paper is to improve this situation by studying one of the simplest ways to combine RL and function approximation, namely, using function approximation in value iteration. The advantage of studying such a simple combination is that some technical difficulties are avoided, yet, as we will see, the problem studied is challenging enough to make its study worthwhile.

Value iteration is a dynamic programming algorithm which uses 'value backups' to generate a sequence of value functions (i.e., functions defined over the state space) in a recursive manner. After a sufficiently large number of iterations the obtained function can be used to compute a good policy. Exact computations of the value backups require computing parametric integrals over the state space. Except in a few special cases, neither such exact computations, nor the exact representation of the resulting functions is possible. The idea underlying *sampling-based fitted value iteration* (FVI) is to calculate the back-ups approximately using Monte-Carlo integration at a finite number of points and then find a best fit within a user-chosen set of functions to the computed values. The hope is that if the function set is rich enough then the fitted value function will be a good approximation to the next iterate, ultimately leading to a good policy. A large number of successful experimental works concerned algorithms that share many similarities with FVI (e.g., Wang and Dietterich, 1999; Dietterich and Wang, 2002; Lagoudakis and Parr, 2003; Jung and Uthmann, 2004; Ernst et al., 2005; Riedmiller, 2005). Hence, in this paper we concentrate on the theoretical analysis of FVI, as we believe that such an analysis can yield to useful insights into why and when sampling-based approximate dynamic programming (ADP) can be expected to perform well. The relative simplicity of the setup allows a simplified analysis, yet it already shares many of the difficulties that one has to overcome in other, more involved scenarios. (In our followup work we studied other variants of sampling-based ADP. The reader interested in such extensions should consult the papers Antos et al. (2006, 2007, 2008).)

Despite the appealing simplicity of the idea and the successful demonstrations of various sampling-based ADP algorithms, without any further considerations it is still unclear whether sampling-based ADP, and in particular sampling-based FVI is indeed a "good" algorithm. In particular, Baird (1995) and Tsitsiklis and Van Roy (1996) gave simple counterexamples showing that FVI can be unstable. These counterexamples are deceptively simple: the MDP is finite, exact backups can be and are computed, the approximate value function is calculated using a linear combination of a number of fixed basis functions and the optimal value function can be represented exactly by such a linear combination. Hence, the function set seems sufficiently rich. Despite this, the iterates diverge. Since value iteration without projection is well behaved, we must conclude that the instable behavior is the result of the errors introduced when the iterates are projected onto the function space. Our aim in this paper is to develop a better understanding of why, despite the conceivable difficulties, practitioners often find that sampling-based FVI is well behaving and, in particular, we want to develop a theory explaining when to expect good performance.

The setting studied in this paper is as follows: We assume that the state space is a compact subset of a Euclidean space and that the MDP has a finite number of actions. The problem is to find a policy (or controller) that maximizes the expectation of the infinite-horizon, discounted sum of rewards. We shall assume that the solver can sample any transition from any state, that is, that a *generative model* (or simulator) of the environment is available. This model has been used in a number of previous works (e.g., Kearns et al., 1999; Ng and Jordan, 2000; Kakade and Langford, 2002; Kakade, 2003). An extension of the present work to the case when only a single trajectory is available for learning is published elsewhere (Antos et al., 2006).

We investigate two versions of the basic algorithm: In the *multi-sample variant* a fresh sample set is generated in each iteration, while in the *single-sample variant* the same sample set is used throughout all the iterations. Interestingly, we find that no serious degradation of performance results from reusing the samples. In fact, we find that when the discount factor is close to one then the single-sample variant can be expected to be more efficient in the sense of yielding smaller errors using fewer samples. The motivation of this comparison is to get prepared for the case when the samples are given or when they are expensive to generate for some reason.

Our results come in the form of high-probability bounds on the performance as a function of the number of samples generated, some properties of the MDP and the function class used for approximating the value functions. We will compare our bounds to those available in supervised learning (regression), where alike bounds have two terms: one bounding the *bias* of the algorithm, while the other bounding the *variance*, or *estimation error*. The term bounding the bias decreases when the *approximation power* of the function class is increased (hence this term is occasionally called the approximation error term). The term bounding the *variance* decreases as the number of samples is increased, but increases when the richness of the function class is increased.

Although our bounds are similar to bounds of supervised learning, there are some notable differences. In regression estimation, the approximation power of the function set is usually measured w.r.t. (with respect to) some fixed reference class $\mathcal{G}$:

$$d(\mathcal{G},\mathcal{F}) = \sup_{g\in\mathcal{G}} \inf_{f\in\mathcal{F}} \|f-g\|.$$

The reference class $\mathcal{G}$ is typically a classical smoothness class, such as a Lipschitz space. This measure is inadequate for our purposes since in the counterexamples of Baird (1995) and Tsitsiklis and Van Roy (1996) the target function (whatever function space it belongs to) can be approximated with zero error, but FVI still exhibits unbounded errors. In fact, our bounds use a different characterization of the approximation power of the function class $\mathcal{F}$, which we call the *inherent Bellman error* of $\mathcal{F}$:

$$d(T\mathcal{F},\mathcal{F}) = \sup_{g\in\mathcal{F}} \inf_{f\in\mathcal{F}} \|f-Tg\|.$$

Here T is the Bellman operator underlying the MDP (capturing the essential properties of the MDP's dynamics) and $\|\cdot\|$ is an appropriate weighted p-norm that is chosen by the user (the exact definitions will be given in Section 2). Observe that no external reference class is used in the definition of $d(T\mathcal{F},\mathcal{F})$: the inherent Bellman error reflects how well the function space $\mathcal{F}$ is 'aligned' to the Bellman operator, that is, the dynamics of the MDP. In the above-mentioned counterexamples the inherent Bellman error of the chosen function space is infinite and so the bound (correctly) indicates the possibility of divergence.

The bounds on the variance are closer to their regression counterparts: Just like in regression our variance bounds depend on the capacity of the function space used and decay polynomially

with the number of samples. However, the rate of decay is slightly worse than the corresponding rate of (optimal) regression procedures. The difference comes from the bias of approximating the maximum of expected values by the maximum of sample averages. Nevertheless the bounds still indicate that the maximal error of the procedure in the limit when the number of samples grows to infinity converges to a finite positive number. This in turn is controlled by the inherent Bellman error of the function space.

As it was already hinted above, our bounds display the usual *bias-variance tradeoff*: In order to keep both the approximation and estimation errors small, the number of samples and the power of the function class has to be increased simultaneously. When this is done in a principled way, the resulting algorithm becomes *consistent*: It's error in the limit disappears for a large class of MDPs. Consistency is an important property of any MDP algorithm. If an algorithm fails to prove to be consistent, we would be suspicious about its use.

Our bounds apply only to those MDPs whose so-called *discounted-average concentrability of future-state distributions* is finite. The precise meaning of this will be given in Section 5, here we note in passing that this condition holds trivially in every finite MDP, and also, more generally, if the MDP's transition kernel possesses a bounded density. This latter class of MDPs has been considered in many previous theoretical works (e.g., Chow and Tsitsiklis, 1989, 1991; Rust, 1996b; Szepesvári, 2001). In fact, this class of MDPs is quite large in the sense that they contain hard instances. This is discussed in some detail in Section 8. As far as practical examples are concerned, let us mention that resource allocation problems will typically have this property. We will also show a connection between our concentrability factor and Lyapunov exponents, well known from the stability analysis of dynamical systems.

Our proofs build on a recent technique proposed by Munos (2003) that allows the propagation of weighted p-norm losses in approximate value iteration. In contrast, most previous analysis of FVI relied on propagating errors w.r.t. the maximum norm (Gordon, 1995; Tsitsiklis and Van Roy, 1996). The advantage of using p-norm loss bounds is that it allows the analysis of algorithms that use p-norm fitting (in particular, 2-norm fitting). Unlike Munos (2003) and the follow-up work (Munos, 2005), we explicitly deal with infinite state spaces, the effects of using a finite random sample, that is, the bias-variance dilemma and the consistency of sampling-based FVI.

The paper is organized as follows: In the next section we introduce the concepts and notation used in the rest of the paper. The problem is formally defined and the algorithms are given in Section 3. Next, we develop finite-sample bounds for the error committed in a single iteration (Section 4). This bound is used in proving our main results in Section 5. We extend these results to the problem of obtaining a good policy in Section 6, followed by a construction that allows one to achieve asymptotic consistency when the unknown MDP is smooth with an unknown smoothness factor (Section 7). Relationship to previous works is discussed in details in Section 8. An experiment in a simulated environment, highlighting the main points of the analysis is given in Section 9. The proofs of the statements are given in the Appendix.

2. Markovian Decision Processes

A discounted *Markovian Decision Process* (discounted MDP) is a 5-tuple $(\mathcal{X}, \mathcal{A}, P, S, \gamma)$, where $\mathcal{X}$ is the *state space*, $\mathcal{A}$ is the *action space*, P is the *transition probability kernel*, S is *the reward kernel* and $0 < \gamma < 1$ is the *discount factor* (Bertsekas and Shreve, 1978; Puterman, 1994). In this paper we consider continuous state space, finite action MDPs (i.e., $|\mathcal{A}| < +\infty$). For the sake of

simplicity we assume that $\mathcal{X}$ is a bounded, closed subset of a Euclidean space, $\mathbb{R}^d$. The system of Borel-measurable sets of $\mathcal{X}$ shall be denoted by $\mathcal{B}(\mathcal{X})$.

The interpretation of an MDP as a control problem is as follows: Each initial state X_0 and action sequence $a_0, a_1, \ldots$ gives rise to a sequence of states $X_1, X_2, \ldots$ and rewards $R_1, R_2, \ldots$ satisfying, for any B and C Borel-measurable sets the equalities

$$\mathbb{P}(X_{t+1} \in B | X_t = x, a_t = a) = P(B|x,a),$$

and

$$\mathbb{P}(R_t \in C | X_t = x, a_t = a) = S(C|x,a).$$

Equivalently, we write $X_{t+1} \sim P(\cdot|X_t, a)$, $R_t \sim S(\cdot|X_t, a)$. In words, we say that when action a_t is executed from state $X_t = x$ the process makes a transition from x to the next state X_{t+1} and a reward, R_t, is incurred. The history of the process up to time t is $H_t = (X_0, a_0, R_0, \ldots, X_{t-1}, a_{t-1}, R_{t-1}, X_t)$. We assume that the random rewards $\{R_t\}$ are bounded by some positive number $\hat{R}_{\max}$, w.p. 1 (with probability one).[1]

A *policy* is a sequence of functions that maps possible histories to probability distributions over the space of actions. Hence if the space of histories at time step t is denoted by $\mathcal{H}_t$ then a policy π is a sequence $\pi_0, \pi_1, \ldots$, where π_t maps $\mathcal{H}_t$ to $M(\mathcal{A})$, the space of all probability distributions over $\mathcal{A}$.[2] 'Following a policy' means that for any time step t given the history $x_0, a_0, \ldots, x_t$ the probability of selecting an action a equals $\pi_t(x_0, a_0, \ldots, x_t)(a)$. A policy is called *stationary* if π_t depends only on the last state visited. Equivalently, a policy $\pi = (\pi_0, \pi_1, \ldots)$ is called stationary if $\pi_t(x_0, a_0, \ldots, x_t) = \pi_0(x_t)$ holds for all $t \geq 0$. A policy is called *deterministic* if for any history $x_0, a_0, \ldots, x_t$ there exists some action a such that $\pi_t(x_0, a_0, \ldots, x_t)$ is concentrated on this action. Hence, any deterministic stationary policy can be identified by some mapping from the state space to the action space and so in the following, at the price of abusing the notation and the terminology slightly, we will call such mappings policies, too.

The goal is to find a policy π that maximizes the expected total discounted reward given any initial state. Under this criterion the value of a policy π and a state $x \in \mathcal{X}$ is given by

$$V^\pi(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t^\pi | X_0 = x\right],$$

where R_t^π is the reward incurred at time t when executing policy π. The optimal expected total discounted reward when the process is started from state x shall be denoted by $V^*(x)$; V^* is called the *optimal value function* and is defined by $V^*(x) = \sup_\pi V^\pi(x)$. A policy π is called *optimal* if it attains the optimal values for *any* state $x \in \mathcal{X}$, that is if $V^\pi(x) = V^*(x)$ for all $x \in \mathcal{X}$. We also let $Q^*(x,a)$ denote the long-term total expected discounted reward when the process is started from state x, the first executed action is a and it is assumed that after the first step an optimal policy is followed. Since by assumption the action space is finite, the rewards are bounded, and we assume discounting, the existence of deterministic stationary optimal policies is guaranteed (Bertsekas and Shreve, 1978).

1. This condition could be replaced by a standard moment condition on the random rewards (Györfi et al., 2002) without changing the results.
2. In fact, π_t must be a measurable mapping so that we are allowed to talk about the probability of executing an action. Measurability issues by now are well understood and hence we shall not deal with them here. For a complete treatment the interested reader is referred to Bertsekas and Shreve (1978).

Let us now introduce a few function spaces and operators that will be needed in the rest of the paper. Let us denote the space of bounded measurable functions with domain $\mathcal{X}$ by $B(\mathcal{X})$. Further, the space of measurable functions bounded by $0 < V_{\max} < +\infty$ shall be denoted by $B(\mathcal{X}; V_{\max})$. A deterministic stationary policy $\pi : \mathcal{X} \to \mathcal{A}$ defines the transition probability kernel P^π according to $P^\pi(dy|x) = P(dy|x, \pi(x))$. From this kernel two related operators are derived: a right-linear operator, $P^\pi \cdot : B(\mathcal{X}) \to B(\mathcal{X})$, defined by

$$(P^\pi V)(x) = \int V(y) P^\pi(dy|x),$$

and a left-linear operator, $\cdot P^\pi : M(\mathcal{X}) \to M(\mathcal{X})$, defined by

$$(\mu P^\pi)(dy) = \int P^\pi(dy|x) \mu(dx).$$

Here $\mu \in M(\mathcal{X})$ and $M(\mathcal{X})$ is the space of all probability distributions over $\mathcal{X}$.

In words, $(P^\pi V)(x)$ is the expected value of V after following π for a single time-step when starting from x, and μP^π is the distribution of states if the system is started from $X_0 \sim \mu$ and policy π is followed for a single time-step. The product of two kernels P^{π_1} and P^{π_2} is defined in the natural way:

$$(P^{\pi_1} P^{\pi_2})(dz|x) = \int P^{\pi_1}(dy|x) P^{\pi_2}(dz|y).$$

Hence, $\mu P^{\pi_1} P^{\pi_2}$ is the distribution of states if the system is started from $X_0 \sim \mu$, policy π_1 is followed for the first step and then policy π_2 is followed for the second step. The interpretation of $(P^{\pi_1} P^{\pi_2} V)(x)$ is similar.

We say that a (deterministic stationary) policy π is *greedy* w.r.t. a function $V \in B(\mathcal{X})$ if, for all $x \in \mathcal{X}$,

$$\pi(x) \in \arg\max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int V(y) P(dy|x,a) \right\},$$

where $r(x,a) = \int z S(dz|x,a)$ is the expected reward of executing action a in state x. We assume that r is a bounded, measurable function. Actions maximizing $r(x,a) + \gamma \int V(y) P(dy|x,a)$ are said to be *greedy* w.r.t. V. Since $\mathcal{A}$ is finite the set of greedy actions is non-empty for any function V.

Define operator $T : B(\mathcal{X}) \to B(\mathcal{X})$ by

$$(TV)(x) = \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int V(y) P(dy|x,a) \right\}, \quad V \in B(\mathcal{X}).$$

Operator T is called the *Bellman operator* underlying the MDP. Similarly, to any stationary deterministic policy π there corresponds an operator $T^\pi : B(\mathcal{X}) \to B(\mathcal{X})$ defined by

$$(T^\pi V)(x) = r(x, \pi(x)) + (P^\pi V)(x).$$

It is well known that T is a contraction mapping in supremum norm with contraction coefficient γ: $\|TV - TV'\|_\infty \le \gamma \|V - V'\|_\infty$. Hence, by Banach's fixed-point theorem, T possesses a unique fixed point. Moreover, this fixed point turns out to be equal to the optimal value function, V^*. Then a simple contraction argument shows that the so-called *value-iteration algorithm*,

$$V_{k+1} = TV_k,$$

with arbitrary $V_0 \in B(\mathcal{X})$ yields a sequence of iterates, V_k, that converge to V^* at a geometric rate. The contraction arguments also show that if $|r(x,a)|$ is bounded by $R_{\max} > 0$ then V^* is bounded by $R_{\max}/(1-\gamma)$ and if $V_0 \in B(\mathcal{X}; R_{\max}/(1-\gamma))$ then the same holds for V_k, too. Proofs of these statements can be found in many textbooks such as in that of Bertsekas and Shreve (1978).

Our initial set of assumptions on the class of MDPs considered is summarized as follows:

Assumption A0 [MDP Regularity] The MDP $(\mathcal{X}, \mathcal{A}, P, S, \gamma)$ satisfies the following conditions: $\mathcal{X}$ is a bounded, closed subset of some Euclidean space, $\mathcal{A}$ is finite and the discount factor γ satisfies $0 < \gamma < 1$. The reward kernel S is such that the immediate reward function r is a bounded measurable function with bound $R_{\max}$. Further, the support of $S(\cdot|x,a)$ is included in $[-\hat{R}_{\max}, \hat{R}_{\max}]$ independently of $(x,a) \in \mathcal{X} \times \mathcal{A}$.

Let μ be a distribution over $\mathcal{X}$. For a real-valued measurable function g defined over $\mathcal{X}$, $\|g\|_{p,\mu}$ is defined by $\|g\|_{p,\mu}^p = \int |g(x)|^p \mu(dx)$. The space of functions with bounded $\|\cdot\|_{p,\mu}$-norm shall be denoted by $L^p(\mathcal{X}; \mu)$.

3. Sampling-based Fitted Value Iteration

The parameters of sampling-based FVI are a distribution, $\mu \in M(\mathcal{X})$, a function set $\mathcal{F} \subset B(\mathcal{X})$, an initial value function, $V_0 \in \mathcal{F}$, and the integers N, M and K. The algorithm works by computing a series of functions, $V_1, V_2, \ldots \in \mathcal{F}$ in a recursive manner. The $(k+1)$th iterate is obtained from the kth function as follows: First a Monte-Carlo estimate of TV_k is computed at a number of random states $(X_i)_{1 \le i \le N}$:

$$\hat{V}(X_i) = \max_{a \in \mathcal{A}} \frac{1}{M} \sum_{j=1}^{M} \left[R_j^{X_i,a} + \gamma V_k(Y_j^{X_i,a}) \right], \quad i = 1, 2, \ldots, N.$$

Here the *basepoints*, $X_1, \ldots, X_N$, are sampled from the distribution μ, independently of each other. For each of these basepoints and for each possible action $a \in \mathcal{A}$ the next states, $Y_j^{X_i,a} \in \mathcal{X}$, and rewards, $R_j^{X_i,a} \in \mathbb{R}$, are drawn via the help of the generative model of the MDP:

$$\begin{aligned} Y_j^{X_i,a} &\sim P(\cdot|X_i,a), \\ R_j^{X_i,a} &\sim S(\cdot|X_i,a), \end{aligned}$$

$(j = 1, 2, \ldots, M,\ i = 1, \ldots, N)$. By assumption, $(Y_j^{X_i,a}, R_j^{X_i,a})$ and $(Y_{j'}^{X_{i'},a'}, R_{j'}^{X_{i'},a'})$ are independent of each other whenever $(i,j,a) \neq (i',j',a')$. The next iterate V_{k+1} is obtained as the best fit in $\mathcal{F}$ to the data $(X_i, \hat{V}(X_i))_{i=1,2,\ldots,N}$ w.r.t. the p-norm based empirical loss

$$V_{k+1} = \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{N} |f(X_i) - \hat{V}(X_i)|^p. \tag{1}$$

These steps are iterated $K > 0$ times, yielding the sequence $V_1, \ldots, V_K$.

We study two variants of this algorithm: When a fresh sample is generated in each iteration, we call the algorithm the *multi-sample variant*. The *total number* of samples used by the multi-sample algorithm is thus $K \times N \times M$. Since in a single iteration only a fraction of these samples is used, one may wonder if it were more sample efficient to use *all the samples* in all the iterations.[3] We shall call

3. Sample-efficiency becomes an issue when the sample generation process is not controlled (the samples are given) or when it is expensive to generate the samples due to the high cost of simulation.

the version of the algorithm the *single-sample variant* (details will be given in Section 5). A possible counterargument against the single-sample variant is that since the samples used in subsequent iterations are correlated, the bias due to sampling errors may get amplified. One of our interesting theoretical findings is that the bias-amplification effect is not too severe and, in fact, the single-sample variant of the algorithm is well behaving and can be expected to outperform the multi-sample variant. In the experiments we will see such a case.

Let us now discuss the choice of the function space $\mathcal{F}$. Generally, $\mathcal{F}$ is selected to be a finitely parameterized class of functions:

$$\mathcal{F} = \{ f_\theta \in B(\mathcal{X}) \,|\, \theta \in \Theta \}, \quad \dim(\Theta) < +\infty.$$

Our results apply to both linear ($f_\theta(x) = \theta^T \phi(x)$) and non-linear ($f_\theta(x) = f(x;\theta)$) parameterizations, such as wavelet based approximations, multi-layer neural networks or kernel-based regression techniques. Another possibility is to use the kernel mapping idea underlying many recent state-of-the-art supervised-learning methods, such as support-vector machines, support-vector regression or Gaussian processes (Cristianini and Shawe-Taylor, 2000). Given a (positive definite) kernel function $\mathcal{K}$, $\mathcal{F}$ can be chosen as a closed convex subset of the reproducing-kernel Hilbert-space (RKHS) associated to $\mathcal{K}$. In this case the optimization problem (1) still admits a finite, closed-form solution despite that the function space $\mathcal{F}$ cannot be written in the above form for any finite dimensional parameters space (Kimeldorf and Wahba, 1971; Schölkopf and Smola, 2002).

4. Approximating the Bellman Operator

The purpose of this section is to bound the error introduced in a single iteration of the algorithm. There are two components of this error: The approximation error caused by projecting the iterates into the function space $\mathcal{F}$ and the estimation error caused by using a finite, random sample.

The approximation error can be best explained by introducing the metric projection operator: Fix the sampling distribution $\mu \in M(\mathcal{X})$ and let $p \geq 1$. The *metric projection* of TV onto $\mathcal{F}$ w.r.t. the μ-weighted p-norm is defined by

$$\Pi_{\mathcal{F}} TV = \operatorname*{argmin}_{f \in \mathcal{F}} \| f - TV \|_{p,\mu}.$$

Here $\Pi_{\mathcal{F}} : L^p(\mathcal{X};\mu) \to \mathcal{F}$ for $g \in L^p(\mathcal{X};\mu)$ gives the *best approximation* to g in $\mathcal{F}$.[4] The *approximation error* in the kth step for $V = V_k$ is $d_{p,\mu}(TV, \mathcal{F}) = \| \Pi_{\mathcal{F}} TV - TV \|_{p,\mu}$. More generally, we let

$$d_{p,\mu}(TV, \mathcal{F}) = \inf_{f \in \mathcal{F}} \| f - TV \|_{p,\mu}.$$

Hence, the approximation error can be made small by selecting $\mathcal{F}$ to be large enough. We shall discuss how this can be accomplished for a large class of MDPs in Section 7.

4. The existence and uniqueness of best approximations is one of the fundamental problems of approximation theory. Existence can be guaranteed under fairly mild conditions, such as the compactness of $\mathcal{F}$ w.r.t. $\|\cdot\|_{p,\mu}$, or if $\mathcal{F}$ is finite dimensional (Cheney, 1966). Since the metric projection operator is needed for discussion purposes only, here we simply assume that $\Pi_{\mathcal{F}}$ is well-defined.

Let us now turn to the discussion of the estimation error. In the kth iteration, given $V = V_k$, the function $V' = V_{k+1}$ is computed as follows:

$$\hat{V}(X_i) = \max_{a \in \mathcal{A}} \frac{1}{M} \sum_{j=1}^{M} \left[R_j^{X_i,a} + \gamma V(Y_j^{X_i,a}) \right], \quad i = 1, 2, \ldots, N, \tag{2}$$

$$V' = \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{N} \left| f(X_i) - \hat{V}(X_i) \right|^p, \tag{3}$$

where the random samples satisfy the conditions of the previous section and, for the sake of simplicity, we assume that the minimizer in Equation (3) exists.

Clearly, for a fixed X_i, $\max_a 1/M \sum_{j=1}^{M} (R_j^{X_i,a} + \gamma V(Y_j^{X_i,a})) \to (TV)(X_i)$ as $M \to \infty$ w.p.1. Hence, for large enough M, $\hat{V}(X_i)$ is a good approximation to $(TV)(X_i)$. On the other hand, if N is big then for any $(f,g) \in \mathcal{F} \times \mathcal{F}$, the *empirical p-norm loss*, $1/N \sum_{i=1}^{N} (f(X_i) - g(X_i))^p$, is a good approximation to the *true loss* $\|f - g\|_{p,\mu}^p$. Hence, we expect to find the minimizer of (3) to be close to the minimizer of $\|f - TV\|_{p,\mu}^p$. Since the function x^p is strictly increasing for $x > 0$, $p > 0$, the minimizer of $\|f - TV\|_{p,\mu}^p$ over $\mathcal{F}$ is just the metric projection of TV on $\mathcal{F}$, hence for N, M big, V' can be expected to be close to $\Pi_{\mathcal{F}} TV$.

Note that Equation (3) looks like an ordinary p-norm function fitting. One difference though is that in regression the target function equals the regressor $g(x) = E[\hat{V}(X_i) | X_i = x]$, whilst in our case the target function is TV and $TV \neq g$. This is because

$$\mathbb{E}\left[\max_{a \in \mathcal{A}} \frac{1}{M} \sum_{j=1}^{M} \left[R_j^{X_i,a} + \gamma V(Y_j^{X_i,a}) \right] \,\middle|\, X_i \right] \geq \max_{a \in \mathcal{A}} \mathbb{E}\left[\frac{1}{M} \sum_{j=1}^{M} \left[R_j^{X_i,a} + \gamma V(Y_j^{X_i,a}) \right] \,\middle|\, X_i \right].$$

In fact, if we had an equality here then we would have no reason to set $M > 1$: in a pure regression setting it is always better to have a completely fresh pair of samples than to have a pair where the covariate is set to be equal to some previous sample. Due to $M > 1$ the rate of convergence with the sample size of sampling-based FVI will be slightly worse than the rates available for regression.

Above we argued that for N large enough and for a fixed pair $(f,g) \in \mathcal{F} \times \mathcal{F}$, the empirical loss will approximate the true loss, that is, the estimation error will be small. However, we need this property to hold for V'. Since V' is the minimizer of the empirical loss, it depends on the random samples and hence it is a random object by itself and so the argument that the estimation error is small for any *fixed*, deterministic pair of functions cannot be used with V'. This is, however, the situation in supervised learning problems, too. A simple idea developed there is to bound the estimation error of V' by the worst-case estimation error over $\mathcal{F}$:

$$\left| \frac{1}{N} \sum_{i=1}^{N} |V'(X_i) - g(X_i)|^p - \|V' - g\|_{p,\mu}^p \right| \leq \sup_{f \in \mathcal{F}} \left| \frac{1}{N} \sum_{i=1}^{N} |f(X_i) - g(X_i)|^p - \|f - g\|_{p,\mu}^p \right|.$$

This inequality holds w.p. 1 since for any random event ω, $V' = V'(\omega)$ is an element of $\mathcal{F}$. The right-hand side here is the maximal deviation of a large number of empirical averages from their respective means. The behavior of this quantity is the main focus of empirical process theory and we shall use the tools developed there, in particular Pollard's tail inequality (cf., Theorem 5 in Appendix A).

When bounding the size of maximal deviations, the size of the function set becomes a major factor. When the function set has a finite number of elements, a bound follows by exponential tail

inequalities and a union bounding argument. When $\mathcal{F}$ is infinite, the 'capacity' of $\mathcal{F}$ measured by the (empirical) *covering number* of $\mathcal{F}$ can be used to derive an appropriate bound: Let $\varepsilon > 0$, $q \geq 1$, $x^{1:N} \stackrel{\text{def}}{=} (x_1, \ldots, x_N) \in \mathcal{X}^N$ be fixed. The (ε, q)-*covering number of the set* $\mathcal{F}(x^{1:N}) = \{(f(x_1), \ldots, f(x_N)) \,|\, f \in \mathcal{F}\}$ is the smallest integer m such that $\mathcal{F}(x^{1:N})$ can be covered by m balls of the normed-space $(\mathbb{R}^N, \|\cdot\|_q)$ with centers in $\mathcal{F}(x^{1:N})$ and radius $N^{1/q}\varepsilon$. The (ε, q)-covering number of $\mathcal{F}(x^{1:N})$ is denoted by $\mathcal{N}_q(\varepsilon, \mathcal{F}(x^{1:N}))$. When $q = 1$, we use $\mathcal{N}$ instead of $\mathcal{N}_1$. When $X^{1:N}$ are i.i.d. with common underlying distribution μ then $\mathbb{E}\left[\mathcal{N}_q(\varepsilon, \mathcal{F}(X^{1:N}))\right]$ shall be denoted by $\mathcal{N}_q(\varepsilon, \mathcal{F}, N, \mu)$. By Jensen's inequality $\mathcal{N}_p \leq \mathcal{N}_q$ for $p \leq q$. The logarithm of $\mathcal{N}_q$ is called the *q-norm metric entropy* of $\mathcal{F}$. For $q = 1$, we shall call $\log \mathcal{N}_1$ the *metric entropy* of $\mathcal{F}$ (without any qualifiers).

The idea underlying covering numbers is that what really matters when bounding maximal deviations is how much the functions in the function space vary *at the actual samples*. Of course, without imposing any conditions on the function space, covering numbers can grow as a function of the sample size.

For specific choices of $\mathcal{F}$, however, it is possible to bound the covering numbers of $\mathcal{F}$ *independently* of the number of samples. In fact, according to a well-known result due to Haussler (1995), covering numbers can be bounded as a function of the so-called *pseudo-dimension* of the function class. The pseudo-dimension, or VC-subgraph dimension $V_{\mathcal{F}^+}$ of $\mathcal{F}$ is defined as the VC-dimension of the subgraphs of functions in $\mathcal{F}$.[5] The following statement gives the bound that does not depend on the number of sample points:

Proposition 1 (Haussler (1995), Corollary 3) *For any set $\mathcal{X}$, any points $x^{1:N} \in \mathcal{X}^N$, any class $\mathcal{F}$ of functions on $\mathcal{X}$ taking values in $[0, L]$ with pseudo-dimension $V_{\mathcal{F}^+} < \infty$, and any $\varepsilon > 0$,*

$$\mathcal{N}(\varepsilon, \mathcal{F}(x^{1:N})) \leq e(V_{\mathcal{F}^+} + 1)\left(\frac{2eL}{\varepsilon}\right)^{V_{\mathcal{F}^+}}.$$

For a given set of functions $\mathcal{F}$ let $a + \mathcal{F}$ denote the set of functions shifted by the constant a: $a + \mathcal{F} = \{f + a \,|\, f \in \mathcal{F}\}$. Clearly, neither the pseudo-dimension nor covering numbers are changed by shifts. This allows one to extend Proposition 1 to function sets with functions taking values in $[-L, +L]$.

Bounds on the pseudo-dimension are known for many popular function classes including linearly parameterized function classes, multi-layer perceptrons, radial basis function networks, several non-and semi-parametric function classes (cf., Niyogi and Girosi, 1999; Anthony and Bartlett, 1999; Györfi et al., 2002; Zhang, 2002, and the references therein). If q is the dimensionality of the function space, these bounds take the form $O(\log(q))$, $O(q)$ or $O(q \log q)$.[6]

Another route to get a useful bound on the number of samples is to derive an upper bound on the metric entropy that grows with the number of samples at a *sublinear* rate. As an example consider the following class of bounded-weight, linearly parameterized functions:

$$\mathcal{F}_A = \{f_\theta : \mathcal{X} \to \mathbb{R} \,|\, f_\theta(x) = \theta^T \phi(x), \|\theta\|_q \leq A\}.$$

5. The VC-dimension of a set system is defined as follows (Sauer, 1972; Vapnik and Chervonenkis, 1971): Given a set system $\mathcal{C}$ with base set $\mathcal{U}$ we say that $\mathcal{C}$ shatters the points of $A \subset \mathcal{U}$ if all possible $2^{|A|}$ subsets of A can be obtained by intersecting A with elements of $\mathcal{C}$. The VC-dimension of $\mathcal{C}$ is the cardinality of the largest subset $A \subset \mathcal{U}$ that can be shattered.

6. Again, similar bounds are known to hold for the supremum-norm metric entropy.

It is known that for finite-dimensional smooth parametric classes their metric entropy scales with $\dim(\phi) < +\infty$. If ϕ is the feature map associated with some positive definite kernel function $\mathcal{K}$ then ϕ can be infinite dimensional (this class arises if one 'kernelizes' FVI). In this case the bounds due to Zhang (2002) can be used. These bound the metric entropy by $\lceil A^2B^2/\varepsilon^2 \rceil \log(2N+1)$, where B is an upper bound on $\sup_{x\in\mathcal{X}} \|\phi(x)\|_p$ with $p = 1/(1-1/q)$.[7]

4.1 Finite-sample Bounds

The following lemma shows that with high probability, V' is a good approximation to TV when some element of $\mathcal{F}$ is close to TV and if the number of samples is high enough:

Lemma 1 *Consider an MDP satisfying Assumption A0. Let $V_{\max} = R_{\max}/(1-\gamma)$, fix a real number $p \geq 1$, integers $N, M \geq 1$, $\mu \in M(\mathcal{X})$ and $\mathcal{F} \subset B(\mathcal{X}; V_{\max})$. Pick any $V \in B(\mathcal{X}; V_{\max})$ and let $V' = V'(V, N, M, \mu, \mathcal{F})$ be defined by Equation (3). Let $\mathcal{N}_0(N) = \mathcal{N}\left(\frac{1}{8}\left(\frac{\varepsilon}{4}\right)^p, \mathcal{F}, N, \mu\right)$. Then for any $\varepsilon, \delta > 0$,*

$$\left\|V' - TV\right\|_{p,\mu} \leq d_{p,\mu}(TV, \mathcal{F}) + \varepsilon$$

holds w.p. at least $1-\delta$ provided that

$$N > 128 \left(\frac{8V_{\max}}{\varepsilon}\right)^{2p} \left(\log(1/\delta) + \log(32\mathcal{N}_0(N))\right) \tag{4}$$

and

$$M > \frac{8\left(\hat{R}_{\max} + \gamma V_{\max}\right)^2}{\varepsilon^2} \left(\log(1/\delta) + \log(8N|\mathcal{A}|)\right). \tag{5}$$

As we have seen before, for a large number of choices of $\mathcal{F}$, the metric entropy of $\mathcal{F}$ is independent of N. In such cases Equation (4) gives an explicit bound on N and M. In the resulting bound, the total number of samples per iteration, $N \times M$, scales with $\varepsilon^{-(2p+2)}$ (apart from logarithmic terms). The comparable bound for the pure regression setting is ε^{-2p}. The additional quadratic factor is the price to pay because of the biasedness of the values $\hat{V}(X_i)$.

The main ideas of the proof are illustrated using Figure 1 (the proof of the Lemma can be found in Appendix A). The left-hand side of this figure depicts the space of bounded functions over $\mathcal{X}$, while the right-hand side figure shows a corresponding vector space. The spaces are connected by the mapping $f \mapsto \tilde{f} \stackrel{\text{def}}{=} (f(X_1), \ldots, f(X_N))^T$. In particular, this mapping sends the set $\mathcal{F}$ into the set $\tilde{F} = \{\tilde{f} \mid f \in \mathcal{F}\}$.

The proof goes by upper bounding the distance between V' and TV in terms of the distance between f^* and TV. Here $f^* = \Pi_{\mathcal{F}} TV$ is the best fit to TV in $\mathcal{F}$. The choice of f^* is motivated by the fact that V' is the best fit in $\mathcal{F}$ to the data $(X_i, \hat{V}(X_i))_{i=1,\ldots,N}$ w.r.t. the p-norm $\|\cdot\|_p$. The bound is developed by relating a series of distances to each other: In particular, if N is large then $\|V' - TV\|_{p,\mu}^p$ and $\|\widetilde{V'} - \widetilde{TV}\|_p^p$ are expected to be close to each other. On the other hand, if M is large then $\hat{V}$ and $\widetilde{TV}$ are expected to be close to each other. Hence, $\|\widetilde{V'} - \widetilde{TV}\|_p^p$ and $\|\widetilde{V'} - \hat{V}\|_p^p$ are expected to be close to each other. Now, since $\widetilde{V'}$ is the best fit to $\hat{V}$ in $\widetilde{\mathcal{F}}$, the distance between $\widetilde{V'}$ and $\hat{V}$ is not larger than the distance between the image $\widetilde{f}$ of an arbitrary function $f \in \mathcal{F}$ and $\hat{V}$. Choosing $f = f^*$ we conclude that the distance between $\widetilde{f^*}$ and $\hat{V}$ is not smaller than $\|\widetilde{V'} - \hat{V}\|_p^p$.

7. Similar bounds exist for the supremum-norm metric entropy.

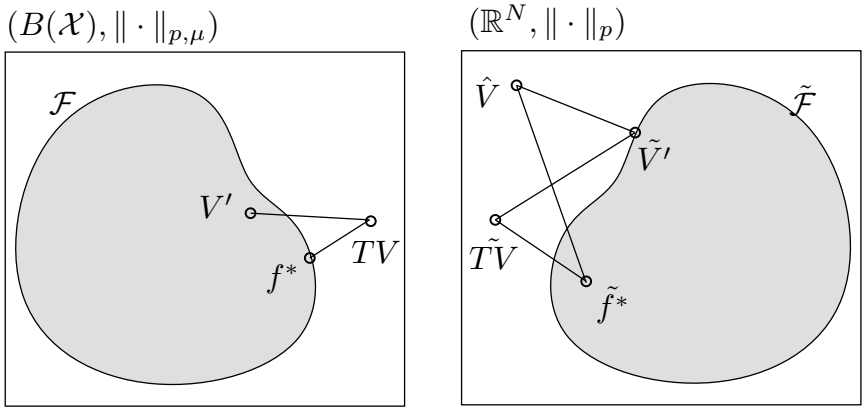


Figure 1: Illustration of the proof of Lemma 1 for bounding the distance of V' and TV in terms of the distance of f^* and TV, where f^* is the best fit to TV in $\mathcal{F}$ (cf., Equations 2, 3). For a function $f \in B(\mathcal{X})$, $\widetilde{f} = (f(X_1), \ldots, f(X_N))^T \in \mathbb{R}^N$. The set $\widetilde{\mathcal{F}}$ is defined by $\{\widetilde{f} \,|\, f \in \mathcal{F}\}$. Segments on the figure connect objects whose distances are compared in the proof.

Exploiting again that M is large, we see that the distance between $\widetilde{f^*}$ and $\hat{V}$ must be close to that of between $\widetilde{f^*}$ and $\widetilde{TV}$, which in turn must be close to the $L^p(\mathcal{X};\mu)$ distance of f^* and TV if N is big enough. Hence, if $\|f^* - TV\|_{p,\mu}^p$ is small then so is $\|V' - TV\|_{p,\mu}^p$.

4.2 Bounds for the Single-Sample Variant

When analyzing the error of sampling-based FVI, we would like to use Lemma 1 for bounding the error committed when approximating TV_k starting from V_k based on a new sample. When doing so, however, we have to take into account that V_k is random. Yet Lemma 1 requires that V, the function whose Bellman image is approximated, is some fixed (non-random) function. The problem is easily resolved in the multi-sample variant of the algorithm by noting that the samples used in calculating V_{k+1} are independent of the samples used to calculate V_k. A formal argument is presented in Appendix B.3. The same argument, however, does not work for the single-sample variant of the algorithm when V_{k+1} and V_k are *both* computed using the *same set of random variables*. The purpose of this section is to extend Lemma 1 to cover this case.

In formulating this result we will need the following definition: For $\mathcal{F} \subset B(\mathcal{X})$ let us define

$$\mathcal{F}_{T-} = \{\, f - Tg \,|\, f \in \mathcal{F}, g \in \mathcal{F} \,\}.$$

The following result holds:

Lemma 2 *Denote by Ω the sample-space underlying the random variables $\{X_i\}$, $\{Y_j^{X_i,a}\}$, $\{R_j^{X_i,a}\}$, $i = 1,\ldots,N, j = 1,\ldots,M, a \in \mathcal{A}$. Then the result of Lemma 1 continues to hold if V is a random function satisfying $V(\omega) \in \mathcal{F}$, $\omega \in \Omega$ provided that*

$$N = O(V_{\max}^2 (1/\varepsilon)^{2p} \log(\mathcal{N}(c\varepsilon, \mathcal{F}_{T-}, N, \mu)/\delta))$$

and

$$M = O((\hat{R}_{\max} + \gamma V_{\max})^2/\varepsilon^2 \log(N|\mathcal{A}|\mathcal{N}(c'\varepsilon, \mathcal{F}, M, \mu)/\delta)),$$

where $c, c' > 0$ are constants independent of the parameters of the MDP and the function space $\mathcal{F}$.

The proof can be found in Appendix A.1. Note that the sample-size bounds in this lemma are similar to those of Lemma 1, except that N now depends on the metric entropy of $\mathcal{F}_{T-}$ and M depends on the metric entropy of $\mathcal{F}$. Let us now give two examples when explicit bounds on the covering number of $\mathcal{F}_{T-}$ can be given using simple means:

For the first example note that if $g : (\mathbb{R}\times\mathbb{R}, \|\cdot\|_1) \to \mathbb{R}$ is Lipschitz[8] with Lipschitz constant G then the ε-covering number of the space of functions of the form $h(x) = g(f_1(x), f_2(x))$, $f_1 \in \mathcal{F}_1$, $f_2 \in \mathcal{F}_2$ can be bounded by $\mathcal{N}(\varepsilon/(2G), \mathcal{F}_1, n, \mu)\mathcal{N}(\varepsilon/(2G), \mathcal{F}_2, n, \mu)$ (this follows directly from the definition of covering numbers). Since $g(x,y) = x - y$ is Lipschitz with $G = 1$, $\mathcal{N}(\varepsilon, \mathcal{F}_{T-}, n, \mu) \le \mathcal{N}(\varepsilon/2, \mathcal{F}, n, \mu)\mathcal{N}(\varepsilon/2, \mathcal{F}_T, n, \mu)$. Hence it suffices to bound the covering numbers of the space $\mathcal{F}_T = \{Tf | f \in \mathcal{F}\}$. One possibility to do this is as follows: Assume that $\mathcal{X}$ is compact, $\mathcal{F} = \{f_\theta | \theta \in \Theta\}$, Θ is compact and the mapping $H : (\Theta, \|\cdot\|) \to (B(\mathcal{X}), L^\infty)$ defined by $H(\theta) = f_\theta$ is Lipschitz with coefficient L. Fix $x^{1:n}$ and consider $\mathcal{N}(\varepsilon, \mathcal{F}_T(x^{1:n}))$. Let θ_1, θ_2 be arbitrary. Then $|Tf_{\theta_1}(x) - Tf_{\theta_2}(x)| \le \|Tf_{\theta_1} - Tf_{\theta_2}\|_\infty \le \gamma \|f_{\theta_1} - f_{\theta_2}\|_\infty \le \gamma L \|\theta_1 - \theta_2\|$. Now assume that $C = \{\theta_1, \dots, \theta_m\}$ is an $\varepsilon/(L\gamma)$-cover of the space Θ and consider any $n \ge 1$, $(x_1, \dots, x_n) \in \mathcal{X}^n$, $\theta \in \Theta$. Let θ_i be the nearest neighbor of θ in C. Then, $\left\|(Tf_\theta)(x^{1:n}) - (Tf_{\theta_i})(x^{1:n})\right\|_1 \le n\|Tf_\theta - Tf_{\theta_i}\|_\infty \le n\varepsilon$. Hence, $\mathcal{N}(\varepsilon, \mathcal{F}_T(x^{1:n})) \le \mathcal{N}(\varepsilon/(L\gamma), \Theta)$.

Note that the mapping H can be shown to be Lipschitzian for many function spaces of interest. As an example let us consider the space of linearly parameterized functions taking the form $f_\theta = \theta^T\phi$ with a suitable basis function $\phi : \mathcal{X} \to \mathbb{R}^{d_\phi}$. By the Cauchy-Schwarz inequality, $\left\|\theta_1^T\phi - \theta_2^T\phi\right\|_\infty = \sup_{x\in\mathcal{X}} |\langle\theta_1 - \theta_2, \phi(x)\rangle| \le \|\theta_1 - \theta_2\|_2 \sup_{x\in\mathcal{X}} \|\phi(x)\|_2$. Hence, by choosing the ℓ^2 norm in the space Θ, we get that $\theta \mapsto \theta^T\phi$ is Lipschitz with coefficient $\|\|\phi(\cdot)\|_2\|_\infty$ (this gives a bound on the metric entropy that is linear in d_ϕ).

5. Main Results

For the sake of specificity, let us reiterate the algorithms. Let $V_0 \in \mathcal{F}$. The *single-sample variant* of sampling-based FVI produces a sequence of function $\{V_k\}_{0\le k\le K} \subset \mathcal{F}$ satisfying

$$V_{k+1} = \operatorname*{argmin}_{f\in\mathcal{F}} \sum_{i=1}^{N} \left| f(X_i) - \max_{a\in\mathcal{A}} \frac{1}{M} \sum_{j=1}^{M} \left[R_j^{X_i,a} + \gamma V_k(Y_j^{X_i,a}) \right] \right|^p. \tag{6}$$

The *multi-sample variant* is obtained by using a fresh set of samples in each iteration:

$$V_{k+1} = \operatorname*{argmin}_{f\in\mathcal{F}} \sum_{i=1}^{N} \left| f(X_i^k) - \max_{a\in\mathcal{A}} \frac{1}{M} \sum_{j=1}^{M} \left[R_j^{X_i^k,a,k} + \gamma V_k(Y_j^{X_i^k,a,k}) \right] \right|^p. \tag{7}$$

Let π_k be a greedy policy w.r.t. V_k. We are interested in bounding the loss due to using policy π_k instead of an optimal one, where the loss is measured by a weighted p-norm:

$$L_k = \|V^* - V^{\pi_k}\|_{p,\rho}.$$

Here ρ is a distribution whose role is to put more weight on those parts of the state space where performance matters more. A particularly sensible choice is to set ρ to be the distribution over the

8. A mapping g between normed function spaces $(B_1, \|\cdot\|)$ and $(B_2, \|\cdot\|)$ is Lipschitz with factor $C > 0$ if $\forall x, y \in B_1$, $\|g(x) - g(y)\| \le \|x - y\|$.

states from which we start to use π_k. In this case if $p = 1$ then L_k measures the expected loss. For $p > 1$ the loss does not have a similarly simple interpretation, except that with $p \to \infty$ we recover the supremum-norm loss. Hence increasing p generally means that the evaluation becomes more pessimistic.

Let us now discuss how we arrive at a bound on the expected p-norm loss. By the results of the previous section we have a bound on the error introduced in any given iteration. Hence, all we need to show is that the errors do not blow up as they are propagated through the algorithm. Since the previous section's bounds are given in terms of weighted p-norms, it is natural to develop weighted p-norm bounds for the whole algorithm. Let us concentrate on the case when in all iterations the error committed is bounded. Since we use weighted p-norm bounds, the usual supremum-norm analysis does not work. However, a similar argument can be used.

The sketch of this argument is as follows: Since we are interested in developing a bound on the performance of the greedy policy w.r.t. the final estimate of V^*, we first develop a pointwise analogue of supremum-norm Bellman-error bounds:

$$(I - \gamma P^{\pi})(V^* - V^{\pi}) \leq \gamma (P^{\pi^*} - P^{\pi})(V^* - V).$$

Here V plays the role of the final value function estimate, π is a greedy policy w.r.t. V, and V^{π} is its value-function. Hence, we see that it suffices to develop upper and lower bounds on $V^* - V$ with $V = V_K$. For the upper estimate, we use that $V^* - TV_k = TV^* - TV_k = T^{\pi^*}V^* - T^{\pi_k}V_k \leq T^{\pi^*}V^* - T^{\pi^*}V_k = \gamma P^{\pi^*}(V^* - V_k)$. Hence, if $V_{k+1} = TV_k - \varepsilon_k$ then $V^* - V_{k+1} \leq \gamma P^{\pi^*}(V^* - V_k) + \varepsilon_k$. An analogous reasoning results in the lower bound $V^* - V_{k+1} \geq \gamma P^{\pi_k}(V^* - V_k) + \varepsilon_k$. Here π_k is a policy greedy w.r.t. V_k. Now, exploiting that the operator P^{π} is linear for any π, iterating these bounds yields upper and lower bounds on $V^* - V_K$ as a function of $\{\varepsilon_k\}_k$. A crucial step of the argument is to replace T, the non-linear Bellman operator by linear operators (P^{π}, for suitable π) since propagating errors through linear operators is easy, while in general, it is impossible to do the same with non-linear operators. Actually, as we propagate the errors, it is not hard to foresee that operator products of the form $P^{\pi_K}P^{\pi_{K-1}}\dots P^{\pi_k}$ enter our bounds and that the error amplification caused by these product operators is the major source of the possible increase of the error.

Note that if a supremum-norm analysis were followed ($p = \infty$), we would immediately find that the maximum amplification by these product operators is bounded by one: Since, as it is well known, for any policy π, $|\int V(y)P(dy|x,\pi(x))| \leq \int |V(y)|P(dy|x,\pi(x)) \leq \|V\|_\infty \int P(dy|x,\pi(x)) = \|V\|_\infty$, that is, $\|P^{\pi}\|_\infty \leq 1$. Hence

$$\|P^{\pi_K}\dots P^{\pi_k}\|_\infty \leq \|P^{\pi_K}\|_\infty \dots \|P^{\pi_k}\|_\infty \leq 1,$$

and starting from the pointwise bounds, one recovers the well-known supremum-norm bounds by just taking the supremum of the bounds' two sides. Hence, the pointwise bounding technique yields as tight bounds as the previous supremum-norm bounding technique. However, since in the algorithm only the weighted p-norm errors are controlled, instead of taking the pointwise supremum, we integrate the pointwise bounds w.r.t. the measure ρ to derive the desired p-norm bounds provided that the induced operator-norm of these operator products w.r.t. weighted p-norms can be bounded. One simple assumption that allows this is as follows:

Assumption A1 [Uniformly stochastic transitions] For all $x \in \mathcal{X}$ and $a \in \mathcal{A}$, assume that $P(\cdot|x,a)$ is absolutely continuous w.r.t. μ and the Radon-Nikodym derivative of P w.r.t. μ is bounded uniformly

with bound C_μ:

$$C_\mu \stackrel{\text{def}}{=} \sup_{x\in\mathcal{X},a\in\mathcal{A}} \left\| \frac{dP(\cdot|x,a)}{d\mu} \right\|_\infty < +\infty.$$

Assumption A1 can be written in the form $P(\cdot|x,a) \le C_\mu \mu(\cdot)$, an assumption that was introduced by Munos (2003) in a finite MDP context for the analysis of approximate policy iteration. Clearly, if Assumption A1 holds then for $p \ge 1$, by Jensen's inequality, $|\int V(y)P(dy|x,\pi(x))|^p \le \int |V(y)|^p P(dy|x,\pi(x)) \le \int C_\mu |V(y)|^p d\mu(dy)$, hence $\|P^\pi V\|_{p,\rho} \le C_\mu^{1/p} \|V\|_{p,\mu}$ and thus $\|P^\pi\|_{p,\rho} \le C_\mu^{1/p}$. Note that when μ is the Lebesgue-measure over $\mathcal{X}$ then Assumption A1 becomes equivalent to assuming that the transition probability kernel $P(dy|x,a)$ admits a uniformly bounded density. The noisier the dynamics, the smaller the constant C_μ. Although $C_\mu < +\infty$ looks like a strong restriction, the class of MDPs that admit this restriction is still quite large in the sense that there are hard instances in it (this is discussed in detail in Section 8). However, the above assumption certainly excludes completely or partially deterministic MDPs, which might be important, for example, in financial applications.

Let us now consider another assumption that allows for such systems, too. The idea is that for the analysis we only need to reason about the operator norms of weighted sums of the product of arbitrary stochastic kernels. This motivates the following assumption:

Assumption A2 [Discounted-average concentrability of future-state distributions] Given ρ, μ, $m \ge 1$ and an arbitrary sequence of stationary policies $\{\pi_m\}_{m\ge 1}$, assume that the future-state distribution $\rho P^{\pi_1} P^{\pi_2} \dots P^{\pi_m}$ is absolutely continuous w.r.t. μ. Assume that

$$c(m) \stackrel{\text{def}}{=} \sup_{\pi_1,\dots,\pi_m} \left\| \frac{d(\rho P^{\pi_1} P^{\pi_2} \dots P^{\pi_m})}{d\mu} \right\|_\infty \tag{8}$$

satisfies

$$C_{\rho,\mu} \stackrel{\text{def}}{=} (1-\gamma)^2 \sum_{m\ge 1} m\gamma^{m-1} c(m) < +\infty.$$

We shall call $c(m)$ the m-step concentrability of a future-state distribution, while we call $C_{\rho,\mu}$ the *discounted-average concentrability coefficient* of the future-state distributions.
The number $c(m)$ measures how much ρ can get amplified in m steps as compared to the reference distribution μ. Hence, in general we expect $c(m)$ to grow with m. In fact, the condition that $C_{\rho,\mu}$ is finite is a growth rate condition on $c(m)$. Thanks to discounting, $C_{\rho,\mu}$ is finite for a reasonably large class of systems: In fact, we will now argue that Assumption A2 is weaker than Assumption A1 and that $C_{\rho,\mu}$ is finite when the top-Lyapunov exponent of the MDP is finite.

To show the first statement it suffices to see that $c(m) \le C_\mu$ holds for any m. This holds since by definition for any distribution ν and policy π, $\nu P^\pi \le C_\mu \mu$. Then take $\nu = \rho P^{\pi_1} \dots P^{\pi_{m-1}}$ and $\pi = \pi_m$ to conclude that $\rho P^{\pi_1} \dots P^{\pi_{m-1}} P^{\pi_m} \le C_\mu \mu$ and so $c(m) \le C_\mu$.

Let us now turn to the comparison with the top-Lyapunov exponent of the MDP. As our starting point we take the definition of top-Lyapunov exponent associated with sequences of finite dimensional matrices: If $\{P_t\}_t$ is sequence of square matrices with non-negative entries and $\{y_t\}_t$ is a sequence of vectors that satisfy $y_{t+1} = P_t y_t$ then, by definition, the top-Lyapunov exponent is $\hat{\gamma}_{\text{top}} = \limsup_{t\to\infty} (1/t) \log^+(\|y_t\|_\infty)$. If the top-Lyapunov exponent is positive then the associated system is sensitive to its initial conditions (unstable). A negative top-Lyapunov exponent, on the other hand, indicates that the system is stable; in case of certain stochastic systems the existence

of strictly stationary non-anticipating realizations is equivalent to a negative Lyapunov exponent (Bougerol and Picard, 1992).[9]

Now, one may think of y_t as a probability distribution over the state space and the matrices as the transition kernels. One way to generalize the above definition to controlled systems and infinite state spaces is to identify y_t with the future state distribution when the policies are selected to maximize the growth rate of $\|y_t\|_\infty$. This gives rise to $\hat{\gamma}_{\text{top}} = \limsup_{m\to\infty} \frac{1}{m} \log c(m)$, where $c(m)$ is defined by (8).[10] Then, by elementary arguments, we get that if $\hat{\gamma}_{\text{top}} < \log(1/\gamma)$ then $\sum_{m\geq 0} m^p \gamma^m c(m) < \infty$. In fact, if $\hat{\gamma}_{\text{top}} \leq 0$ then $C(\rho,\nu) < \infty$. Hence, we interpret $C(\rho,\nu) < +\infty$ as a weak stability condition.

Since Assumption A1 is stronger than Assumption A2 in the proofs we will proceed by first developing a proof under Assumption A2. The reason Assumption A1 is still considered is that it will allow us to derive supremum-norm performance bounds even though in the algorithm we control only the weighted p-norm bounds.

As a final preparatory step before the presentation of our main results, let us define the *inherent Bellman error* associated with the function space $\mathcal{F}$ (as in the introduction) by

$$d_{p,\mu}(T\mathcal{F},\mathcal{F}) = \sup_{f\in\mathcal{F}} d_{p,\mu}(Tf,\mathcal{F}).$$

Note that $d_{p,\mu}(T\mathcal{F},\mathcal{F})$ generalizes the notion of Bellman errors to function spaces in a natural way: As we have seen the error in iteration k depends on $d_{p,\mu}(T\hat{V}_k,\mathcal{F})$. Since $\hat{V}_k \in \mathcal{F}$, the inherent Bellman error gives a uniform bound on the errors of the individual iterations.[11]

The next theorem is the main result of the paper. It states that with high probability the final performance of the policy found by the algorithm can be made as close to a constant times the inherent Bellman error of the function space $\mathcal{F}$ as desired by selecting a sufficiently high number of samples. Hence, sampling-based FVI can be used to find near-optimal policies if $\mathcal{F}$ is sufficiently rich:

Theorem 2 *Consider an MDP satisfying Assumption A0 and A2. Fix $p \geq 1$, $\mu \in M(\mathcal{X})$ and let $V_0 \in \mathcal{F} \subset B(\mathcal{X};V_{\max})$. Then for any $\varepsilon,\delta > 0$, there exist integers K,M and N such that K is linear in $\log(1/\varepsilon)$, $\log V_{\max}$ and $\log(1/(1-\gamma))$, N, M are polynomial in $1/\varepsilon$, $\log(1/\delta)$, $\log(1/(1-\gamma))$, $V_{\max}$, $\hat{R}_{\max}$, $\log(|\mathcal{A}|)$, $\log(\mathcal{N}(c\varepsilon(1-\gamma)^2/(C_{\rho,\mu}^{1/p}\gamma),\mathcal{F},N,\mu))$ for some constant $c > 0$, such that if the multi-sample variant of sampling-based FVI is run with parameters $(N,M,\mu,\mathcal{F})$ and π_K is a policy greedy w.r.t. the Kth iterate then w.p. at least $1-\delta$,*

$$\|V^* - V^{\pi_K}\|_{p,\rho} \leq \frac{2\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p}\, d_{p,\mu}(T\mathcal{F},\mathcal{F}) + \varepsilon.$$

If, instead of Assumption A2, Assumption A1 holds then w.p. at least $1-\delta$,

$$\|V^* - V^{\pi_K}\|_{\infty} \leq \frac{2\gamma}{(1-\gamma)^2} C_{\mu}^{1/p}\, d_{p,\mu}(T\mathcal{F},\mathcal{F}) + \varepsilon.$$

Further, the results continue to hold for the single-sample variant of sampling-based FVI with the exception that N depends on $\log(\mathcal{N}(c\varepsilon,\mathcal{F}_{T-},N,\mu))$ and M depends on $\log(\mathcal{N}(c'\varepsilon,\mathcal{F},M,\mu))$ for appropriate $c,c' > 0$.

9. The lack of existence of such solutions would probably preclude any sample-based estimation of the system.
10. Here we allow the sequence of policies to be changed with each m. It is an open question is a single sequence of policies would give the same result.
11. More generally, $d_{p,\mu}(\mathcal{G},\mathcal{F}) \stackrel{\text{def}}{=} \sup_{g\in\mathcal{G}} d_{p,\mu}(g,\mathcal{F}) \stackrel{\text{def}}{=} \sup_{g\in\mathcal{G}} \inf_{f\in\mathcal{F}} \|g-f\|_{p,\mu}$.

The proof is given in Appendix B. Assuming that the pseudo-dimension of the function-space $\mathcal{F}$ is finite as in Proposition 1, a close examination of the proof gives the following high-probability bound for the multi-sample variant:

$$\|V^* - V^{\pi_K}\|_{p,\rho} \leq \frac{2\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p} \, d_{p,\mu}(T\mathcal{F}, \mathcal{F}) + O(\gamma^K V_{\max})$$
$$+ \quad O\left\{ \left(\frac{V_{\mathcal{F}^+}}{N} (\log(N) + \log(K/\delta)) \right)^{1/2p} + \left(\frac{1}{M} (\log(N|\mathcal{A}|) + \log(K/\delta)) \right)^{1/2} \right\}. \qquad (9)$$

Here N, M, K are arbitrary integers and the bound holds w.p. $1-\delta$. The first term bounds the approximation error, the second arises due to the finite number of iterations, while the last two terms bound the estimation error.

This form of the bound allows us to reason about the likely best choice of N and M given a fixed budget of $n = K \times N \times M$ samples (or $\hat{n} = N \times M$ samples per iteration). Indeed, optimizing the bound yields that the best choice of N and M (apart from constants) is given by $N = (V_{\mathcal{F}^+})^{1/(p+1)} \hat{n}^{p/(p+1)}$, $M = (\hat{n}/V_{\mathcal{F}^+})^{1/(p+1)}$, resulting in the bound $(n/(KV_{\mathcal{F}^+}))^{-1/(2p+2)}$ for the estimation error, disregarding logarithmic terms. Note that the choice of N, M does not influence the other error terms.

Now, let us consider the single-sample variant of FVI. A careful inspection of the proof results in an inequality identical to (9) just with the pseudo-dimension of $\mathcal{F}$ replaced by the pseudo-dimension of $\mathcal{F}^-$ and $1/M$ replaced by $V_{\mathcal{F}^+}/M$. We may again ask the question of how to choose N, M, given a fixed-size budget of $n = N \times M$ samples. The formulae are similar to the previous ones. The resulting optimized bound on the estimation error is $(n/(V_{\mathcal{F}^-} V_{\mathcal{F}^+}))^{-1/(2p+2)}$. It follows that given a fixed budget of n samples provided that $K > V_{\mathcal{F}^-}$ the bound for the single-sample variant is better than the one for the multi-sample variant. In both cases a logical choice is to set K to minimize the respective bounds. In fact, the optimal choice turns out to be $K \sim 1/\log(1/\gamma) \approxeq 1/(1-\gamma)$ in both cases. Hence as γ approaches one, the single-sample variant of FVI can be expected to become more efficient, provided that everything else is kept the same. It is interesting to note that as γ becomes larger the number of times the samples are reused increases, too. That the single-sample variant becomes more efficient is because the variance reduction effect of sample reuse is stronger than the increase of the bias. Our computer simulations (Section 9) confirm this experimentally.

Another way to use the above bound is to make comparisons with the rates available in non-parametric regression: First, notice that the approximation error of $\mathcal{F}$ is defined as the inherent Bellman error of $\mathcal{F}$ instead of using an external reference class. This seems reasonable since we are trying to find an approximate fixed point of T within $\mathcal{F}$. The estimation error, for a sample size of n, can be seen to be bounded by $O(n^{-1/(2(p+1))})$, which for $p = 2$ gives $O(n^{-1/6})$. In regression, the comparable error (when using a bounding technique similar to ours) is bounded by $n^{-1/4}$ (Györfi et al., 2002). With considerably more work, using the techniques of Lee et al. (1996) (see also Chapter 11 of Györfi et al., 2002) in regression it is possible to get a rate of $n^{-1/2}$, at the price of multiplying the approximation error by a constant larger than one. It seems possible to use these techniques to improve the exponent of N from $-1/2p$ to $-1/p$ in Equation (9) (at the price of increasing the influence of the approximation error). Then the new rate would become $n^{-1/4}$. This is still worse than the best possible rate for non-parametric regression. The additional factor comes from the need to use the samples to control the bias of the target values (i.e., that we need $M \to \infty$). Thus, in the case of FVI, the inferior rate as compared with regression seems unavoidable. By

switching from state value functions to action-value functions it seems quite possible to eliminate this inefficiency. In this case the capacity of the function space would increase (in particular, in Equation (9) $V_{\mathcal{F}^+}$ would be replaced by $|\mathcal{A}|V_{\mathcal{F}^+}$).

6. Randomized Policies

The previous result shows that by making the inherent Bellman error of the function space small enough, we can ensure a close-to-optimal performance if one uses a policy greedy w.r.t. the last value-function estimate, V_K. However, the computation of such a greedy policy requires the evaluation of some expectations, whose exact values are however often difficult to compute. In this section we show that by computations analogous to that used in obtaining the iterates we can compute a randomized near-optimal policy based on V_K.

Let us call an action a α-*greedy* w.r.t. the function V and state x, if

$$r(x,a)+\gamma\int V(y)P(dy|x,a)\geq (TV)(x)-\alpha.$$

Given V_K and a state $x\in\mathcal{X}$ we can use sampling to draw an α-greedy action w.p. at least $1-\lambda$ by executing the following procedure: Let $R_j^{x,a}\sim S(\cdot,x,a)$, $Y_j^{x,a}\sim P(\cdot|x,a)$, $j=1,2,\ldots,M'$ with $M'=M'(\alpha,\lambda)$ and compute the approximate value of a at state x using

$$Q_{M'}(x,a)=\frac{1}{M'}\sum_{j=1}^{M'}\left[R_j^{x,a}+\gamma V_K(Y_j^{x,a})\right].$$

Let the policy $\pi_{\alpha,\lambda}^K:\mathcal{X}\to\mathcal{A}$ be defined by

$$\pi_{\alpha,\lambda}^K(x)=\arg\max_{a\in\mathcal{A}}Q_{M'}(x,a).$$

The following result holds:

Theorem 3 *Consider an MDP satisfying Assumptions A0 and A2. Fix $p\geq 1$, $\mu\in M(\mathcal{X})$ and let $V_0\in\mathcal{F}\subset B(\mathcal{X};V_{\max})$. Select $\alpha=(1-\gamma)\varepsilon/8$, $\lambda=\frac{\varepsilon}{8}\frac{(1-\gamma)}{V_{\max}}$ and let $M'=O(|\mathcal{A}|\hat{R}_{\max}^2\log(|\mathcal{A}|/\lambda)/\alpha^2)$. Then, for any $\varepsilon,\delta>0$, there exist integers K,M and N such that K is linear in $\log(1/\varepsilon)$, $\log V_{\max}$ and $\log(1/(1-\gamma))$, N,M are polynomial in $1/\varepsilon$, $\log(1/\delta)$, $1/(1-\gamma)$, $V_{\max}$, $\hat{R}_{\max}$, $\log(|\mathcal{A}|)$, $\log(\mathcal{N}(c\varepsilon(1-\gamma)^2/C_{\rho,\mu}^{1/p}),\mathcal{F},\mu)))$ for some $c>0$, such that if $\{V_k\}_{k=1}^K$ are the iterates generated by multi-sample FVI with parameters $(N,M,K,\mu,\mathcal{F})$ then for the policy $\pi_{\alpha,\lambda}^K$ as defined above, w.p. at least $1-\delta$, we have*

$$\left\|V^*-V^{\pi_{\alpha,\lambda}^K}\right\|_{p,\mu}\leq\frac{4\gamma}{(1-\gamma)^2}C_{\rho,\mu}^{1/p}d_{p,\mu}(T\mathcal{F},\mathcal{F})+\varepsilon.$$

An analogous result holds for the supremum-norm loss under Assumptions A0 and A1 with $C_{\rho,\mu}$ replaced by C_μ.

The proof can be found in Appendix C.

A similar result holds for the single-sample variant of FVI. We note that in place of the above uniform sampling model one could also use the Median Elimination Algorithm of Even-Dar et al. (2002), resulting in a reduction of M' by a factor of $\log(|\mathcal{A}|)$. However, for the sake of compactness we do not explore this option here.

7. Asymptotic Consistency

A highly desirable property of any learning algorithm is that as the number of samples grows to infinity, the error of the algorithm should converge to zero; in other words, the algorithm should be *consistent*. Sampling based FVI with a *fixed* function space $\mathcal{F}$ is not consistent: Our previous results show that in such a case the loss converges to $\frac{2\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p} d_{p,\mu}(T\mathcal{F},\mathcal{F})$. A simple idea to remedy this situation is to let the function space grow with the number of samples. In regression the corresponding method was proposed by Grendander (1981) and is called the *method of sieves*. The purpose of this section is to show that FVI combined with this method gives a consistent algorithm for a large class of MDPs, namely for those that have Lipschitzian rewards and transitions. It is important to emphasize that although the results in this section assume these smoothness conditions, the method itself does not require the knowledge of the smoothness factors. It is left for future work to determine whether similar results hold for larger classes of MDPs.

The smoothness of the transition probabilities and rewards is defined w.r.t. changes in the initial state: $\forall (x,x',a) \in \mathcal{X}\times\mathcal{X}\times\mathcal{A}$,

$$\begin{aligned} \|P(\cdot|x,a) - P(\cdot|x',a)\| &\leq L_P \left\|x - x'\right\|^{\alpha}, \\ |r(x,a) - r(x',a)| &\leq L_r \left\|x - x'\right\|^{\alpha}. \end{aligned}$$

Here $\alpha, L_P, L_r > 0$ are the unknown smoothness parameters of the MDP and $\|P(\cdot|x,a) - P(\cdot|x',a)\|$ denotes the total variation norm of the signed measure $P(\cdot|x,a) - P(\cdot|x',a)$.[12]

The method is built on the following observation: If the MDP is smooth in the above sense and if $V \in B(\mathcal{X})$ is uniformly bounded by $V_{\max}$ then TV is $L = (L_r + \gamma V_{\max} L_P)$-Lipschitzian (with exponent $0 < \alpha \leq 1$):

$$|(TV)(x) - (TV)(x')| \leq (L_r + \gamma V_{\max} L_P)\|x - x'\|^{\alpha}, \quad \forall x, x' \in \mathcal{X}.$$

Hence, if $\mathcal{F}_n$ is restricted to $V_{\max}$-bounded functions then $T\mathcal{F}_n \stackrel{\text{def}}{=} \{TV \,|\, V \in \mathcal{F}_n\}$ contains L-Lipschitz $V_{\max}$-bounded functions only:

$$T\mathcal{F}_n \subset \mathrm{Lip}(\alpha; L, V_{\max}) \stackrel{\text{def}}{=} \{ f \in B(\mathcal{X}) \,|\, \quad \|f\|_\infty \leq V_{\max}, |f(x) - f(y)| \leq L\|x - y\|^{\alpha} \}.$$

By the definition of $d_{p,\mu}$,

$$d_{p,\mu}(T\mathcal{F}_n, \mathcal{F}_n) \leq d_{p,\mu}(\mathrm{Lip}(\alpha; L, V_{\max}), \mathcal{F}_n).$$

Hence if we make the right-hand side converge to zero as $n \to \infty$ then so will do $d_{p,\mu}(T\mathcal{F}_n, \mathcal{F}_n)$. The quantity, $d_{p,\mu}(\mathrm{Lip}(\alpha; L, V_{\max}), \mathcal{F}_n)$ is nothing but the approximation error of functions in the Lipschitz class $\mathrm{Lip}(\alpha; L, V_{\max})$ by elements of $\mathcal{F}_n$. Now, $d_{p,\mu}(\mathrm{Lip}(\alpha; L, V_{\max}), \mathcal{F}_n) \leq d_{p,\mu}(\mathrm{Lip}(\alpha; L), \mathcal{F}_n)$, where $\mathrm{Lip}(\alpha; L)$ is the set of Lipschitz-functions with Lipschitz constant L and we exploited that $\mathrm{Lip}(\alpha, L) = \cup_{V_{\max}>0} \mathrm{Lip}(\alpha; L, V_{\max})$. In approximation theory an approximation class $\{\mathcal{F}_n\}$ is said to be *universal* if for any $\alpha, L > 0$,

$$\lim_{n\to\infty} d_{p,\mu}(\mathrm{Lip}(\alpha; L), \mathcal{F}_n) = 0.$$

12. Let μ be a signed measure over $\mathcal{X}$. Then the total variation measure, $|\mu|$ of μ is defined by $|\mu|(B) = \sup \sum_{i=1}^{\infty} |\mu(B_i)|$, where the supremum is taken over all at most countable partitions of B into pairwise disjoint parts from the Borel sets over $\mathcal{X}$. The total variation norm $\|\mu\|$ of μ is $\|\mu\| = |\mu|(\mathcal{X})$.

For a large variety of approximation classes (e.g., approximation by polynomials, Fourier basis, wavelets, function dictionaries) not only universality is established, but variants of Jackson's theorem give us rates of convergence of the approximation error: $d_{p,\mu}(\mathrm{Lip}(\alpha;L),\mathcal{F}_n) = O(Ln^{-\alpha})$ (e.g., DeVore, 1997).

One remaining issue is that classical approximation spaces are not uniformly bounded (i.e., the functions in them do not assume a uniform bound), while our previous argument showing that the image space $T\mathcal{F}_n$ is a subset of Lipschitz functions critically relies on that $\mathcal{F}_n$ is uniformly bounded. One solution is to use truncations: Let $\mathcal{T}_{V_{\max}}$ be the truncation operator,

$$\mathcal{T}_{V_{\max}} r = \begin{cases} \mathrm{sign}(r) V_{\max}, & \text{if } |r| > V_{\max}, \\ r, & \text{otherwise.} \end{cases}$$

Now, a simple calculation shows that

$$d_{p,\mu}(\mathrm{Lip}(\alpha;L) \cap B(\mathcal{X};V_{\max}), \mathcal{T}_{V_{\max}}\mathcal{F}_n) \le d_{p,\mu}(\mathrm{Lip}(\alpha;L),\mathcal{F}_n),$$

where $\mathcal{T}_{V_{\max}}\mathcal{F}_n = \{\mathcal{T}_{V_{\max}} f \mid f \in \mathcal{F}_n\}$. This, together with Theorem 2 gives rise to the following result:

Corollary 4 *Consider an MDP satisfying Assumptions A0 and A2 and assume that both its immediate reward function and transition kernel are Lipschitzian. Fix $p \ge 1$, $\mu \in M(\mathcal{X})$ and let $\{\mathcal{F}_n\}$, be a universal approximation class such that the pseudo-dimension of $\mathcal{T}_{V_{\max}}\mathcal{F}_n$ grows sublinearly in n. Then, for each $\varepsilon,\delta > 0$ there exist an index n_0 such that for any $n \ge n_0$ there exist integers K,N,M that are polynomial in $1/\varepsilon$,$\log(1/\delta)$, $1/(1-\gamma)$, $V_{\max}$, $\hat{R}_{\max}$, $\log(|\mathcal{A}|)$, and $V_{(\mathcal{T}_{V_{\max}}\mathcal{F}_n)^+}$ such that if V_K is the output of multi-sample FVI when it uses the function set $\mathcal{T}_{V_{\max}}\mathcal{F}_n$ and $X_i \sim \mu$ then $\|V^* - V^{\pi_K}\|_{p,\rho} \le \varepsilon$ holds w.p. at least $1-\delta$. An identical result holds for $\|V^* - V^{\pi_K}\|_\infty$ when Assumption A2 is replaced by Assumption A1.*

The result extends to single-sample FVI as before.

One aspect in which this corollary is not satisfactory is that solving the optimization problem defined by Equation (1) over $\mathcal{T}_{V_{\max}}\mathcal{F}_n$ is computationally challenging even when $\mathcal{F}_n$ is a class of linearly parameterized functions and $p = 2$. One idea is to do the optimization first over $\mathcal{F}_n$ and then truncate the obtained functions. The resulting procedure can be shown to be consistent (cf., Chapter 10 of Györfi et al., 2002, for an alike result in a regression setting).

It is important to emphasize that the construction used in this section is just one example of how our main result may lead to consistent algorithms. An immediate extension of the present work would be to target the best possible convergence rates for a given MDP by using penalized estimation. We leave the study of such methods for future work.

8. Discussion of Related Work

Sampling based FVI has roots that date back to the early days of dynamic programming. One of the first examples of using value-function approximation methods is the work of Samuel who used both linear and non-linear methods to approximate value functions in his programs that learned to play the game of checkers (Samuel, 1959, 1967). At the same time, Bellman and Dreyfus (1959) explored the use of polynomials for accelerating dynamic programming. Both in these works and also in most later works (e.g., Reetz, 1977; Morin, 1978) FVI with representative states was considered.

Of these authors, only Reetz (1977) presents theoretical results who, on the other hand, considered only one-dimensional feature spaces.

FVI is a special case of *approximate value iteration* (AVI) which encompasses any algorithm of the form $V_{t+1} = TV_t + \varepsilon_t$, where the errors ε_t are controlled in some way. If the error terms, ε_t, are bounded in supremum norm, then a straightforward analysis shows that asymptotically, the worst-case performance-loss for the policy greedy w.r.t. the most recent iterates can be bounded by $\frac{2\gamma}{(1-\gamma)^2} \sup_{t\geq 1} \|\varepsilon_t\|_\infty$ (e.g., Bertsekas and Tsitsiklis, 1996). When V_{t+1} is the best approximation of TV_t in $\mathcal{F}$ then $\sup_{t\geq 1} \|\varepsilon_t\|_\infty$ can be upper bounded by the inherent Bellman error $d_\infty(T\mathcal{F}, \mathcal{F}) = \sup_{f\in\mathcal{F}} \inf_{g\in\mathcal{F}} \|g - Tf\|_\infty$ and we get the loss-bound $\frac{2\gamma}{(1-\gamma)^2} d_\infty(T\mathcal{F}, \mathcal{F})$. Apart from the smoothness factors ($C_{\rho,\mu}$, C_μ) and the estimation error term, our loss-bounds have the same form (cf., Equation 9). In particular, if μ is absolutely continuous w.r.t. the Lebesgue measure then letting $p \to \infty$ allows us to recover these previous bounds (since then $C_\mu^{1/p} d_{p,\mu}(T\mathcal{F}, \mathcal{F}) \to d_\infty(T\mathcal{F}, \mathcal{F})$). Further, we expect that the p-norm bounds would be tighter since the supremum norm is sensitive to outliers.

A different analysis, originally proposed by Gordon (1995) and Tsitsiklis and Van Roy (1996), goes by assuming that the iterates satisfy $V_{t+1} = \Pi TV_t$, where Π is an operator that maps bounded functions to the function space $\mathcal{F}$. While Gordon (1995) and Tsitsiklis and Van Roy (1996) considered the planning scenario with known dynamics and making use of a set of representative states, subsequent results by Singh et al. (1995), Ormoneit and Sen (2002) and Szepesvári and Smart (2004) considered less restricted problem settings, though none of these authors presented finite-sample bounds. The main idea in these analyses is that the above iterates must converge to some limit V_∞ if the composite operator ΠT is a supremum-norm contraction. Since T is a contraction, this holds whenever Π is a supremum-norm non-expansion. In this case, the loss of using the policy greedy w.r.t. V_∞ can be bounded by $\frac{4\gamma}{(1-\gamma)^2}\varepsilon_\Pi$, where ε_Π is the best approximation to V^* by fixed points of Π: $\varepsilon_\Pi = \inf_{f\in\mathcal{F}:\Pi f=f} \|f - V^*\|_\infty$ (e.g., Tsitsiklis and Van Roy, 1996, Theorem 2).

In practice a special class of approximation methods called *averagers* are used (Gordon, 1995). For these methods Π is guaranteed to be a non-expansion. Kernel regression methods, such as k-nearest neighbors smoothing with fixed centers, tree based smoothing (Ernst et al., 2005), or linear interpolation with a fixed set of basis functions such as spline interpolation with fixed knots all belong to this class. In all these examples Π is a linear operator and takes the form $\Pi f = \alpha + \sum_{i=1}^n (L_i f)\phi_i$ with some function α, appropriate basis functions, ϕ_i, and linear functionals L_i ($i = 1, 2, \ldots, n$). One particularly interesting case is when $L_i f = f(x_i)$ for some points $\{x_i\}$, $\phi_0 \geq 0$, $\sum_i \phi_i \equiv 1$, $\alpha \equiv 0$, $(\Pi f)(x_i) = f(x_i)$ and $(\phi_i(x_j))_{ij}$ has full rank. In this case all members of the space spanned by the basis functions $\{\phi_i\}$ are fixed points of Π. Hence $\varepsilon_\Pi = d_\infty(\mathrm{span}(\phi_1, \ldots, \phi_n), V^*)$ and so the loss of the procedure is directly controlled by the size of $\mathcal{F}_n = \mathrm{span}(\phi_1, \ldots, \phi_n)$.

Let us now discuss the choice of the function spaces in averagers and sampling-based FVI. In the case of averagers, the class is restricted, but the approximation requirement, making ε_Π small, seems to be easier to satisfy than the corresponding requirement which asks for making the inherent Bellman residual of the function space $\mathcal{F}_n$ small. We think that in the lack of knowledge of V^* this advantage might be minor and can be offset by the larger freedom to choose $\mathcal{F}_n$ (i.e., nonlinear, or kernel-based methods are allowed). In fact, when V^* is unknown one must resort to the generic properties of the class of MDPs considered (e.g., smoothness) in order to find the appropriate function space. Since the optimal policy is unknown, too, it is not quite immediate that the fact that only a single function (that depends on an unknown MDP) must be well approximated should be an advantage. Still, one may argue that the self-referential nature of the inherent Bellman-

error makes the design for sampling-based FVI harder. As we have shown in Section 7, provided that the MDPs are smooth, designing these spaces is not necessarily harder than designing a function approximator for some regression task.

Let us now discuss some other related works where the authors consider the error resulting from some Monte-Carlo procedure. One set of results closely related to the ones presented here is due to Tsitsiklis and Roy (2001). These authors studied sampling-based fitted value iteration with linear function approximators. However, they considered a different class of MDPs: finite horizon, optimal stopping with discounted total rewards. In this setting the next-state distribution under the condition of not stopping is uncontrolled—the state of the market evolves independently of the decision maker. Tsitsiklis and Roy (2001) argue that in this case it is better to sample full trajectories than to generate samples in some other, arbitrary way. Their algorithm implements approximate backward propagation of the values (by L^2 fitting with linear function approximators), exploiting that the problem has a fixed, finite horizon. Their main result shows that the estimation error converges to zero w.p. 1 as the number of samples grows to infinity. Further, a bound on the asymptotic performance is given. Due to the special structure of the problem, this bound depends only on how well the optimal value function is approximated by the chosen function space. Certainly, because of the known counterexamples (Baird, 1995; Tsitsiklis and Van Roy, 1996), we cannot hope such a bound to hold in the general case.

The work presented here builds on our previous work. For finite state-space MDPs, Munos (2003, 2005) considered planning scenarios with known dynamics analyzing the stability of both approximate policy iteration and value iteration with weighted L^2 (resp., L^p) norms. Preliminary versions of the results presented here were published in Szepesvári and Munos (2005). Using techniques similar to those developed here, recently we have proved results for the learning scenario when only a single trajectory of some fixed behavior policy is known (Antos et al., 2006). We know of no other work that would have considered the weighted p-norm error analysis of sampling-based FVI for continuous state-space MDPs and in a discounted, infinite-horizon settings.

One work where the author studies fitted value iteration and which comes with a finite-sample analysis is by Murphy (2005), who, just like Tsitsiklis and Roy (2001), studied finite horizon problems with no discounting.[13] Because of the finite-horizon setting, the analysis is considerable simpler (the algorithm works backwards). The samples come from a number of independent trajectories just like in the case of Tsitsiklis and Roy (2001). The error bounds come in the form of performance differences between a pair of greedy policies: One of the policies from the pair is greedy w.r.t. the value function returned by the algorithm, while the other is greedy w.r.t. to some arbitrary 'test' function from the function set considered in the algorithm. The derived bound shows that the number of samples needed is exponential in the horizon of the problem and is proportional to ε^{-4}, where ε is the desired estimation error. The approximation error of the procedure, however, is not considered: Murphy suggests that the optimal action-value function could be added at virtually no cost to the function sets used by the algorithm. Accordingly, her bounds scale only with the complexity of the function class and do not scale directly with the dimensionality of the state space (just through

13. We learnt of the results of Murphy (2005) after submitting our paper. One interesting aspect of this paper is that the results are presented for partially observable problems. However, since all value-function approximation methods introduce state aliasing anyway, results worked out for the fully observable case carry through to the limited feedback case without any change except that the approximation power of the function approximation method is further limited by the information that is fed into the approximator. Based on this observation one may wonder if it is possible to get consistent algorithms that avoid an explicit 'state estimation' component. However, this remains the subject of future work.

the complexity of the function class). One interpretation of this is that if we are lucky to choose a function approximator so that the optimal value function (at all stages) can be represented exactly with it then the rate of convergence can be fast. In the unlucky case, no bound is given. We will come back to the discussion of worst-case sample complexity after discussing the work by Kakade and Langford (Kakade and Langford, 2002; Kakade, 2003).

The algorithm considered by Kakade and Langford is called conservative policy iteration (CPI). The algorithm is designed for discounted infinite horizon problems. The general version searches in a fixed policy space, Π, in each step an optimizer picking a policy that maximizes the average of the empirical advantages of the previous policy at a number of states (basepoints) sampled from some distribution. These advantages could be estimated by sampling sufficiently long trajectories from the basepoints. The policy picked this way is mixed into the previous policy to prevent performance drops due to drastic changes, hence the name of the algorithm.

Theorems 7.3.1 and 7.3.3 Kakade (2003) give bounds on the loss of using the policy returned by this procedure relative to using some other policy π (e.g., a near-optimal policy) as a function of the total variation distance between ν, the distribution used to sample the basepoints (this distribution is provided by the user), and the discounted future-state distribution underlying π when π is started from a random state sampled from ν ($d_{\pi,\nu}$). Thus, unlike in the present paper the error of the procedure can only be controlled by finding a distribution that minimizes the distance to $d_{\pi,\gamma}$, where π is a near-optimal policy. This might be as difficult as the problem of finding a good policy. Theorem 6.2 in Kakade and Langford (2002) bounds the expected performance loss under ν as a function of the imprecision of the optimizer and the Radon-Nykodim derivative of $d_{\pi^*,\nu}$ and $d_{\pi_0,\nu}$, where π_0 is the policy returned by the algorithm. However this result applies only to the case when the policy set is unrestricted, and hence the result is limited to finite MDPs.

Now let us discuss the worst-case sample complexity of solving MDPs. A very simple observation is that it should be impossible to get bounds that scale polynomially with the dimension of the state-space unless special conditions are made on the problem. This is because the problem of estimating the value function of a policy in a trivial finite-horizon problem with a single time step is equivalent to regression. Hence known lower bounds for regression must apply to RL, as well (see Stone, 1980, 1982 and Chapter 3 of Györfi, Kohler, Krzyżak, and Walk, 2002 for such bounds). In particular, from these bounds it follows that the minimax sample complexity of RL is exponential in the dimensionality of the state space provided the class of MDPs is large enough. Hence, it is not surprising that unless very special conditions are imposed on the class of MDPs considered, FVI and its variants are subject to the curse-of-dimensionality. One way to help with this exponential scaling is when the algorithm is capable of taking advantage of the possible advantageous properties of the MDP to be solved. In our opinion, one major open problem in RL is to design such methods (or to show that some existing method possesses this property).

The curse-of-dimensionality is not specific to FVI variants. In fact, a result of Chow and Tsitsiklis (1989) states the following: Consider a class of MDPs with $\mathcal{X} = [0,1]^d$. Assume that for any MDP in the class, the transition probability kernel underlying the MDP has a density w.r.t. the Lebesgue measure and these densities have a common upper bound. Further, the MDPs within the class are assumed to be uniformly smooth: for any MDP in the class the Lipschitz constant of the reward function of the MDP is bounded by an appropriate constant and the same holds for the Lipschitz constant of the density function. Fix a desired precision, ε. Then, any algorithm that is guaranteed to return an ε-optimal approximation to the optimal value function must query (sample) the reward function and the transition probabilities at least $\Omega(1/\varepsilon^d)$-times, for some MDP within the

class considered. Hence, even classes of smooth MDPs with uniformly bounded transition densities have very hard instances.

The situation changes dramatically if one is allowed to interleave computations and control. In this case, building on the *random-discretization method* of Rust (1996b), it is possible to achieve near-optimal behavior by using a number of samples per step that scales polynomially in the important quantities (Szepesvári, 2001). In particular, this result shows that it suffices to let the number of samples scale linearly in the dimensionality of the state space. Interestingly, this result holds for a class of MDPs that subsumes the one considered by Chow and Tsitsiklis (1989). The random-discretization method requires that the MDPs in the class satisfy Assumption A1 with a common constant and also the knowledge of the density underlying the transition probability kernel. When the density does not exist or is not known, it could be estimated. However, estimating conditional density functions itself is also subject to the curse of dimensionality, hence, the advantage of the random-discretization method melts away in such situations, making sampling-based FVI a viable alternative. This is the case indeed, since the results presented here require weaker assumptions on the transition probability kernel (Assumption A2) and thus apply to a broader class of problems.

Another method that interleaves computations and control is the sparse trajectory-tree method of Kearns et al. (1999). The sparse trajectory-tree method builds a random lookahead tree to compute sample based approximation to the values of each of the actions available at the current state. This method does not require the knowledge of the density underlying the transition probability kernel, nor does it require any assumptions on the MDP. Unfortunately, the computational cost of this method scales exponentially in the 'ε-horizon', $\log_\gamma(R_{\max}/(\varepsilon(1-\gamma)))$. This puts severe limits on the utility of this method when the discount factor is close to one and the number of actions is moderate. Kearns et al. (1999) argue that without imposing additional assumptions (i.e., smoothness) on the MDP the exponential dependency on the effective horizon time is unavoidable (a similar dependence on the horizon shows up in the bounds of Murphy, 2005 and Kakade, 2003).

9. Simulation Study

The purpose of this section is to illustrate the tradeoffs involved in using FVI. Since identical or very similar algorithms have been used successfully in many prior empirical studies (e.g., Longstaff and Shwartz, 2001; Haugh, 2003; Jung and Uthmann, 2004), we do not attempt a thorough empirical evaluation of the algorithm.

9.1 An Optimal Replacement Problem

The problem used as a testbed is a simple one-dimensional optimal replacement problem, described for example by Rust (1996a). The system has a one-dimensional state. The state variable, $x_t \in \mathbb{R}_+$, measures the accumulated utilization of a product, such as the odometer reading on a car. By convention, we let $x_t = 0$ denote a brand new product. At each time step, t, there are two possible decisions: either keep ($a_t = \mathbf{K}$) or replace ($a_t = \mathbf{R}$) the product. This latter action implies an additional cost C of selling the existing product and replacing it by a new one. The transition to

a new state occurs with the following exponential densities:

$$p(y|x,\mathbf{K}) = \begin{cases} \beta e^{-\beta(y-x)}, & \text{if } y \geq x; \\ 0, & \text{if } y < x, \end{cases}$$
$$p(y|x,\mathbf{R}) = \begin{cases} \beta e^{-\beta y}, & \text{if } y \geq 0; \\ 0, & \text{if } y < 0. \end{cases}$$

The reward function is $r(x,\mathbf{K}) = -c(x)$, where $c(x)$ represents the cost of maintaining the product. By assumptions, c is monotonically increasing. The reward associated with the replacement of the product is independent of the state and is given by $r(x,\mathbf{R}) = -C - c(0)$.

The optimal value function solves the Bellman optimality equation:

$$V^*(x) = \max\Big[-c(x) + \gamma \int_x^\infty p(y|x,\mathbf{K})V^*(y)dy, -C - c(0) + \gamma \int_0^\infty p(y|x,\mathbf{R})V^*(y)dy\Big].$$

Here the first argument of max represents the total future reward given that the product is not replaced, while the second argument gives the total future reward provided that the product is replaced. This equation has a closed form solution:

$$V^*(x) = \begin{cases} \int_x^{\bar{x}} \frac{c'(y)}{1-\gamma}(1-\gamma e^{-\beta(1-\gamma)(y-x)})dy - \frac{c(\bar{x})}{1-\gamma}, & \text{if} \quad x \leq \bar{x}; \\ \frac{-c(\bar{x})}{1-\gamma}, & \text{if} \quad x > \bar{x}, \end{cases}$$

Here $\bar{x}$ is the unique solution to

$$C = \int_0^{\bar{x}} \frac{c'(y)}{1-\gamma}(1-\gamma e^{-\beta(1-\gamma)y})dy.$$

The optimal policy is $\pi^*(x) = \mathbf{K}$ if $x \in [0,\bar{x}]$, and $\pi^*(x) = \mathbf{R}$ if $x > \bar{x}$.

9.2 Results

We chose the numerical values $\gamma = 0.6$, $\beta = 0.5$, $C = 30$, $c(x) = 4x$. This gives $\bar{x} \simeq 4.8665$ and the optimal value function, plotted in Figure 2, is

$$V^*(x) = \begin{cases} -10x + 30(e^{0.2(x-\bar{x})} - 1), & \text{if} \quad x \leq \bar{x}; \\ -10\bar{x}, & \text{if} \quad x > \bar{x}. \end{cases}$$

We consider approximation of the value function using polynomials of degree l. As suggested in Section 7, we used truncation. In order to make the state space bounded, we actually consider a problem that closely approximates the original one. For this we fix an upper bound for the states, $x_{\max} = 10 \gg \bar{x}$, and modify the problem definition such that if the next state y happens to be outside of the domain $[0, x_{\max}]$ then the product is replaced immediately, and a new state is drawn as if action $\mathbf{R}$ were chosen in the previous time step. By the choice of $x_{\max}$, $\int_{x_{\max}}^\infty p(dy|x,\mathbf{R})$ is negligible and hence the optimal value function of the altered problem closely matches that of the original problem when it is restricted to $[0, x_{\max}]$.

We chose the distribution μ to be uniform over the state space $[0, x_{\max}]$. The transition density functions $p(\cdot|x,a)$ are bounded by β, thus Assumption A1 holds with $C_\mu = \beta x_{\max} = 5$.

Figure 2 illustrates two iterates ($k = 2$ and $k = K = 20$) of the multi-sample version of sampling-based FVI: the dots represents the points $\{(X_n, \hat{V}_{M,k+1}(X_n))\}_{1 \leq n \leq N}$ for $N = 100$, where X_i is drawn

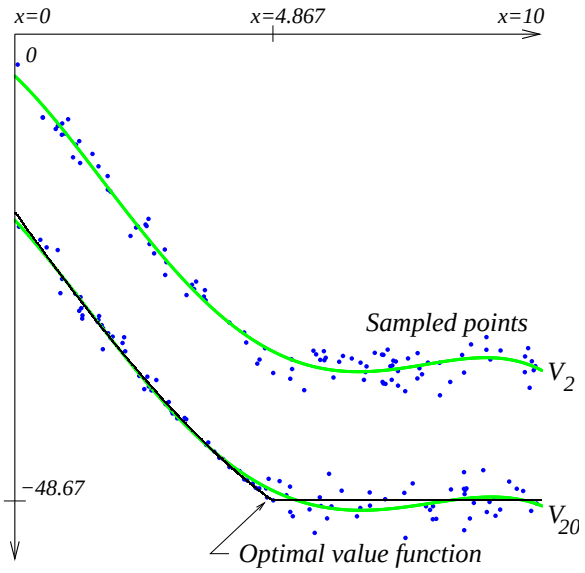


Figure 2: Illustration of two iteration steps of Sampling based FVI (up: $k = 2$, down: $k = 20$). The dots represent the pairs $(X_i, \hat{V}_{M,k}(X_i))$, $i = 1,\ldots,N$ based on $M = 10$ sampled transitions per basepoint and $N = 100$ basepoints. The grey curve is the best fit among polynomials of degree $l = 4$. The thin black curve is the optimal value function.

from μ and $\{\hat{V}_{M,k+1}(X_n)\}_{1\le n\le N}$ is computed using (2) with $V = V_k$ and $M = 10$ samples. The grey curve is the best fit (minimizing the least square error to the data, that is, $p = 2$) in $\mathcal{F}$ (for $l = 4$) and the thin black curve is the optimal value function.

Figure 3 shows the L_∞ approximation errors $||V^* - V_K||_\infty$ for different values of the degree l of the polynomial regression, and different values of the number of basepoints N and the number of sampled next states M. The number of iterations was set to $K = 20$. The reason that the figure shows the error of approximating V^* by V_K (i.e., $\varepsilon_K = \|V^* - V_K\|_\infty$) instead of $\|V^* - V^{\pi_K}\|_\infty$ is that in this problem this latter error converges very fast and thus is less interesting. (The technique developed in the paper can be readily used to derive a bound on the estimation error of V^*.) Of course, the performance loss is always upper bounded (in L_∞-norm) by the approximation error, thanks to the well-known bound (e.g., Bertsekas and Tsitsiklis, 1996): $||V^* - V^{\pi_K}||_\infty \le 2/(1-\gamma)||V^* - V_K||_\infty$.

From Figure 3 we observe when the degree l of the polynomials increases, the error decreases first because of the decrease of the inherent approximation error, but eventually increases because of overfitting. This graph thus illustrates the different components of the bound (9) where the approximation error term $d_{p,\mu}(T\mathcal{F}, \mathcal{F})$ decreases with l (as discussed in Section 7) whereas the estimation error bound, being a function of the pseudo-dimension of $\mathcal{F}$, increases with l (in such a linear approximation architecture $V_{\mathcal{F}^+}$ equals the number of basis function plus one, that is, $V_{\mathcal{F}^+} = l + 2$) with rate $O\big(\big(\frac{l+2}{N}\big)^{1/2p}\big) + O(1/M^{1/2})$, disregarding logarithmic factors. According to this bound, the estimation error decreases when the number of samples increases, which is corroborated by the experiments that show that overfitting decreases when the number of samples N, M increases. Note that truncation never happens except when the degree of the polynomial is very large compared with the number of samples.

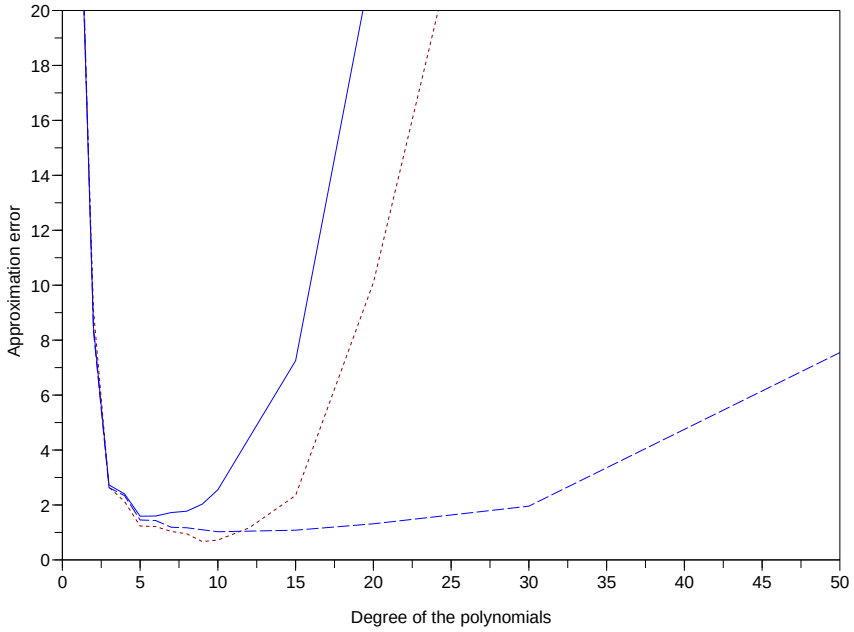


Figure 3: Approximation errors $||V^* - V_K||_\infty$ of the function V_K returned by sampling-based FVI after $K = 20$ iterations, for different values of the polynomials degree l, for $N = 100$, $M = 10$ (plain curve), $N = 100$, $M = 100$ (dot curve), and $N = 1000$, $M = 10$ (dash curve) samples. The plotted values are the average over 100 independent runs.

In our second set of experiments we investigated whether in this problem the single-sample or the multi-sample variant of the algorithm is more advantageous provided that sample collection is expensive or limited in some way.

Figure 4 shows the distributional character of $V_K - V^*$ as a function of the state. The order of the fitted polynomials is 5. The solid (black) curve shows the mean error (representing the bias) for 50 independent runs, the dashed (blue) curves show the upper and lower confidence intervals at 1.5-times the observed standard deviation, while the dash-dotted (red) curves show the minimum/maximum approximation errors. Note the peak at $\bar{x}$: The value function at this point is non-smooth, introducing a bias that converges to zero rather slowly (the same effect in Fourier analysis is known as the Gibbs phenomenon). It is also evident from the figure that the approximation error near the edges of the state space is larger. In polynomial interpolation for the uniform arrangements of the basepoints, the error actually blows up at the end of the intervals as the order of interpolation is increased (Runge's phenomenon). A general suggestion to avoid this is to increase the denseness of points near the edges or to introduce more flexible methods (e.g., splines). In FVI the edge effect is ultimately washed out, but it may still cause a considerable slow down of the procedure when the behavior of the value-function near the boundaries is critical.

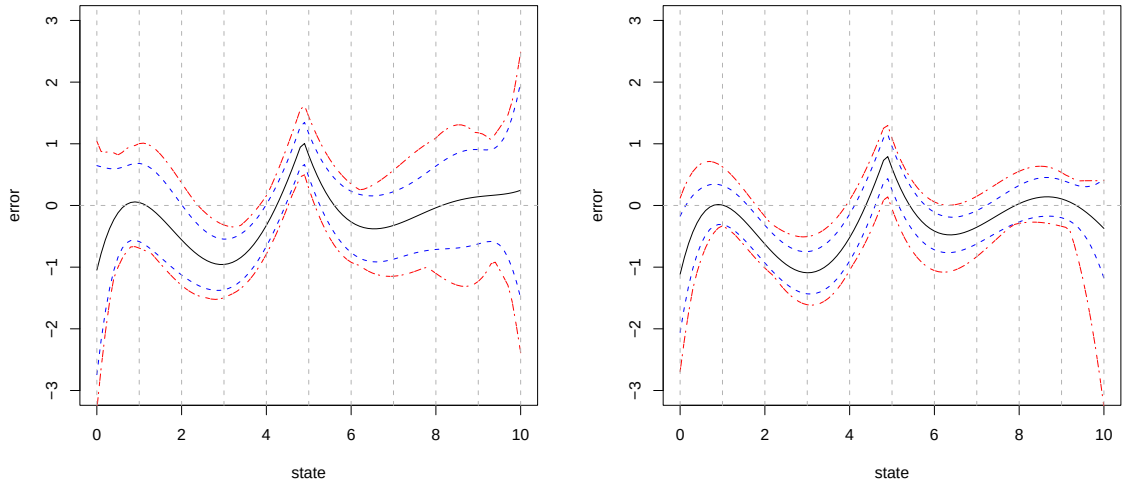


Figure 4: Approximation errors for the multi-sample (left figure) and the single-sample (right) variants of sampling based FVI. The figures show the distribution of errors for approximating the optimal value function as a function of the state, as measured using 50 independent runs. For both version, $N = 100$, $K = 10$. However, for the multi-sample variant $M = 10$, while for the single-sample variant $M = 100$, making the total number of samples used equal in the two cases.

Now, as to the comparison of the single- and multi-sample algorithms, it should be apparent from this figure that for this specific setup, the single-sample variant is actually preferable: The bias does not seem to increase much due to the reuse of samples, while the variance of the estimates decreases significantly.

10. Conclusions

We considered sampling-based FVI for discounted, large (possibly infinite) state space, finite-action Markovian Decision Processes when only a generative model of the environment is available. In each iteration, the image of the previous iterate under the Bellman operator is approximated at a finite number of points using a simple Monte-Carlo technique. A regression method is used then to fit a function to the data obtained. The main contributions of the paper are performance bounds for this procedure that holds with high probability. The bounds scale with the inherent Bellman error of the function space used in the regression step, and the stochastic stability properties of the MDP. It is an open question if the finiteness of the inherent Bellman error is necessary for the stability of FVI, but the counterexamples discussed in the introduction suggest that the inherent Bellman residual of the function space should indeed play a crucial role in the final performance of FVI. Even less is known about whether the stochastic stability conditions are necessary or if they can be relaxed.

We argued that by increasing the number of samples and the richness of the function space at the same time, the resulting algorithm can be shown to be consistent for a wide class of MDPs. The

derived rates show that, in line with our expectations, FVI would typically suffer from the curse-of-dimensionality except when some specific conditions (extreme smoothness, only a few state variables are relevant, sparsity, etc.) are met. Since these conditions could be difficult to verify a priori for any practical problem, adaptive methods are needed. We believe that the techniques developed in this paper may serve as a solid foundations for developing and studying such algorithms.

One immediate possibility along this line would be to extend our results to penalized empirical risk minimization when a penalty term penalizing the roughness of the candidate functions is added to the empirical risk. The advantage of this approach is that without any a priori knowledge of the smoothness class, the method allows one to achieve the optimal rate of convergence (see Györfi et al., 2002, Section 21.2).

Another problem left for future work is to improve the scaling of our bounds. An important open question is to establish tight lower bounds for the rate of convergence for value-function based RL methods.

There are other ways to improve the performance of our algorithm that are more directly related to specifics of RL. Both Tsitsiklis and Roy (2001) and Kakade (2003) argued that μ, the distribution used to sample the states should be selected to match the future state distribution of a (near-)optimal policy. Since the only way to learn about the optimal policy is by running the algorithm, one idea is to change the sampling distribution by moving it closer to the future-state distribution of the most recent policy. The improvement presumably manifests itself by decreasing the term including $C_{\rho,\mu}$. Another possibility is to adaptively choose M, the number of sampled next states based on the available local information like in active learning, hoping that this way the sample-efficiency of the algorithm could be further improved.

Acknowledgments

Csaba Szepesvári greatly acknowledges the support received from the Hungarian National Science Foundation (OTKA), Grant No. T047193, the Hungarian Academy of Sciences (Bolyai Fellowship), the Alberta Ingenuity Fund, NSERC and the Computer and Automation Research Institute of the Hungarian Academy of Sciences. We would like to thank Barnabás Póczos and the anonymous reviewers for helpful comments, suggestions and discussions.

Appendix A. Proof of Lemma 1

In order to prove Lemma 1 we use the following inequality due to Pollard:

Theorem 5 (Pollard, 1984) *Let $\mathcal{F}$ be a set of measurable functions $f : \mathcal{X} \to [0,K]$ and let $\varepsilon > 0$, N be arbitrary. If X_i, $i = 1,\dots,N$ is an i.i.d. sequence taking values in the space $\mathcal{X}$ then*

$$\mathbb{P}\left(\sup_{f\in\mathcal{F}}\left|\frac{1}{N}\sum_{i=1}^{N} f(X_i) - \mathbb{E}\left[f(X_1)\right]\right| > \varepsilon\right) \le 8\mathbb{E}\left[\mathcal{N}(\varepsilon/8, \mathcal{F}(X^{1:N}))\right] e^{-\frac{N\varepsilon^2}{128K^2}}.$$

Here one should perhaps work with outer expectations because, in general, the supremum of an uncountably many random variables cannot be guaranteed to be measurable. However, since for specific examples of function space $\mathcal{F}$, measurability can typically be established by routine separability arguments, we will altogether ignore these measurability issues in this paper.

Now, let us prove Lemma 1 that stated the finite-sample bound for a single iterate.

Proof Let Ω denote the sample space underlying the random variables. Let $\varepsilon'' > 0$ be arbitrary and let f^* be such that $\|f^* - TV\|_{p,\mu} \le \inf_{f\in\mathcal{F}} \|f - TV\|_{p,\mu} + \varepsilon''$. Define $\|\cdot\|_{p,\hat{\mu}}$ by

$$\|f\|_{p,\hat{\mu}}^p = \frac{1}{N}\sum_{i=1}^N |f(X_i)|^p.$$

We will prove the lemma by showing that the following sequence of inequalities hold simultaneously on a set of events of measure not smaller than $1-\delta$:

$$\begin{aligned}
\left\|V' - TV\right\|_{p,\mu} &\le \left\|V' - TV\right\|_{p,\hat{\mu}} + \varepsilon' && (10)\\
&\le \left\|V' - \hat{V}\right\|_{p,\hat{\mu}} + 2\varepsilon' && (11)\\
&\le \left\|f^* - \hat{V}\right\|_{p,\hat{\mu}} + 2\varepsilon' && (12)\\
&\le \|f^* - TV\|_{p,\hat{\mu}} + 3\varepsilon' && (13)\\
&\le \|f^* - TV\|_{p,\mu} + 4\varepsilon' && (14)\\
&= d_{p,\mu}(TV,\mathcal{F}) + 4\varepsilon' + \varepsilon''.
\end{aligned}$$

It follows then that $\|V' - TV\|_{p,\mu} \le \inf_{f\in\mathcal{F}} \|f - TV\|_{p,\mu} + 4\varepsilon' + \varepsilon''$ w.p. at least $1-\delta$. Since $\varepsilon'' > 0$ was arbitrary, it also follows that $\|V' - TV\|_{p,\mu} \le \inf_{f\in\mathcal{F}} \|f - TV\|_{p,\mu} + 4\varepsilon'$ w.p. at least $1-\delta$. Now, the Lemma follows by choosing $\varepsilon' = \varepsilon/4$.

Let us now turn to the proof of (10)–(14). First, observe that (12) holds due to the choice of V' since $\left\|V' - \hat{V}\right\|_{p,\hat{\mu}} \le \left\|f - \hat{V}\right\|_{p,\hat{\mu}}$ holds for all functions f from $\mathcal{F}$ and thus the same inequality holds for $f^* \in \mathcal{F}$, too.

Thus, (10)–(14) will be established if we prove that (10), (11), (13) and (14) all hold w.p. at least $1-\delta'$ with $\delta' = \delta/4$. Let

$$Q = \max\left(\left|\left\|V' - TV\right\|_{p,\mu} - \left\|V' - TV\right\|_{p,\hat{\mu}}\right|, \left|\|f^* - TV\|_{p,\mu} - \|f^* - TV\|_{p,\hat{\mu}}\right|\right).$$

We claim that

$$\mathbb{P}\left(Q > \varepsilon'\right) \le \delta', \tag{15}$$

where $\delta' = \delta/4$. From this, (10) and (14) will follow.

In order to prove (15) note that for all $\omega \in \Omega$, $V' = V'(\omega) \in \mathcal{F}$. Hence,

$$\sup_{f\in\mathcal{F}} \left|\|f - TV\|_{p,\mu} - \|f - TV\|_{p,\hat{\mu}}\right| \ge \left|\left\|V' - TV\right\|_{p,\mu} - \left\|V' - TV\right\|_{p,\hat{\mu}}\right|$$

holds pointwise in Ω. Therefore the inequality

$$\sup_{f\in\mathcal{F}} \left|\|f - TV\|_{p,\mu} - \|f - TV\|_{p,\hat{\mu}}\right| > Q \tag{16}$$

holds pointwise in Ω, too and hence

$$\mathbb{P}\left(Q > \varepsilon'\right) \le \mathbb{P}\left(\sup_{f\in\mathcal{F}} \left|\|f - TV\|_{p,\mu} - \|f - TV\|_{p,\hat{\mu}}\right| > \varepsilon'\right).$$

We claim that

$$\mathbb{P}\left(\sup_{f\in\mathcal{F}}\left|\|f-TV\|_{p,\mu}-\|f-TV\|_{p,\hat{\mu}}\right|>\varepsilon'\right)\leq\mathbb{P}\left(\sup_{f\in\mathcal{F}}\left|\|f-TV\|_{p,\mu}^{p}-\|f-TV\|_{p,\hat{\mu}}^{p}\right|>(\varepsilon')^{p}\right). \tag{17}$$

Consider any event ω such that

$$\sup_{f\in\mathcal{F}}\left|\|f-TV\|_{p,\mu}-\|f-TV\|_{p,\hat{\mu}}\right|>\varepsilon'.$$

For any such event, ω, there exist a function $f'\in\mathcal{F}$ such that

$$\left|\|f'-TV\|_{p,\mu}-\|f'-TV\|_{p,\hat{\mu}}\right|>\varepsilon'.$$

Pick such a function. Assume first that $\|f'-TV\|_{p,\hat{\mu}}\leq\|f'-TV\|_{p,\mu}$. Hence, $\|f'-TV\|_{p,\hat{\mu}}+\varepsilon'<\|f'-TV\|_{p,\mu}$. Since $p\geq 1$, the elementary inequality $x^p+y^p\leq(x+y)^p$ holds for any non-negative numbers x,y. Hence we get $\|f'-TV\|_{p,\hat{\mu}}^{p}+\varepsilon^p\leq(\|f'-TV\|_{p,\hat{\mu}}+\varepsilon)^p<\|f'-TV\|_{p,\mu}^{p}$ and thus

$$\left|\|f'-TV\|_{p,\hat{\mu}}^{p}-\|f'-TV\|_{p,\mu}^{p}\right|>\varepsilon^p.$$

This inequality can be shown to hold by an analogous reasoning when $\|f'-TV\|_{p,\hat{\mu}}>\|f'-TV\|_{p,\mu}$.

Inequality (17) now follows since

$$\sup_{f\in\mathcal{F}}\left|\|f-TV\|_{p,\mu}^{p}-\|f-TV\|_{p,\hat{\mu}}^{p}\right|\geq\left|\|f'-TV\|_{p,\mu}^{p}-\|f'-TV\|_{p,\hat{\mu}}^{p}\right|.$$

Now, observe that $\|f-TV\|_{p,\mu}^{p}=\mathbb{E}\left[|(f(X_1)-(TV)(X_1))|^p\right]$, and $\|f-TV\|_{p,\hat{\mu}}^{p}$ is thus just the sample average approximation of $\|f-TV\|_{p,\mu}^{p}$. Hence, by noting that the covering number associated with $\{f-TV|f\in\mathcal{F}\}$ is the same as the covering number of $\mathcal{F}$, calling for Theorem 5 results in

$$\mathbb{P}\left(\sup_{f\in\mathcal{F}}\left|\|f-TV\|_{p,\mu}^{p}-\|f-TV\|_{p,\hat{\mu}}^{p}\right|>(\varepsilon')^{p}\right)\leq 8\mathbb{E}\left[\mathcal{N}(\tfrac{(\varepsilon')^p}{8},\mathcal{F}(X^{1:N}))\right]e^{-\frac{N}{2}\left(\frac{1}{8}\left(\frac{\varepsilon'}{2V_{\max}}\right)^p\right)^2}.$$

By making the right-hand side upper bounded by $\delta'=\delta/4$ we find a lower bound on N, displayed in turn in (4). This finishes the proof of (15).

Now, let us prove inequalities (11) and (13). Let f denote an arbitrary random function such that $f=f(x;\omega)$ is measurable for each $x\in X$ and assume that f is uniformly bounded by $V_{\max}$. Making use of the triangle inequality

$$\left|\|f-g\|_{p,\hat{\mu}}-\|f-h\|_{p,\hat{\mu}}\right|\leq\|g-h\|_{p,\hat{\mu}},$$

we get that

$$\left|\|f-TV\|_{p,\hat{\mu}}-\|f-\hat{V}\|_{p,\hat{\mu}}\right|\leq\|TV-\hat{V}\|_{p,\hat{\mu}}. \tag{18}$$

Hence, it suffices to show that $\|TV-\hat{V}\|_{p,\hat{\mu}}\leq\varepsilon'$ holds w.p $1-\delta'$.

For this purpose we shall use Hoeffding's inequality (Hoeffding, 1963) and union bound arguments. Fix any index i ($1 \le i \le N$). Let $K_1 = \hat{R}_{\max} + \gamma V_{\max}$. Then, by assumption $R_j^{X_i,a} + \gamma V(Y_j^{X_i,a}) \in [-K_1, K_1]$ holds w.p. 1 and thus by Hoeffding's inequality,

$$\mathbb{P}\left(\left| \mathbb{E}\left[R_1^{X_i,a} + \gamma V(Y_1^{X_i,a}) \,|\, X^{1:N} \right] - \frac{1}{M} \sum_{j=1}^{M} R_j^{X_i,a} + \gamma V(Y_j^{X_i,a}) \right| > \varepsilon' \,|\, X^{1:N} \right) \le 2 e^{-\frac{2M(\varepsilon')^2}{K_1^2}}, \qquad (19)$$

where $X^{1:N} = (X_1, \ldots, X_N)$. Making the right-hand side upper bounded by $\delta'/(N|\mathcal{A}|)$ we find a lower bound on M (cf., Equation 5). Since

$$\left| (TV)(X_i) - \hat{V}(X_i) \right| \le \max_{a \in A} \left| \mathbb{E}\left[R_1^{X_i,a} + \gamma V(Y_1^{X_i,a}) \,|\, X^{1:N} \right] - \frac{1}{M} \sum_{j=1}^{M} \left[R_j^{X_i,a} + \gamma V(Y_j^{X_i,a}) \right] \right|$$

it follows by a union bounding argument that

$$\mathbb{P}\left(\left| (TV)(X_i) - \hat{V}(X_i) \right| > \varepsilon' \,|\, X^{1:N} \right) \le \delta'/N,$$

and hence another union bounding argument yields

$$\mathbb{P}\left(\max_{i=1,\ldots,N} \left| (TV)(X_i) - \hat{V}(X_i) \right|^p > (\varepsilon')^p \,|\, X^{1:N} \right) \le \delta'.$$

Taking the expectation of both sides of this inequality gives

$$\mathbb{P}\left(\max_{i=1,\ldots,N} \left| (TV)(X_i) - \hat{V}(X_i) \right|^p > (\varepsilon')^p \right) \le \delta'.$$

Hence also

$$\mathbb{P}\left(\frac{1}{N} \sum_{i=1}^{N} \left| (TV)(X_i) - \hat{V}(X_i) \right|^p > (\varepsilon')^p \right) \le \delta'$$

and therefore by (18),

$$\mathbb{P}\left(\left| \|f - TV\|_{p,\hat{\mu}} - \left\|f - \hat{V}\right\|_{p,\hat{\mu}} \right| > \varepsilon' \right) \le \delta'$$

Using this with $f = V'$ and $f = f^*$ shows that inequalities (11) and (13) each hold w.p. at least $1 - \delta'$. This finishes the proof of the lemma. ■

Now, let us turn to the proof of Lemma 2, which stated a finite-sample bound for the single-sample variant of the algorithm.

A.1 Proof of Lemma 2

Proof The proof is analogous to that of Lemma 1, hence we only give the differences. Up to (16) the two proofs proceed in an identical way, however, from (16) we continue by concluding that

$$\sup_{g \in \mathcal{F}} \sup_{f \in \mathcal{F}} \left| \|f - Tg\|_{p,\mu} - \|f - Tg\|_{p,\hat{\mu}} \right| > Q$$

holds pointwise in Ω. From this point onward, $\sup_{f\in\mathcal{F}}$ is replaced by $\sup_{g,f\in\mathcal{F}}$ throughout the proof of (15): The proof goes through as before until the point where Pollard's inequality is used. At this point, since we have two suprema, we need to consider covering numbers corresponding to the function set $\mathcal{F}_{T-} = \{ f - Tg \,|\, f \in \mathcal{F}, g \in \mathcal{F} \}$.

In the second part of the proof we must also use Pollard's inequality in place of Hoeffding's. In particular, (19) is replaced with

$$\mathbb{P}\left(\sup_{g\in\mathcal{F}} \left| \mathbb{E}\left[R_1^{X_i,a} + \gamma g(Y_1^{X_i,a}) \,|\, X^{1:N} \right] - \frac{1}{M}\sum_{j=1}^{M} R_j^{X_i,a} + \gamma g(Y_j^{X_i,a}) \right| > \varepsilon' \,\middle|\, X^{1:N} \right)$$
$$\leq \quad 8\mathbb{E}\left[\mathcal{N}(\varepsilon'/8, \mathcal{F}_+(Z_{i,a}^{1:M})) \right] e^{-\frac{M(\varepsilon')^2}{128K_1^2}},$$

where $Z_{i,a}^j = (R_j^{X_i,a}, Y_j^{X_i,a})$. Here $\mathcal{F}_+ = \{ h : \mathbb{R}\times\mathcal{X} \to \mathbb{R} \,|\, h(s,x) = s\mathbb{I}_{\{|s|\leq V_{\max}\}} + f(x) \text{ for some } f \in \mathcal{F} \}$. The proof is concluded by noting that the covering numbers of $\mathcal{F}_+$ can be bounded in terms of the covering numbers of $\mathcal{F}$ using the arguments presented after the Lemma at the end of Section 4. ■

Appendix B. Proof of Theorem 2

The theorem states PAC-bounds on the sample size of sampling-based FVI. The idea of the proof is to show that *(i)* if the errors in each iteration are small then the final error will be small when K, the number of iterations is high enough and *(ii)* the previous results (Lemma 1 and 2) show that the errors stay small with high probability in each iteration provided that M, N is high enough. Putting these results together gives the main result. Hence, we need to show *(i)*.

First, note that iteration (7) or (6) may be written

$$V_{k+1} = TV_k - \varepsilon_k$$

where ε_k, defined by $\varepsilon_k = TV_k - V_{k+1}$, is the approximation error of the Bellman operator applied to V_k due to sampling. The proof is done in two steps: we first prove a statement that gives pointwise bounds (i.e., the bounds hold for any state $x \in \mathcal{X}$) which is then used to prove the necessary L^p bounds. Parts *(i)* and *(ii)* are connected in Sections B.3, B.4.

B.1 Pointwise Error Bounds

Lemma 3 *We have*

$$V^* - V^{\pi_K} \leq \quad (I - \gamma P^{\pi_K})^{-1} \Big\{ \textstyle\sum_{k=0}^{K-1} \gamma^{K-k} \left[(P^{\pi^*})^{K-k} + P^{\pi_K} P^{\pi_{K-1}} \dots P^{\pi_{k+1}} \right] |\varepsilon_k| \qquad (20)$$
$$+ \gamma^{K+1} \left[(P^{\pi^*})^{K+1} + (P^{\pi_K} P^{\pi_{K-1}} \dots P^{\pi_0}) \right] |V^* - V_0| \Big\}.$$

Proof Since $TV_k \geq T^{\pi^*} V_k$, we have

$$V^* - V_{k+1} = T^{\pi^*} V^* - T^{\pi^*} V_k + T^{\pi^*} V_k - TV_k + \varepsilon_k \leq \gamma P^{\pi^*}(V^* - V_k) + \varepsilon_k,$$

from which we deduce by induction

$$V^* - V_K \leq \sum_{k=0}^{K-1} \gamma^{K-k-1} (P^{\pi^*})^{K-k-1} \varepsilon_k + \gamma^K (P^{\pi^*})^K (V^* - V_0). \qquad (21)$$

Similarly, from the definition of π_k and since $TV^* \geq T^{\pi_k}V^*$, we have

$$V^* - V_{k+1} = TV^* - T^{\pi_k}V^* + T^{\pi_k}V^* - TV_k + \varepsilon_k \geq \gamma P^{\pi_k}(V^* - V_k) + \varepsilon_k.$$

Thus, by induction,

$$V^* - V_K \geq \sum_{k=0}^{K-1} \gamma^{K-k-1}(P^{\pi_{K-1}}P^{\pi_{K-2}}\dots P^{\pi_{k+1}})\varepsilon_k + \gamma^K(P^{\pi_{K-1}}P^{\pi_{K-2}}\dots P^{\pi_0})(V^* - V_0). \qquad (22)$$

Now, from the definition of π_K, $T^{\pi_K}V_K = TV_K \geq T^{\pi^*}V_K$, and we have

$$\begin{aligned} V^* - V^{\pi_K} &= T^{\pi^*}V^* - T^{\pi^*}V_K + T^{\pi^*}V_K - TV_K + T^{\pi_K}V_K - T^{\pi_K}V^{\pi_K} \\ &\leq \gamma P^{\pi^*}(V^* - V_K) + \gamma P^{\pi_K}(V_K - V^* + V^* - V^{\pi_K}) \\ (I - \gamma P^{\pi_K})(V^* - V^{\pi_K}) &\leq \gamma(P^{\pi^*} - P^{\pi_K})(V^* - V_K), \end{aligned}$$

and since $(I - \gamma P^{\pi_K})$ is invertible and its inverse is a monotonic operator[14] (we may write $(I - \gamma P^{\pi_K})^{-1} = \sum_{m\geq 0}\gamma^m(P^{\pi_K})^m$), we deduce

$$V^* - V^{\pi_K} \leq \gamma(I - \gamma P^{\pi_K})^{-1}(P^{\pi^*} - P^{\pi_K})(V^* - V_K)$$

Now, using (21) and (22),

$$\begin{aligned} V^* - V^{\pi_K} \leq \quad & (I - \gamma P^{\pi_K})^{-1}\Big\{ \textstyle\sum_{k=0}^{K-1}\gamma^{K-k}\big[(P^{\pi^*})^{K-k} - P^{\pi_K}P^{\pi_{K-1}}\dots P^{\pi_{k+1}}\big]\varepsilon_k \\ & + \gamma^{K+1}\big[(P^{\pi^*})^{K+1} - (P^{\pi_K}P^{\pi_{K-1}}\dots P^{\pi_0})\big](V^* - V_0)\Big\} \end{aligned}$$

from which (20) follows by taking the absolute value of both sides. ■

B.2 L^p Error Bounds

We have the following approximation results.

Lemma 4 *For any $\eta > 0$, there exists K that is linear in $\log(1/\eta)$ (and $\log V_{\max}$) such that, if the $L^p(\mu)$ norm of the approximation errors is bounded by some ε ($\|\varepsilon_k\|_{p,\mu} \leq \varepsilon$ for all $0 \leq k < K$) then*

- *Given Assumption A1 we have*

$$\|V^* - V^{\pi_K}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2}C_\mu^{1/p}\varepsilon + \eta. \qquad (23)$$

- *Given Assumption A2 we have*

$$\|V^* - V^{\pi_K}\|_{p,\rho} \leq \frac{2\gamma}{(1-\gamma)^2}C_{\rho,\mu}^{1/p}\varepsilon + \eta. \qquad (24)$$

14. An operator T is monotonic if for any $x \leq y$, $Tx \leq Ty$.

Note that if $\|\varepsilon_k\|_\infty \leq \varepsilon$ then letting $p \to \infty$ we get back the well-known, unimprovable supremum-norm error bounds

$$\limsup_{K\to\infty} \|V^* - V^{\pi_K}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2}\varepsilon$$

for approximate value iteration (Bertsekas and Tsitsiklis, 1996). (In fact, by inspecting the proof below it turns out that for this the weaker condition, $\limsup_{k\to\infty} \|\varepsilon_k\|_\infty \leq \varepsilon$ suffices, too.)

Proof We have seen that if A1 holds then A2 also holds, and for any distribution ρ, $C_{\rho,\mu} \leq C_\mu$. Thus, if the bound (24) holds for any ρ then choosing ρ to be a Dirac at each state proves (23). Thus we only need to prove (24).

We may rewrite (20) as

$$V^* - V^{\pi_K} \leq \frac{2\gamma(1-\gamma^{K+1})}{(1-\gamma)^2}\left[\sum_{k=0}^{K-1} \alpha_k A_k |\varepsilon_k| + \alpha_K A_K |V^* - V_0|\right],$$

with the positive coefficients

$$\alpha_k = \frac{(1-\gamma)\gamma^{K-k-1}}{1-\gamma^{K+1}}, \text{ for } 0 \leq k < K, \text{ and } \alpha_K = \frac{(1-\gamma)\gamma^K}{1-\gamma^{K+1}},$$

(defined such that they sum to 1) and the probability kernels:

$$\begin{aligned} A_k &= \frac{1-\gamma}{2}(I - \gamma P^{\pi_K})^{-1}\left[(P^{\pi^*})^{K-k} + P^{\pi_K} P^{\pi_{K-1}} \ldots P^{\pi_{k+1}}\right], \text{ for } 0 \leq k < K, \\ A_K &= \frac{1-\gamma}{2}(I - \gamma P^{\pi_K})^{-1}\left[(P^{\pi^*})^{K+1} + P^{\pi_K} P^{\pi_{K-1}} \ldots P^{\pi_0}\right]. \end{aligned}$$

We have:

$$\begin{aligned} \|V^* - V^{\pi_K}\|_{p,\rho}^p &= \int \rho(dx)|V^*(x) - V^{\pi_K}(x)|^p \\ &\leq \left[\frac{2\gamma(1-\gamma^{K+1})}{(1-\gamma)^2}\right]^p \int \rho(dx)\left[\sum_{k=0}^{K-1} \alpha_k A_k |\varepsilon_k| + \alpha_K A_K |V^* - V_0|\right]^p (x) \\ &\leq \left[\frac{2\gamma(1-\gamma^{K+1})}{(1-\gamma)^2}\right]^p \int \rho(dx)\left[\sum_{k=0}^{K-1} \alpha_k A_k |\varepsilon_k|^p + \alpha_K A_K |V^* - V_0|^p\right](x), \end{aligned}$$

by using two times Jensen's inequality (since the sum of the coefficients α_k, for $k \in [0,K]$, is 1, and the A_k are positive linear operators with $A_k 1 = 1$) (i.e., convexity of $x \to |x|^p$).

The term $|V^* - V_0|$ may be bounded by $2V_{\max}$. Now, under Assumption A2, $\rho A_k \leq (1-\gamma)\sum_{m\geq 0}\gamma^m c(m+K-k)\mu$ and we deduce

$$\|V^* - V^{\pi_K}\|_{p,\rho}^p \leq \left[\frac{2\gamma(1-\gamma^{K+1})}{(1-\gamma)^2}\right]^p \left[\sum_{k=0}^{K-1} \alpha_k (1-\gamma) \sum_{m\geq 0} \gamma^m c(m+K-k) \|\varepsilon_k\|_{p,\mu}^p + \alpha_K (2V_{\max})^p\right].$$

Replace α_k by their values, and from the definition of $C_{\rho,\mu}$, and since $\|\varepsilon_k\|_{p,\mu} \le \varepsilon$, we have:

$$\begin{aligned}\|V^* - V^{\pi_K}\|_{p,\rho}^p &\le \left[\frac{2\gamma(1-\gamma^{K+1})}{(1-\gamma)^2}\right]^p \left[\frac{(1-\gamma)^2}{1-\gamma^{K+1}} \right. \\ &\qquad \left. \sum_{m\ge 0}\sum_{k=0}^{K-1} \gamma^{m+K-k-1} c(m+K-k)\varepsilon^p + \frac{(1-\gamma)\gamma^K}{1-\gamma^{K+1}}(2V_{\max})^p\right] \\ &\le \left[\frac{2\gamma(1-\gamma^{K+1})}{(1-\gamma)^2}\right]^p \left[\frac{1}{1-\gamma^{K+1}} C_{\rho,\mu}\varepsilon^p + \frac{(1-\gamma)\gamma^K}{1-\gamma^{K+1}}(2V_{\max})^p\right]\end{aligned}$$

Thus there is K linear in $\log(1/\eta)$ and $\log V_{\max}$ such that

$$\gamma^K < \left[\frac{(1-\gamma)^2}{4\gamma V_{\max}}\eta\right]^p$$

such that the second term is bounded by η^p, thus,

$$\|V^* - V^{\pi_K}\|_{p,\rho}^p \le \left[\frac{2\gamma}{(1-\gamma)^2}\right]^p C_{\rho,\mu}\varepsilon^p + \eta^p$$

thus

$$\|V^* - V^{\pi_K}\|_{p,\rho} \le \frac{2\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p}\varepsilon + \eta$$

■

B.3 From Pointwise Expectations to Conditional Expectations

We will need the following lemma in the proof of the theorem:

Lemma 5 *Assume that X, Y are independent random variables taking values in the respective measurable spaces, $\mathcal{X}$ and $\mathcal{Y}$. Let $f : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ be a Borel-measurable function such that $\mathbb{E}[f(X,Y)]$ exists. Assume that for all $y \in \mathcal{Y}$, $\mathbb{E}[f(X,y)] \ge 0$. Then $\mathbb{E}[f(X,Y)|Y] \ge 0$ holds, too, w.p.1.*

This lemma is an immediate consequence of the following result, whose proof is given for the sake of completeness:

Lemma 6 *Assume that X, Y are independent random variables taking values in the respective measurable spaces, $\mathcal{X}$ and $\mathcal{Y}$. Let $f : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ be a Borel-measurable function and assume that $\mathbb{E}[f(X,Y)]$ exists. Let $g(y) = \mathbb{E}[f(X,y)]$. Then $\mathbb{E}[f(X,Y)|Y] = g(Y)$ holds w.p.1.*

Proof Let us first consider the case when f has the form $f(x,y) = \mathbb{I}_{\{x\in A\}}\mathbb{I}_{\{y\in B\}}$, where $A \subset \mathcal{X}$, $B \subset \mathcal{Y}$ are measurable sets. Write $r(x) = \mathbb{I}_{\{x\in A\}}$ and $s(y) = \mathbb{I}_{\{y\in B\}}$. Then $\mathbb{E}[f(X,Y)|Y] = \mathbb{E}[r(X)s(Y)|Y] = r(Y)\mathbb{E}[s(X)|Y]$ since $s(Y)$ is Y-measurable. Since X and Y are independent, so are $s(X)$ and Y and thus $\mathbb{E}[s(X)|Y] = \mathbb{E}[s(X)]$. On the other hand, $g(y) = \mathbb{E}[r(X)s(y)] = s(y)\mathbb{E}[r(X)]$, and thus it indeed holds that $\mathbb{E}[f(X,Y)|Y] = g(Y)$ w.p.1. Now, by the additivity of expectations the same relation holds for sums of functions of the above form and hence, ultimately, for all simple functions. If f is nonnegative valued then we can find a sequence of increasing simple functions f_n with limit f.

By Lebesgue's monotone convergence theorem, $g_n(y) \stackrel{\text{def}}{=} \mathbb{E}[f_n(X,y)] \to \mathbb{E}[f(X,y)] (= g(y))$. Further, since Lebesgue's monotone convergence theorem also holds for conditional expectations, we also have $\mathbb{E}[f_n(X,Y)|Y] \to \mathbb{E}[f(X,Y)|Y]$. Since $g_n(Y) = \mathbb{E}[f_n(X,Y)|Y] \to \mathbb{E}[f(X,Y)|Y]$ w.p.1., and $g_n(Y) \to g(Y)$ w.p.1., we get that $g(Y) = \mathbb{E}[f(X,Y)|Y]$ w.p.1. Extension to an arbitrary function follows by decomposing the function into its positive and negative parts. ■

B.4 Proof of Theorem 2

Proof Let us consider first the multi-sample variant of the algorithm under Assumption A2. Fix $\varepsilon, \delta > 0$. Let the iterates produced by the algorithm be $V_1, \ldots, V_K$. Our aim is to show that by selecting the number of iterates, K and the number of samples, N, M large enough, the bound

$$\|V^* - V^{\pi_K}\|_{p,\rho} \leq \frac{2\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p} \, d_{p,\mu}(T\mathcal{F}, \mathcal{F}) + \varepsilon \tag{25}$$

holds w.p. at least $1-\delta$. First, note that by construction the iterates V_k remain bounded by $V_{\max}$. By Lemma 4, under Assumption A2, for all those events, where the error $\varepsilon_k = TV_k - V_{k+1}$ of the kth iterate is below (in $L^p(\mu)$-norm) some level ε_0, we have

$$\|V^* - V^{\pi_K}\|_{p,\rho} \leq \frac{2\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p} \varepsilon_0 + \eta, \tag{26}$$

provided that $K = \Omega(\log(1/\eta))$. Now, choose $\varepsilon' = (\varepsilon/2)(1-\gamma)^2/(2\gamma C_{\rho,\mu}^{1/p})$ and $\eta = \varepsilon/2$. Let $f(\varepsilon, \delta)$ denote the function that gives lower bounds on N, M in Lemma 1 based on the value of the desired estimation error ε and confidence δ. Let $(N, M) \geq f(\varepsilon', \delta/K)$. One difficulty is that V_k, the kth iterate is random itself, hence Lemma 1 (stated for deterministic functions) cannot be applied directly. However, thanks to the independence of samples between iterates, this is easy to fix via the application of Lemma 5.

To show this let us denote the collection of random variables used in the kth step by S_k. Hence, S_k consists of the N basepoints, as well as $|\mathcal{A}| \times N \times M$ next states and rewards. Further, introduce the notation $V'(V, S_k)$ to denote the result of solving the optimization problem (2)–(3) based on the sample S_k and starting from the value function $V \in B(\mathcal{X})$. By Lemma 1,

$$\mathbb{P}\left(\left\| V'(V, S_k) - TV \right\|_{p,\mu} \leq d_{p,\mu}(TV, \mathcal{F}) + \varepsilon' \right) \geq 1 - \delta/K.$$

Now let us apply Lemma 5 with $X := S_k$, $Y := V_k$ and $f(S, V) = \mathbb{I}_{\{\|V'(V,S) - TV\|_{p,\mu} \leq d_{p,\mu}(TV, \mathcal{F}) + \varepsilon'\}} - (1 - \delta/K)$. Since S_k is independent of V_k the lemma can indeed be applied. Hence,

$$\mathbb{P}\left(\left\| V'(V_k, S_k) - TV_k \right\|_{p,\mu} \leq d_{p,\mu}(TV_k, \mathcal{F}) + \varepsilon' | V_k \right) \geq 1 - \delta/K.$$

Taking the expectation of both sides gives $\mathbb{P}\left(\|V'(V_k, S_k) - TV_k\|_{p,\mu} \leq d_{p,\mu}(TV_k, \mathcal{F}) + \varepsilon' \right) \geq 1 - \delta/K$. Since $V'(V_k, S_k) = V_{k+1}$, $\varepsilon_k = TV_k - V_{k+1}$, we thus have that

$$\|\varepsilon_k\|_{p,\mu} \leq d_{p,\mu}(TV, \mathcal{F}) + \varepsilon' \tag{27}$$

holds except for a set of bad events B_k of measure at most δ/K.

Hence, inequality (27) holds simultaneously for $k = 1, \ldots, K$, except for the events in $B = \cup_k B_k$. Note that $\mathbb{P}(B) \le \sum_{k=1}^K \mathbb{P}(B_k) \le \delta$. Now pick any event in the complementer of B. Thus, for such an event (26) holds with $\varepsilon_0 = d_{p,\mu}(TV, \mathcal{F}) + \varepsilon'$. Plugging in the definitions of ε' and η we obtain (25).

Now assume that the MDP satisfies Assumption A1. As before, we conclude that (27) holds except for the events in B_k and with the same choice of N and M, we still have $\mathbb{P}(B) = \mathbb{P}(\cup_k B_K) \le \delta$. Now, using (23) we conclude that except on the set B, $\|V^* - V^{\pi_K}\|_\infty \le \frac{2\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p} d_{p,\mu}(T\mathcal{F}, \mathcal{F}) + \varepsilon$, concluding the first part of the proof.

For single-sample FVI the proof proceeds identically, except that now one uses Lemma 2 in place of Lemma 1. ■

Appendix C. Proof of Theorem 3

Proof We would like to prove that the policy defined in Section 6 gives close to optimal performance. Let us prove first the statement under Assumption A2.

By the choice of M', it follows using Hoeffding's inequality (see also Even-Dar et al., 2002, Theorem 1) that $\pi_{\alpha,\lambda}^K$ selects α-greedy actions w.p. at least $1-\lambda$.

Let π_α^K be a policy that selects α-greedy actions. A straightforward adaptation of the proof of Lemma 5.17 of Szepesvári (2001) yields that for all state $x \in \mathcal{X}$,

$$|V^{\pi_{\alpha,\lambda}^K}(x) - V^{\pi_\alpha^K}(x)| \le \frac{2V_{\max}\lambda}{1-\gamma}. \tag{28}$$

Now, use the triangle inequality to get

$$\left\|V^* - V^{\pi_{\alpha,\lambda}^K}\right\|_{p,\rho} \le \left\|V^* - V^{\pi_\alpha^K}\right\|_{p,\rho} + \left\|V^{\pi_\alpha^K} - V^{\pi_{\alpha,\lambda}^K}\right\|_{p,\rho}.$$

By (28), the second term can be bounded by $\frac{2V_{\max}\lambda}{1-\gamma}$, so let us consider the first term.

A modification of Lemmas 3 and 4 yields the following result, the proof of which will be given at the end of this section:

Lemma 7 *The following bound*

$$\left\|V^* - V^{\pi_\alpha^K}\right\|_{p,\rho} \le 2^{1-1/p}\left[\frac{2\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p} \max_{0\le k<K} \|\varepsilon_k\|_{p,\mu} + \eta + \frac{\alpha}{1-\gamma}\right] \tag{29}$$

holds for K such that $\gamma^K < \left[\frac{(1-\gamma)^2}{4\gamma V_{\max}}\eta\right]^p$.

Again, let $f(\varepsilon,\delta)$ be the function that gives the bounds on N, M in Lemma 1 for given ε and δ and set $(N,M) \ge f(\varepsilon',\delta/K)$ for ε' to be chosen later. Using the same argument as in the proof of Theorem 2 and Lemma 1 we may conclude that $\|\varepsilon_k\|_{p,\mu} \le d_{p,\mu}(TV_k, \mathcal{F}) + \varepsilon' \le d_{p,\mu}(T\mathcal{F}, \mathcal{F}) + \varepsilon'$ holds except for a set B_k with $\mathbb{P}(B_k) \le \delta/K$.

Thus, except on the set $B = \cup_k B_k$ of measure not more than δ,

$$\begin{aligned}\left\|V^* - V^{\pi_{\alpha,\lambda}^K}\right\|_{p,\rho} &\le 2^{1-1/p}\left[\frac{2\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p}\left(d_{p,\mu}(T\mathcal{F}, \mathcal{F}) + \varepsilon'\right) + \eta + \frac{\alpha}{1-\gamma}\right] + \frac{2V_{\max}\lambda}{1-\gamma} \\ &\le \left[\frac{4\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p} d_{p,\mu}(T\mathcal{F}, \mathcal{F}) + \frac{4\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p}\varepsilon' + 2\eta + \frac{2\alpha}{1-\gamma}\right] + \frac{2V_{\max}\lambda}{1-\gamma}.\end{aligned}$$

Now define $\alpha = \varepsilon(1-\gamma)/8$, $\eta = \varepsilon/8$, $\varepsilon' = \frac{\varepsilon}{4}\frac{(1-\gamma)^2}{4\gamma}C_{\rho,\mu}^{-1/p}$ and $\lambda = \frac{\varepsilon}{4}\frac{(1-\gamma)}{2V_{\max}}$ to conclude that

$$\left\|V^* - V^{\pi^K_{\alpha,\lambda}}\right\|_{p,\rho} \leq \frac{4\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p} d_{p,\mu}(T\mathcal{F},\mathcal{F}) + \varepsilon$$

holds everywhere except on B. Also, just like in the proof of Theorem 2, we get that under Assumption A1 the statement for the supremum norm holds, as well.

It thus remained to prove Lemma 7:

Proof [Lemma 7] Write $\mathbf{1}$ for the constant function that equals to 1. Since π^K_α is α-greedy w.r.t. V_K, we have $TV_K \geq T^{\pi^K_\alpha}V_K \geq TV_K - \alpha\mathbf{1}$. Thus, similarly to the proof of Lemma 3, we have

$$\begin{aligned}
V^* - V^{\pi^K_\alpha} &= T^{\pi^*}V^* - T^{\pi^*}V_K + T^{\pi^*}V_K - TV_K + TV_K - T^{\pi^K_\alpha}V_K + T^{\pi^K_\alpha}V_K - T^{\pi^K_\alpha}V^{\pi^K_\alpha} \\
&\leq \gamma P^{\pi^*}(V^* - V_K) + \gamma P^{\pi^K_\alpha}(V_K - V^* + V^* - V^{\pi^K_\alpha}) + \alpha\mathbf{1} \\
&\leq (I - \gamma P^{\pi^K_\alpha})^{-1}\left[\gamma(P^{\pi^*} - P^{\pi^K_\alpha})(V^* - V_K)\right] + \frac{\alpha\mathbf{1}}{1-\gamma},
\end{aligned}$$

and by using (21) and (22), we deduce

$$\begin{aligned}
V^* - V^{\pi^K_\alpha} \leq\ & (I - \gamma P^{\pi^K_\alpha})^{-1}\Big\{\textstyle\sum_{k=0}^{K-1}\gamma^{K-k}\left[(P^{\pi^*})^{K-k} + P^{\pi^K_\alpha}P^{\pi_{K-1}}\dots P^{\pi_{k+1}}\right]|\varepsilon_k| \\
& + \gamma^{K+1}\left[(P^{\pi^*})^{K+1} + (P^{\pi^K_\alpha}P^{\pi_K}P^{\pi_{K-1}}\dots P^{\pi_1})\right]|V^* - V_0|\Big\} + \tfrac{\alpha\mathbf{1}}{1-\gamma}.
\end{aligned}$$

Now, from the inequality $|a+b|^p \leq 2^{p-1}(|a|^p + |b|^p)$, we deduce, by following the same lines as in the proof of Lemma 4, that

$$\left\|V^* - V^{\pi^K_\alpha}\right\|_{p,\rho}^p \leq 2^{p-1}\left\{\left[\frac{2\gamma}{(1-\gamma)^2}\right]^p C_{\rho,\mu}(\max_{0\leq k<K}\|\varepsilon_k\|_{p,\mu})^p + \eta^p + \left[\frac{\alpha}{1-\gamma}\right]^p\right\},$$

and Lemma 7 follows. ■

■

References

M. Anthony and P.L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge, UK, 1999.

A. Antos, Cs. Szepesvári, and R. Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. In G. Lugosi and H.U. Simon, editors, *The Nineteenth Annual Conference on Learning Theory, COLT 2006, Proceedings*, volume 4005 of *LNCS/LNAI*, pages 574–588, Berlin, Heidelberg, June 2006. Springer-Verlag. (Pittsburgh, PA, USA, June 22–25, 2006.).

A. Antos, Cs. Szepesvári, and R. Munos. Value-iteration based fitted policy iteration: learning with a single trajectory. In *2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2007)*, pages 330–337. IEEE, April 2007. (Honolulu, Hawaii, Apr 1–5, 2007.).

A. Antos, Cs. Szepesvári, and R. Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71:89–129, 2008.

Leemon C. Baird. Residual algorithms: Reinforcement learning with function approximation. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37, San Francisco, CA, 1995. Morgan Kaufmann.

R.E. Bellman and S.E. Dreyfus. Functional approximation and dynamic programming. *Math. Tables and other Aids Comp.*, 13:247–251, 1959.

D. P. Bertsekas and S.E. Shreve. *Stochastic Optimal Control (The Discrete Time Case)*. Academic Press, New York, 1978.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

P. Bougerol and N. Picard. Strict stationarity of generalized autoregressive processes. *Annals of Probability*, 20:1714–1730, 1992.

E.W. Cheney. *Introduction to Approximation Theory*. McGraw-Hill, London, New York, 1966.

C.S. Chow and J.N. Tsitsiklis. The complexity of dynamic programming. *Journal of Complexity*, 5:466–488, 1989.

C.S. Chow and J.N. Tsitsiklis. An optimal multigrid algorithm for continuous state discrete time stochastic control. *IEEE Transactions on Automatic Control*, 36(8):898–914, 1991.

N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines (and other kernel-based learning methods)*. Cambridge University Press, 2000.

R.H. Crites and A.G. Barto. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 9*, 1997.

R. DeVore. *Nonlinear Approximation*. Acta Numerica, 1997.

T. G. Dietterich and X. Wang. Batch value function approximation via support vectors. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

E. Even-Dar, S. Mannor, and Y. Mansour. PAC bounds for multi-armed bandit and Markov decision processes. In *Fifteenth Annual Conference on Computational Learning Theory (COLT)*, pages 255–270, 2002.

G.J. Gordon. Stable function approximation in dynamic programming. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 261–268, San Francisco, CA, 1995. Morgan Kaufmann.

A. Gosavi. A reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis. *Machine Learning*, 55:5–29, 2004.

U. Grendander. *Abstract Inference*. Wiley, New York, 1981.

L. Györfi, M. Kohler, A. Krzyżak, and H. Walk. *A distribution-free theory of nonparametric regression*. Springer-Verlag, New York, 2002.

M. Haugh. Duality theory and simulation in financial engineering. In *Proceedings of the Winter Simulation Conference*, pages 327–334, 2003.

D. Haussler. Sphere packing numbers for subsets of the boolean n-cube with bounded Vapnik-Chervonenkis dimension. *Journal of Combinatorial Theory, Series A*, 69(2):217–232, 1995.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

T. Jung and T. Uthmann. Experiments in value function approximation with sparse support vector regression. In *ECML*, pages 180–191, 2004.

S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

S.M. Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

M. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markovian decision processes. In *Proceedings of IJCAI'99*, pages 1324–1331, 1999.

G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.

M. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

W.S. Lee, P.L. Bartlett, and R.C. Williamson. Efficient agnostic learning of neural networks with bounded fan-in. *IEEE Transactions on Information Theory*, 42(6):2118–2132, 1996.

F. A. Longstaff and E. S. Shwartz. Valuing american options by simulation: A simple least-squares approach. *Rev. Financial Studies*, 14(1):113–147, 2001.

S. Mahadevan, N. Marchalleck, T. Das, and A. Gosavi. Self-improving factory simulation using continuous-time average-reward reinforcement learning. In *Proceedings of the 14th International Conference on Machine Learning (IMLC '97)*, 1997.

T.L. Morin. Computational advances in dynamic programming. In *Dynamic Programming and its Applications*, pages 53–90. Academic Press, 1978.

R. Munos. Error bounds for approximate policy iteration. In *19th International Conference on Machine Learning*, pages 560–567, 2003.

R. Munos. Error bounds for approximate value iteration. *American Conference on Artificial Intelligence*, 2005.

S.A. Murphy. A generalization error for Q-learning. *Journal of Machine Learning Research*, 6: 1073–1097, 2005.

A.Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence*, pages 406–415, 2000.

P. Niyogi and F. Girosi. Generalization bounds for function approximation from scattered noisy data. *Advances in Computational Mathematics*, 10:51–80, 1999.

D. Ormoneit and S. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49:161–178, 2002.

M.L. Puterman. *Markov Decision Processes — Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

D. Reetz. Approximate solutions of a discounted Markovian decision problem. *Bonner Mathematischer Schriften*, 98: Dynamische Optimierungen:77–92, 1977.

M. Riedmiller. Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method. In *16th European Conference on Machine Learning*, pages 317–328, 2005.

J. Rust. Numerical dyanmic programming in economics. In H. Amman, D. Kendrick, and J. Rust, editors, *Handbook of Computational Economics*. Elsevier, North Holland, 1996a.

J. Rust. Using randomization to break the curse of dimensionality. *Econometrica*, 65:487–516, 1996b.

A.L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development*, pages 210–229, 1959. Reprinted in *Computers and Thought*, E.A. Feigenbaum and J. Feldman, editors, McGraw-Hill, New York, 1963.

A.L. Samuel. Some studies in machine learning using the game of checkers, II – recent progress. *IBM Journal on Research and Development*, pages 601–617, 1967.

N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory Series A*, 13:145–147, 1972.

B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

S.P. Singh and D.P. Bertsekas. Reinforcement learning for dynamic channel allocation in cellular telephone systems. In *Advances in Neural Information Processing Systems 9*, 1997.

S.P. Singh, T. Jaakkola, and M.I. Jordan. Reinforcement learning with soft state aggregation. In *Proceedings of Neural Information Processing Systems 7*, pages 361–368. MIT Press, 1995.

C.J. Stone. Optimal rates of convergence for nonparametric estimators. *Annals of Statistics*, 8: 1348–1360, 1980.

C.J. Stone. Optimal global rates of convergence for nonparametric regression. *Annals of Statistics*, 10:1040–1053, 1982.

Cs. Szepesvári. Efficient approximate planning in continuous space Markovian decision problems. *AI Communications*, 13:163–176, 2001.

Cs. Szepesvári. Efficient approximate planning in continuous space Markovian decision problems. *Journal of European Artificial Intelligence Research*, 2000. accepted.

Cs. Szepesvári and R. Munos. Finite time bounds for sampling based fitted value iteration. In *ICML'2005*, pages 881–886, 2005.

Cs. Szepesvári and W.D. Smart. Interpolation-based Q-learning. In D. Schuurmans R. Greiner, editor, *Proceedings of the International Conference on Machine Learning*, pages 791–798, 2004.

G.J. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38: 58–67, March 1995.

J. N. Tsitsiklis and Van B. Roy. Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks*, 12:694–703, 2001.

J. N. Tsitsiklis and B. Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22:59–94, 1996.

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*, 16:264–280, 1971.

X. Wang and T.G. Dietterich. Efficient value function approximation using regression trees. In *Proceedings of the IJCAI Workshop on Statistical Machine Learning for Large-Scale Optimization*, Stockholm, Sweden, 1999.

T. Zhang. Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research*, 2:527–550, 2002.

W. Zhang and T. G. Dietterich. A reinforcement learning approach to job-shop scheduling. In *Proceedings of the International Joint Conference on Artificial Intellience*, 1995.

Journal of Machine Learning Research 9 (2008) 859-891 Submitted 1/07; Revised 12/07; Published 5/08

An Error Bound Based on a Worst Likely Assignment

Eric Bax BAXHOME@YAHOO.COM
PO Box 60543
Pasadena, CA 91116-6543

Augusto Callejas CALLEJAS@ALUMNI.CALTECH.EDU
195 S. Wilson Avenue
Apartment 3
Pasadena, CA 91106

Editor: Gábor Lugosi

Abstract

This paper introduces a new PAC transductive error bound for classification. The method uses information from the training examples and inputs of working examples to develop a set of likely assignments to outputs of the working examples. A likely assignment with maximum error determines the bound. The method is very effective for small data sets.

Keywords: error bound, transduction, nearest neighbor, dynamic programming

1. Introduction

An error bound based on VC dimension (Vapnik and Chervonenkis, 1971; Vapnik, 1998) uses uniform bounds over the largest number of assignments possible from a class of classifiers, based on worst-case arrangements of training and working examples. However, as the number of training examples grows, the probability that training error is a good approximation of working error is so great that the VC error bound succeeds in spite of using uniform bounds based on worst-case assumptions about examples. Also, it is easy to compute VC bounds for any number of examples, assuming the VC dimension for the class is known. This makes VC bounds useful and convenient for large data sets, that is, data sets having thousands of examples. However, VC error bounds have some drawbacks: they are ineffective for smaller data sets, and they do not apply to some classifiers, such as nearest neighbor classifiers.

Transductive inference (Vapnik, 1998) is a training method that uses information provided by inputs of working examples in addition to information provided by training examples. The idea is to develop the best classifier for the inputs of the specific working examples at hand rather than develop a classifier that is good for general inputs and then apply it to the working examples. Transductive inference improves on general VC bounds by using the actual working example inputs, instead of a worst-case arrangement of inputs, to find the number of different assignments that classifiers in each training class can produce. The bounds are then used to select among classes, mediating a tradeoff between small classes that are more likely to have good generalization and large classes that are more likely to capture the dynamics of the training data.

The error bound presented here is designed to provide error bounds for data sets so small that other bounds are ineffective. Like transductive inference, the error bound presented here uses information provided by the inputs of the working examples in addition to information provided by the

training examples. But it also uses information provided by the training procedure rather than just the class of all classifiers that can be produced by the training procedure. However, it requires more computation than VC bounds. In fact, the required computation grows so quickly with data set size that the bound is practical only for small data sets. (Nearest neighbor classifiers are an exception to this limitation; this paper contains an efficient method to compute bounds for them.)

Normally, algorithm designers are concerned about how computation grows as problem size increases, focusing on asymptotic behavior as problem size goes to infinity. This paper does not focus on what happens as data set size goes to infinity—there are already many effective bounds for large data sets. This paper focuses on how much information we can wring out of small data sets. Small data sets do occur in practice, for example in early stage drug tests in medicine and in setting prices and making markets for ad-matching systems that have to deal with tiny markets out in the long tail of information domains. In short, there is "plenty of room at the bottom" for improved error bounds.

The error bound in this paper is based on the fact that if the training and working examples are generated independently and identically distributed (i.i.d.), then each partition of the complete set of training and working examples into a training set and a working set is equally likely. Several error bounds for machine learning are based on this principle. Examples include VC error bounds (Vapnik and Chervonenkis, 1971; Cristianini and Shawe-Taylor, 2000, Section 4.2, p. 55), error bounds for support vector machines (Vapnik, 1998, Chapter 8, pp. 339-343), compression-based error bounds (Littlestone and Warmuth, 1986), and uniform error bounds based on constraints imposed by patterns of agreement and disagreement among classifiers over the working inputs (Bax, 1999). For some other ways of developing and using this idea, refer to Audibert (2004), Blum and Langford (2003), Catoni (2003), Catoni (2004), Derbeko et al. (2003), and El-Yaniv and Gerzon (2005).

This paper is organized as follows. Section 2 introduces concepts and notation for the error bound. Section 3 presents the error bound. Section 4 introduces sampled filters, which reduce computation required for the bound. Section 5 analyzes speed and storage requirements for filters based on all partitions and for sampled filters. Section 6 introduces filters based on virtual partitioning, which do not require explicit computation over multiple partitions of the data into different training and working sets. Section 7 gives an efficient algorithm to compute an error bound for a 1-nearest neighbor classifier. Section 8 presents test results comparing bounds produced by different filters on a set of problems. Section 9 is a discussion of possible directions for future research.

2. Concepts and Notation

This paper concerns validation of classifiers learned from examples. Each example $Z = (X, Y)$ includes an input X and a class label $Y \in \{0,1\}$. We observe

$$(X_1, Y_1), ..., (X_t, Y_t), X_{t+1}, ..., X_{t+w}$$

that is, inputs and outputs of *t training examples* and just the inputs of *w working examples*. (We will use T to denote the set of training examples and W to denote the set of working examples.) A classifier g, which is a mapping from the input space for X to $\{0,1\}$, is developed using the observed data. Then classifier g is used to predict the working example outputs $Y_{t+1}, \ldots, Y_{t+w}$ associated with $X_{t+1}, \ldots, X_{t+w}$.

For any sequence of examples

$$c = (x_1, y_1), ...(x_{t+w}, y_{t+w}) \tag{1}$$

from the joint space of inputs and labels, define the *error* to be

$$E_c = \frac{1}{w} \sum_{i=t+1}^{t+w} I(g(x_i) \neq y_i),$$

where I is the indicator function—one if the argument is true and zero otherwise. Let

$$C = Z_1, ..., Z_{t+w}.$$

Examples in this *complete sequence* $Z_1 = (X_1, Y_1)$, ..., $Z_{t+w} = (X_{t+w}, Y_{t+w})$ are assumed to be drawn i.i.d. from an unknown joint distribution of inputs and labels.

The goal is to produce a PAC (probably approximately correct) bound on E_C, the error on complete sequence C.

The bounds in this paper consider different possible assignments to the unknown labels of the working set examples and use permutations of the complete sequence. So we introduce some notation around assignments and permutations. For any *assignment* $a \in \{0,1\}^w$ and any permutation σ of $\{1, \ldots, t+w\}$, let *C(a, σ)* be the sequence that results from assigning labels to the working examples in C:

$$\forall i \in \{1, ..., w\} : Y_{t+i} = a_i,$$

then permuting the sequence according to σ. Let *T(a, σ)* be the set consisting of the first t examples in *C(a, σ)*. Let *W(a, σ)* be the set consisting of the last w examples in *C(a, σ)*. Let *g(a, σ)* be the classifier produced by applying the training procedure with *T(a, σ)* as the training set and *W(a, σ)* as the working set. Let *E(a, σ)* be the error of *g(a, σ)* over *W(a, σ)*; in other words, let *E(a, σ)* be the error E as defined in Equation (1) that would result from using *C(a, σ)* as the complete sequence.

Let a^* be the actual (unknown) labels of the working examples. Let *Id* be the identity permutation. Then

$$E_C = E(a^*, Id).$$

3. First Error Bound and Algorithm

Section 3.1 presents the error bound. Section 3.2 shows how to compute the bound over partitions instead of permutations. Section 3.3 presents the algorithm for computing the bound.

3.1 Bound

Let *Id* be the identity permutation. Define a *likely set*

$$L = \{a \in \{0,1\}^w | P[E(a, \sigma) \geq E(a, Id)] > \delta\},$$

where the probability is over the uniform distribution of permutations σ of $\{1, \ldots, t+w\}$. The bound is

$$\max_{a \in L} E(a, Id). \tag{2}$$

Theorem 3.1

With probability at least $1 - \delta$,

$$E(a^*, Id) \leq \max_{a \in L} E(a, Id), \tag{3}$$

where the probability is over random complete sequences drawn i.i.d. from a joint input-label distribution.

Proof of Theorem 3.1

If $a^* \in L$, then Equation (3) holds. So it suffices to show that

$$P_C(a^* \notin L) \leq \delta,$$

where subscript C denotes probability over the distribution of complete sequences C. By the definition of L, the LHS is

$$= P_C[P_\sigma[E(a^*, \sigma) \geq E(a^*, Id)] \leq \delta | C],$$

where subscript σ denotes probability over the uniform distribution of permutations σ of $\{1, \ldots, t+w\}$. Convert the probability over the distribution of complete sequences to an integral over complete sequences:

$$\int_C I(P_\sigma[E(a^*, \sigma) \geq E(a^*, Id)] \leq \delta | C) p(C) dC,$$

where *p(C)* is the pdf of C. Since each permutation of a complete sequence is equally likely, we can replace the integral over complete sequences by an integral over sequences Q of $t+w$ examples followed by an average over permutations σ' of Q to form complete sets:

$$= \int_Q \frac{1}{(t+w)!} \sum_{\sigma'} I(P_\sigma[E(a*, \sigma) \geq E(a^*, \sigma')] \leq \delta | C = \sigma' Q) p(Q) dQ. \tag{4}$$

For each set Q, only δ $(t+w)!$ or fewer permutations σ' can rank in the top δ $(t+w)!$ of all $(t+w)!$ permutations for any statistic, including the statistic *E(a*, σ)*. So,

$$\forall Q : \sum_{\sigma'} I(P_\sigma[E(a^*, \sigma) \geq E(a^*, \sigma')] \leq \delta | Q) \leq \delta (t+w)!.$$

Substitute this inequality into Equation (4), to show it is

$$\leq \int_Q \frac{1}{(t+w)!} \delta (t+w)! p(Q) dQ.$$

Cancel terms $(t+w)!$ and integrate to get δ, completing the proof.

3.2 From Permutations to Partitions

For each assignment $a \in \{0,1\}^w$ and each size-t subset S of $\{1, \ldots, t+w\}$, let *T(a, S)* be the set (or multi-set) of examples in *C(a, Id)* indexed by entries of S, and let *W(a, S)* be the remaining examples in *C(a, Id)*. Refer to the pair *T(a, S)* and *W(a, S)* as the partition induced by S. Let *g(a, S)* be the classifier that results from training with *T(a, S)* as the training set and *W(a, S)* as the working set. Let *E(a, S)* be the error of *g(a, S)* over *W(a, S)*.

For each permutation σ of $\{1, \ldots, t+w\}$, let σ_i be the position in *C(a, Id)* of the example in position i in *C(a, σ)*. Note that

$$\{\sigma_1, ..., \sigma_t\} = S \Rightarrow (T(a,\sigma), W(a,\sigma)) = (T(a,S), W(a,S)).$$

For each S, there is the same number, $t!w!$, of permutations σ with $\{\sigma_1, \ldots, \sigma_t\} = S$, because there are $t!$ ways to order the elements of S in $\sigma_1, \ldots, \sigma_t$ and $w!$ ways to order the remaining elements of $\{1, \ldots, t+w\}$ in $\sigma_{t+1}, \ldots, \sigma_{t+w}$. Since there is this $t!w!$-to-one mapping from permutations σ to subsets S with *E(a, σ) = E(a, S)*, the probability over permutations in the definition of L can be replaced by the following probability over partitions induced by subsets S :

$$L = \{a \in \{0,1\}^w | P_S[E(a,S) \geq E(a,S^*)] > \delta\},$$

where the probability is uniform over size-t subsets S of $\{1, \ldots, t+w\}$, and $S^* = \{1, \ldots, t\}$. Hence, we can compute errors *E(a, S)* over size-t subsets S rather than over all permutations in order to compute bound in Equation (2).

3.3 Algorithm

Given an array C of t training examples followed by w working examples and a bound failure probability ceiling δ, Algorithm 3.3.1 returns a valid error bound with probability at least $1 - \delta$. The procedure *E(a, S, C)*, which is not shown, computes *E(a, S)* by assigning a to the labels of the last w entries in C, training a classifier using the entries of C indicated by S as the training set and the remaining entries as the working set, and returning the error of that classifier over that working set.

Algorithm 3.3.1

```
procedure bound(C, delta)
  // Variable bound stores the running max of errors for likely assignments.
  bound := 0;

  // Try all assignments.
  for (a in {0,1}^w)
    // Check if error is high enough to be a new max.
    if (E(a, {1, ..., t}, C) > bound)
      // Variable f[i] stores the frequency of E(a, S) = i/w.
      f[0 ... w] = 0;
      // Find error frequencies.
      for (S subset of {1, ..., t+w} with |S|=t) f[E(a, S, C)]++;
      // Variable tail stores the frequency of error
```

```
      //   greater than for S={1, ..., t}.
      tail := 0;
      // Sum the tail.
      for (i=E(a, {1, ..., t}, C) to w) tail += f[i];
      // If assignment is likely, increase bound.
      if (tail > delta) bound := E(a, {1, ..., t}, C);
    end if
  end for

  return bound;
end procedure
```

4. Sampled Filters and Ranking with Random Tie Breaking

The goal is to reject from L as many false assignments a as possible among those that have $E(a, Id) > E(a^*, Id)$, while only rejecting a^* in a fraction δ or fewer of cases. The bound process in the previous section rejects assignments a for which $E(a, S^*)$ is abnormally high among errors $E(a, S)$ over subsets S of $\{1, \ldots, t+w\}$, that is, among partitions of the complete sequence into training and working sets. For each assignment a, the process is equivalent to ranking all subsets S in order of $E(a, S)$, finding the fraction of subsets that outrank S^*, even with S^* losing all ties, and rejecting a if the fraction is δ or less. Call this filter the *complete filter*, because it compares S^* to all subsets S. This section introduces alternative filters that do not require computation over all subsets and a random tie breaking process that ranks S^* fairly among subsets S having the same error instead of having S^* lose all ties.

4.1 Sampled Filters

The complete filter is expensive to compute. To motivate thinking about alternative filters, note that any filter that accepts a* into L with probability at least 1-δ produces a valid bound. For example, a filter that simply makes a random determination for each assignment, accepting it into L with probability 1-δ, independent of any data about the problem at hand, still produces a valid error bound. Of course, this *random filter* is unlikely to produce a strong bound, because it does not preferentially reject assignments a that have high error $E(a, S^*)$.

The following *sampled filter*, based on errors $E(a, S)$ over a random sample of subsets S, rejects assignments with high error $E(a, S^*)$, and it is less expensive to compute than the complete filter. For each assignment a, generate a sample of n size-t subsets S of $\{1, \ldots, t+w\}$. Generate the sample by drawing subsets i.i.d. with replacement based on a uniform distribution over subsets, or generate the sample by drawing subsets i.i.d. without replacement based on a distribution that is uniform over subsets other than S^* and has zero probability for S^*. After drawing the sample by either method, add S^* to the sample. Then use the sample in place of the set of all subsets S in the algorithm, that is, accept assignment a if the fraction of subsets S in the sample with $E(a, S)$ at least $E(a, S^*)$ is greater than δ . Like the complete filter, this sampled filter has probability at most δ of rejecting the true assignment. Here is the proof for sampling with replacement. The proof for sampling without replacement is similar, and it is outlined after the proof for sampling with replacement.

Theorem 4.1.1

Let R be the set of all size-t subsets of $\{1, \ldots, t+w\}$. Let M be a set (or multi-set) of entries from R, drawn i.i.d. with replacement based on a uniform distribution over R. Let

$$L_M = \left\{a \in \{0,1\}^w | P_{S \in M \cup \{S^*\}}[E(a,S) \geq E(a,S^*)] > \delta\right\}$$

where the probability is uniform over all sets S in M. Then, with probability at least $1-\delta$,

$$E(a^*,S^*) \leq \max_{a \in L_M} E(a,S^*), \tag{5}$$

where the probability is over random complete sequences C drawn i.i.d. from a joint input-label distribution and over random subset samples M.

Proof of Theorem 4.1.1

If $a^* \in L_M$, then Equation (5) holds. So we will show

$$P_{C,M}(a^* \notin L_M) \leq \delta, \tag{6}$$

where the probability is over the joint distribution of complete sequences C and subset samples M. By the definition of L_M, the LHS is

$$= P_{C,M}[P_{S \in M \cup \{S^*\}}[E(a^*,S) \geq E(a^*,S^*)] \leq \delta | C].$$

Convert the probability over C into an integral over sequences Q of $t+w$ examples, followed by a probability over permutations σ' of Q to form complete sets:

$$\int_Q P_{\sigma',M}[P_{S \in M \cup \{S^*\}}[E(a^*,S) \geq E(a^*,S^*)] \leq \delta | C = \sigma' Q] p(Q) dQ, \tag{7}$$

where the first probability is over a joint distribution of σ' and M, with σ' drawn uniformly at random from permutations of $t+w$ elements and independently of M. For any fixed sequence Q, consider the expression from within Equation (7):

$$P_{\sigma',M}[P_{S \in M \cup \{S^*\}}[E(a^*,S) \geq E(a^*,S^*)] \leq \delta | C = \sigma' Q]. \tag{8}$$

Define multi-set

$$H(Q) = \{E(a^*,S) | S \in M \cup \{S^*\}\}.$$

Random draws of σ' and M make $H(Q)$ a multi-set of random values drawn i.i.d. from a uniform distribution over the set

$$U(Q) = \{E(a^*,S) | S \subseteq \{1,...,t+w\} \wedge |S| = t\}.$$

Since the elements of $H(Q)$ are drawn i.i.d., the probability that $E(a^*, S^*)$ ranks in the top $\delta |H(Q)|$ of the positions in a ranking of entries in $H(Q)$ is at most δ. Note that Equation (8) is this probability. So

$$\forall Q : P_{\sigma',M}[P_{S \in M \cup \{S^*\}}[E(a^*,S) \geq E(a^*,S^*)] \leq \delta | C = \sigma' Q] \leq \delta.$$

Substitute this inequality into Equation (7), showing that the LHS of Equation (6) is

$$\leq \int_Q \delta p(Q) dQ.$$

Integrate to get δ, completing the proof.

Now consider the case of sampling subsets without replacement:

Theorem 4.1.2

Let R' be the set of all size-t subsets of $\{1, \ldots, t+w\}$, except for S^*. Let M' be a set of entries from R', drawn i.i.d. without replacement based on a uniform distribution over R'. Let

$$L_{M'} = \left\{ a \in \{0,1\}^w | P_{S \in M' \cup \{S^*\}} [E(a,S) \geq E(a,S^*)] > \delta \right\},$$

where the probability is uniform over all sets S in M'. Then, with probability at least $1 - \delta$,

$$E(a^*, S^*) \leq \max_{a \in L_{M'}} E(a, S^*),$$

where the probability is over random complete sequences C drawn i.i.d. from a joint input-label distribution and over random subset samples M.

Proof of Theorem 4.1.2

The proof is almost the same as the proof of Theorem 4.1.1, substituting M' for M. The set *H(Q)* becomes a set of random variables drawn i.i.d. from *U(Q)* without replacement, rather than with replacement. But with or without replacement, the probability that *E(a*, S*)* ranks in the top δ $|H(Q)|$of the positions in a ranking of entries in *H(Q)* is at most δ. Otherwise, the proof is the same.

4.2 Random Tie Breaking

Both the complete filter and the sampled filter accept an assignment if the fraction of a set of subsets S with *E(a, S)* at least *E(a, S*)* is greater than δ. In essence, if other subsets S have the same error as S^*, then this rule errs on the side of safety by treating those subsets S as having greater error than S^*. This ensures that the bound is valid, but it makes the bound weaker than necessary. To close the gap, use random tie breaking to rank S^* at random among the subsets S that have *E(a, S) = E(a, S*)*. Let k be the number of subsets S with the same error as S^*, including S^* itself. Generate an integer uniformly at random in [1,k] to be the number of subsets S with the same error that rank at or above S^* after random tie breaking. If that number plus the number of subsets S with error greater than for S^* is a larger fraction of the partitions than δ, then accept the assignment.

5. Speed and Storage Requirements for Complete and Sampled Filters

Consider the storage requirements for the bound process. Since the maximum error *E(a, S)* over assignments in L is obtained by maintaining a running maximum as assignments are added to L, there is no need to store L explicitly. So the storage requirements are mild, including space for a data set, for two classifiers, and for training a classifier.

Using the complete filter to produce a bound requires time to train

$$O(2^w \begin{pmatrix} t+w \\ t \end{pmatrix})$$

classifiers. Using the sampled filter instead requires time to train

$$O(2^w n)$$

classifiers, where n is the number of sample partitions per assignment. Both types of filters can be computed in parallel, using different machines to filter different sets of assignments, each keeping a running maximum of *E(a, S*)* over accepted assignments, and then finishing by fanning in the maximum over the machines.

To reduce computation, evaluate assignments a for membership in L in decreasing order of *E(a, S*)*. When the first assignment a is accepted into L, return *E(a, S*)* as the error bound, and stop. To order assignments by *E(a, S*)*, train a classifier g, and apply it to each input in W to form the assignment with zero error. Invert that assignment to form the assignment with maximum error. Invert single elements of that assignment to produce assignments with the next greatest error rates. Then invert pairs of elements, then triples, etc. (This technique reduces computation only by a small fraction when the bound is effective; the reduction is only about half for an error bound of 0.5 and even less for smaller error bounds. On the other hand, it does contribute to "fast failure" when the bound is not effective.)

6. Virtual Partitions

Section 6.1 introduces the virtual partition filter. Section 6.2 describes the leave-one-out filter. Section 6.3 presents scoring functions. Section 6.4 suggests a scoring function for support vector machines.

6.1 Virtual Partition Filter

Define a *general likely set* L_h, based on some function h:

$$L_h = \{a \in \{0,1\}^w | P_S[h(a,S) \geq h(a,S^*)] > \delta\},$$

where the probability is over subsets S of $\{1, \ldots, t+w\}$. Define a *general error bound*

$$E_h(a,S) = \max_{a \in L_h} E(a,S^*).$$

If the filter

$$P_S[h(a,S) \geq h(a,S^*)] > \delta$$

can be computed for each assignment a without explicitly computing *h(a, S)* over some subsets S, then we call it a *virtual partition filter*.

6.2 Leave-One-Out Filter

For example, let *h(a, S)* be the number of leave-one-out errors in *W(a, S)*. (A leave-one-out error is an example in *C(a)* that has a different label than the closest other example in *C(a)*, with distance based on some metric over the input domain.) A filter based on leave-one-out errors excludes assignments *a* that cause an improbably large fraction of the leave-one-out errors in *C(a)* to be in the working set. Frequencies of leave-one-out errors in *W(a, S)* over subsets *S* can be computed without explicitly iterating over the subsets. The frequencies have a hypergeometric distribution—if there are *m* leave-one-out errors in *C(a)*, then

$$P_S[h(a,S) = j] = \frac{\left(\begin{array}{c} m \\ j \end{array} \right) \left(\begin{array}{c} t+w-m \\ w-j \end{array} \right)}{\left(\begin{array}{c} t+w \\ w \end{array} \right)},$$

where the probability is uniform over size-t subsets S of $\{1, \ldots, t+w\}$.

Compute the filter for each assignment as follows. Set the labels of $Z_{t+1}, \ldots, Z_{t+w}$ according to *a*. Then compute the number of leave-one-out errors in *C(a)*; call it *m*. Next, compute frequencies:

$$\forall j \in \{\max(0, m-t), ..., \min(w, m)\} : f_j = \left(\begin{array}{c} m \\ j \end{array} \right) \left(\begin{array}{c} t+w-m \\ w-j \end{array} \right).$$

Let *j*=h(a,S*)*, that is, the number of leave-one-out errors in *W(a, S*)*. Let

$$v = \sum_{j=j^*}^{\min(w,m)} f_j.$$

For random tie breaking, subtract from v a number drawn uniformly at random from $[0, f_{j*} - 1]$.Then divide by the number of partitions:

$$\left(\begin{array}{c} t+w \\ w \end{array} \right).$$

If the result is δ or less, then reject assignment *a*.

6.3 Scoring Functions

In general, let *s* be a *scoring function* on examples in *C(a)* that returns an integer. Let

$$h(a,S) = \sum_{Z \in W(a,S)} s(Z).$$

Let *n(Z)* be the nearest neighbor to Z in $T \cup W - \{Z\}$. For example, when counting leave-one-out errors,

$$s(Z) = \begin{cases} 1 & if\ n(Z).y \neq Z.y \\ 0 & otherwise \end{cases}$$

where *.y* after an example denotes the output, *Y*, for the example.

Another useful scoring function counts leave-one-out errors caused by example Z:

$$s(Z) = |\{Q \in T \cup W | n(Q) = Z \text{ and } Q.y \neq Z.y\}|.$$

For scoring functions, such as this one, that have a range other than {0,1}, the hypergeometric distribution does not apply. However, dynamic programming allows efficient computation of the frequencies, as follows. Let

$$c_{ijk} \equiv \left| \left\{ A \subseteq \{1,...,i\} \,|\, |A| = j \text{ and } \sum_{b \in A} s(Z_b) = k \right\} \right|,$$

that is, the number of size-*j* subsets of the first *i* examples in C(a) that have sum of scores *k*. The base cases are

$$\forall (j,k) : c_{0jk} = 0$$

except

$$c_{000} = 1,$$

and

$$\forall k < 0 : c_{ijk} = 0.$$

The recurrence is

$$c_{ijk} = c_{i-1,j,k} + c_{i-1,j-1,k-s(i)},$$

where *s(i)* is the score of example *i*. The frequencies are:

$$P_S[h(a,S) = k] = c_{t+w,w,k}.$$

Computing an error bound using virtual partitions requires $O(2^w poly(t+w))$ time because the scoring function is computed for each of the 2^w assignments. (This assumes O(poly(t+w)) time to compute the filter for each assignment.) The computation requires O(poly(t+w)) space since each assignment can be filtered without reference to others, and the maximum error of a likely assignment can be maintained using a single variable.

6.4 Scoring Functions for SVMs

Now consider filters with virtual partitions for support vector machines (SVMs). A leave-one-out filter can require much computation—for each assignment, training separate SVMs with each example held out in order to compute the number of leave-one-out errors. Joachims discovered a method to bound the number of leave-one-out errors based on the results of training a single SVM on all examples. The method is called εα-estimation (Joachims, 2002, Ch. 5). Computing the εα-estimator involves producing a set of examples that are potential leave-one-out errors. The set can be used as the basis for a filter that is binary-valued—each example in the set scores one and each other example scores zero. The εα-estimation procedure can also be used as the basis of a more complex filter, because it computes scores for examples before using a threshold to determine which ones are in the set. So the scores (or discretized scores) can be used directly as the scoring function for a filter.

7. Efficient Computation of Error Bounds for 1-Nearest Neighbor Classifiers

When using virtual partitions based on leave-one-out errors to produce an error bound for a 1-nearest neighbor classifier, there is a way to avoid iterating over all assignments to compute the bound. Avoiding this iteration leads to an efficient method to compute an error bound for a 1-nearest neighbor classifier, that is, a method that requires time polynomial in the size of the problem. This section begins with some preliminary concepts before presenting the recurrences and dynamic programming algorithm. Next there is a small example to illustrate the algorithm. Then there are details on how to compute the recurrences efficiently. This section ends with a note on how to extend the algorithm to improve the bounds by allowing random tie breaking for ranking.

7.1 Preliminaries and Concepts

To begin, ensure that each example has a unique nearest neighbor by randomly perturbing the inputs of examples that tie to be nearest neighbors to any example. Perturb by so little that no new nearest neighbor can be introduced. Repeat until each example has a unique minimum distance to another example.

Lemma 7.1.1

This form of random tie breaking makes it impossible for a cycle of three or more examples to have each example in the cycle the nearest neighbor of the next.

The proof is by contradiction. Let $n(Z)$ be the nearest neighbor of example Z in $T \cup W - \{Z\}$. Suppose there is a cycle of examples $Z_1, \ldots, Z_m, Z_1$ with $m > 2$ and each example the nearest neighbor of the next, that is,

$$\forall i \in \{1,...,m\} : Z_i = n(Z_{i+1}),$$

and

$$Z_m = n(Z_1)$$

Let d be the distance metric over example inputs. Then

$$d(Z_1,Z_2) \leq ... \leq d(Z_m,Z_1) \leq d(Z_1,Z_2).$$

For cycles greater than length two, the tie breaking makes equality impossible. So we have

$$d(Z_1,Z_2) < ... < d(Z_m,Z_1) < d(Z_1,Z_2).$$

Having the same distance on the left and right implies that the distance from the first example to the second is greater than itself, which is impossible, completing the proof.

Let G be a directed graph with each example in $Z_1, \ldots, Z_{t+w}$ a node and with edges

$$\{(n(Z_1),Z_1),...,(n(Z_{t+w}),Z_{t+w})\},$$

that is, an edge to each example from its nearest neighbor. By Lemma 7.1.1, G has no cycles of length greater than two. So G is a directed tree or forest, plus some directed edges that complete length two cycles by going back along tree edges. Let F be a directed forest created by removing from G one edge from each two-cycle. The algorithm to efficiently compute an error bound uses dynamic programming, starting at the leaves of F and working up to the root or roots.

7.2 Recurrences and Algorithm

Let *F(k)* be the subtree of F rooted at example Z_k, that is, Z_k and all nodes that can be reached by following directed sequences of edges from Z_k. Let *A(i,j,k)* be the subset of assignments in $\{0,1\}^w$ that have i leave-one-out errors among the training examples in *F(k)* and j leave-one-out errors among the working examples of *F(k)*. Let *n(Z,T)* be the nearest neighbor of example Z among the training examples. Define

$$e_{ijky} = \max_{a \in A(i,j,k)} |\{Z \in F(k) \cap W | Z.y = y \text{ and } Z.y \neq n(Z,T).y\}|,$$

that is, it is the maximum number of working examples in the subtree of F rooted at example Z_k that are misclassified by their nearest training examples, with the maximum being over assignments that have i leave-one-out errors on the training examples in the subtree, j leave-one-out errors on the working examples in the subtree, and label y on example Z_k. If there are no such assignments, then define

$$e_{ijky} = -1,$$

to signify that the value is "undefined."

The base cases are leaves of F. For a leaf example Z_k that is in T and has label y,

$$e_{00ky} = 0,$$

and, for all other combinations of i, j, and y,

$$e_{ijky} = -1.$$

For a leaf example Z_k that is in W and has label y,

$$e_{00ky} = 0,$$

$$e_{0,0,k,1-y} = 1,$$

and, for all other combinations of i, j, and y,

$$e_{ijky} = -1.$$

Before defining the general recurrence, we first define some terms that express how interactions between examples and their parent examples in F influence the numbers of leave-one-errors in T and W and the error. Let Z_i be an example, Let $y_i = Z_i.y$, let Z_k be the parent of Z_i in F, and let $y_k = Z_k.y$. Define

$$c_T(i,y_i,k,y_k) = \begin{cases} 1 & Z_i \in T \text{ and } y_i \neq y_k \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

to count whether Z_k having label y_k causes example Z_i to be a leave-one-out error in T. Define

$$d_T(i,y_i,k,y_k) = \begin{cases} 1 & Z_k \in T, y_i \neq y_k, \text{ and } Z_i = n(Z_k) \\ 0 & \text{otherwise} \end{cases}, \tag{10}$$

to count whether example Z_k having label y_k causes example Z_k to be a leave-one-out error in T. Define

$$c_W(i,y_i,k,y_k) = \begin{cases} 1 & Z_i \in W \text{ and } y_i \neq y_k \\ 0 & \text{otherwise} \end{cases}, \tag{11}$$

to count whether example Z_k having label y_k causes example Z_i to be a leave-one-out error in W if it has label y_i. Define

$$d_W(i,y_i,k,y_k) = \begin{cases} 1 & Z_k \in W, y_i \neq y_k, \text{ and } Z_i = n(Z_k) \\ 0 & \text{otherwise} \end{cases}, \tag{12}$$

to count whether example Z_i having label y_i causes example Z_k to be a leave-one-out error in W if it has label y_k. Define $n_T(Z)$ to be the nearest neighbor of example Z in T. Define

$$h(k,y_k) = \begin{cases} 1 & Z_k \in W \text{ and } y_k \neq n_T(Z_k).y \\ 0 & \text{otherwise} \end{cases},$$

to count whether example Z_k having label y_k causes example Z_k to be misclassified by its nearest training example.

Let *r(Z)* be the parent of example Z in F. Define

$$B(k) \equiv \{b | r(Z_b) = Z_k\},$$

that is, *B(k)* is the set of positions in C of the children of Z_k in F. Let $y_b = Z_b.y$. Then

$$e_{ijky} = \max_{\substack{\{(i_b,j_b,b,y_b) : b \in B(k), e_{i_b j_b b y_b} \neq -1\} \text{ s.t.} \\ \sum_{b \in B(k)} [i_b + c_T(b,y_b,k,y) + d_T(b,y_b,k,y)] = i, \text{ and} \\ \sum_{b \in B(k)} [j_b + c_W(b,y_b,k,y) + d_W(b,y_b,k,y)] = j}} h(k,y_k) + \sum_{b \in B(k)} e_{i_b j_b b y_b}. \tag{13}$$

Let *A(i,j)* be the subset of assignments in $\{0,1\}^w$ that have *i* leave-one-out errors on training examples and *j* leave-one-out errors on working examples. Define

$$v_{ij} \equiv \max_{a \in A(i,j)} |\{Z \in W | Z.y \neq n(Z,T).y\}|,$$

that is, the maximum error over assignments that have *i* leave-one-out errors on training examples and *j* leave-one-out errors on working examples. If there are no such assignments, then let

$$v_{ij} = 0.$$

Define

$$B \equiv \{b | Z_b \text{ is a root of } F\}.$$

Then

$$v_{ij} = \max_{\substack{\{(i_b,j_b,b,y_b) : b \in B, e_{i_b j_b b y_b} \neq -1\} \text{ s.t.} \\ \sum_{b \in B} i_b = i, \text{and} \\ \sum_{b \in B} j_b = j}} \sum_{b \in B} e_{i_b j_b b y_b}.$$

Let u_{ij} be the probability that j or more leave-one-out errors are in $W(S)$ for a random size-t subset S of $\{1, \ldots, t+w\}$, given that there are $i+j$ leave-one-out errors in $T \cup W$. Then

$$u_{ij} = \sum_{z=j}^{\min(w,i+j)} \frac{\binom{i+j}{z}\binom{t+w-i-j}{w-z}}{\binom{t+w}{w}}. \tag{14}$$

For a given δ, the value

$$\max_{u_{ij} \geq \delta} v_{ij} \tag{15}$$

bounds the number of working examples misclassified by their training examples, with probability at least 1-δ.

Note that the recurrence for each term e_{ijky} depends on terms e_{ijby} for all $b \in B(k)$. So produce an ordering σ_k on examples in C that places all children before their parents in F. Compute terms e_{ijky} in that order, to ensure that each term is computed prior to computing any term that depends on it. Next compute terms v_{ij} based on terms e_{ijby} for all $b \in B$. Then compute values u_{ij} using Equation (14), and compute the bound according to Equation (15).

7.3 Example of Computing Values e_{ijky}

Consider a small example to demonstrate the recurrence for e_{ijky}. Use the following examples:

example	input	output	set
0	11.1	0	T
1	12.3	1	T
2	15.6	?	W

The graph G of nearest neighbors has edges (0,1), (1,0), and (1,2). Removing the first edge produces a tree F, with node 1 as root and the other nodes as leaves. An ordering that places children before parents in F is (0, 2, 1). So compute terms e_{ijky} for $k = 0$, then $k = 2$, then $k = 1$.

For $k = 0$, node 0 is a leaf in F. Since example 0 is in T and has output $y_0 = 0$,

$$e_{0000} = 0,$$

meaning that, in the single-node subtree consisting of node 0, there are no leave-one-out errors in T or in W, and there are no examples in W misclassified by nearest neighbors in T. For all other i, j, and y

$$e_{ij0y} = -1.$$

For $k = 2$, node 2 is a leaf in F. Since example 2 is in W, it may have output $y_2 = 0$ or $y_2 = 1$. If $y_2 = 0$, then example 2 is misclassified by its nearest neighbor in T, which is example 1. So

$$e_{0020} = 1.$$

If $y_2 = 1$, then example 2 is properly classified by example 1, so

$$e_{0021} = 0.$$

For k = 1, example 1 is in T, and y_1 = 1. Node 1 has two children in G—nodes 0 and 2. For child node 0, only the term e_{000} is defined. For child node 2, terms e_{0020} and e_{0021} are defined. Each pair of terms with one from each child node can produce a term for node 1.

Begin with the pair e_{000} and e_{0020}. Relationships between node 1 and each child node contribute to the term. For the relationship between node 1 and node 0, the values are n = 0, y_n = 0, k = 1, and y_k = 1. With these arguments:

1. c_T(0,0,1,1) = 1 because example 1 (the parent) misclassifies example 0 (the child), causing a leave-one-out error in T.
2. d_T(0,0,1,1) = 1 because example 0 (the child) misclassifies example 1 (the parent), causing a leave-one-out error in T.
3. c_W(0,0,1,1) = 0 and d_W(0,0,1,1) = 0 because neither example is in W.

For the relationship between node 1 and node 2, the values are n = 2, y_n = 0, k = 1, and y_k = 1. With these arguments:

1. c_T(2,0,1,1) = 0 because example 2 (the child) is not in T.
2. d_T(2,0,1,1) = 0 because example 2 (the child) does not classify example 1 (the parent).
3. c_W(2,0,1,1) = 1 because example 1 (the parent) misclassifies example 2 (the child), causing a leave-one-out error in W.
4. d_W(2,0,1,1) = 0 because example 1 (the parent) is not in W.

The resulting term has

$$\begin{aligned} i &= [i_0 + c_T(0,0,1,1) + d_T(0,0,1,1)] + [i_2 + c_T(2,0,1,1) + d_T(2,0,1,1)] \\ &= [0+1+1] + [0+0+0] = 2 \end{aligned}$$

and

$$\begin{aligned} j &= [j_0 + c_W(0,0,1,1) + d_W(0,0,1,1)] + [j_2 + c_W(2,0,1,1) + d_W(2,0,1,1)] \\ &= [0+0+0] + [0+1+0] = 1. \end{aligned}$$

So the term is e_{2111}. The value is

$$e_{2111} = h(1,1) + e_{0000} + e_{0020} = 0 + 0 + 1 = 1.$$

(The value of h(1,1) is zero since example 1 is not in W.) The value e_{2111} = 1 means that, in the subtree of F rooted at node 1, that is, in F, it is possible to have two leave-one-out errors in T, one leave-one-out error in W, and one example in W misclassified by its nearest neighbor in T.

Now consider the other pair of terms from children, e_{0000} and e_{0021}. The relationship between node 1 and node 0 is the same as for the previous pair. The relationship between node 1 and node 2 changes because now y_2 = 1, so n = 2, y_n = 1, k = 1, and y_k = 1. With these arguments:

1. $c_T(2,1,1,1) = 0$ because example 2 (the child) is not in T.
2. $d_T(2,1,1,1) = 0$ because example 2 (the child) does not classify example 1 (the parent).
3. $c_W(2,1,1,1) = 0$ because example 1 (the parent) properly classifies example 2 (the child), causing no leave-one-out error in W.
4. $d_W(2,1,1,1) = 0$ because example 1 (the parent) is not in W.

The resulting term has

$$\begin{aligned} i &= [i_0 + c_T(0,0,1,1) + d_T(0,0,1,1)] + [i_2 + c_T(2,1,1,1) + d_T(2,1,1,1)] \\ &= [0+1+1] + [0+0+0] = 2 \end{aligned}$$

and

$$\begin{aligned} j &= [j_0 + c_W(0,0,1,1) + d_W(0,0,1,1)] + [j_2 + c_W(2,1,1,1) + d_W(2,1,1,1)] \\ &= [0+0+0] + [0+0+0] = 0. \end{aligned}$$

So the term is e_{2011}. The value is

$$e_{2011} = h(1,1) + e_{0000} + e_{0021} = 0+0+0 = 0,$$

which means that it is possible to have two leave-one-out errors in T, zero leave-one-out errors in W, and zero examples in W misclassified by nearest neighbors in T. Other than e_{2111} and e_{2011}, for all other i, j, and y

$$e_{ij1y} = -1.$$

This completes the computation of e_{ijky} values for this problem.

7.4 Efficient Computation

The bound can be computed using storage and time O(poly(t+w)). Computing values e_{ijky} and v_{ij} directly from the recurrences is inefficient. First consider values e_{ijky}. Recurrence (13) handles terms for all children of example Z_k at the same time. To improve efficiency, accumulate terms from one child at a time, as follows. Define $e_{\bullet\bullet ky}$ to be the "slice" of values with the specified k and y and all values of i and j. To compute each slice, iterate through children Z_b of Z_k, using a slice-sized array *prev* to store the accumulation over terms from children before Z_b and a slice-sized array *next* to store the accumulation of terms from children up to and including Z_b. In other words, when the iteration begins for child Z_{b^*}, $prev_{ij}$ is the value that e_{ijky} would have if the subtree in F rooted at k had children

$$\bigcup_{\{b \in B(k) | b < b^*\}} Z_b.$$

And when the iteration ends for child Z_{b^*}, $next_{ij}$ is the value that e_{ijky} would have if the subtree in F rooted at k had children

$$\bigcup_{\{b \in B(k) | b \leq b^*\}} Z_b.$$

Compute the iteration for child Z_{b^*} according to the recurrence:

$$next_{ij} = \max_{\substack{(i_a, j_a, i_{b^*}, j_{b^*}, y_{b^*}) : prev_{i_a j_a} \neq -1, e_{i_{b^*} j_{b^*} b^* y_{b^*}} \neq -1, \\ i_a + i_{b^*} + c_T(b^*, y_{b^*}, k, y) + d_T(b^*, y_{b^*}, k, y) = i, and \\ j_a + j_{b^*} + c_W(b^*, y_{b^*}, k, y) + d_W(b^*, y_{b^*}, k, y) = j.}} prev_{i_a j_a} + e_{i_{b^*} j_{b^*} b^* y_{b^*}}.$$

The base cases for this recurrence are the definitions for values of *prev* for the first child. By the definition of *prev*, these values should treat Z_k as a leaf in F. So use the base case values given previously for terms e_{ijky} for leaves in F to initialize *prev*.

Algorithm 7.4.1 computes an error bound efficiently. The inputs are:

1. Z – An array of examples, ordered such that children in F come before their parents. (The variables Z[k].x and Z[k].y store (X_k, Y_k), the input and output for example k.)
2. B – An array of arrays B, with B[k] the array of indices b such that Z[b] is a child of Z[k] in F.
3. R – An array of indices b such that Z[b] is a root in F.
4. u – An array with u[i][j] = u_{ij} as defined in Equation (14).
5. delta – The acceptable probability of the bound being invalid, δ.

The algorithm uses subprocedures:

1. cT, dT, cW, and dW – As defined in Equations (9) to (12).
2. nT – Returns the nearest neighbor to an example among examples in T, that is, $n_T(Z)$.

Algorithm 7.4.1

```
procedure bound(Z, B, R, u, delta)
  e[0...t][0...w][0...t+w][0...1] := -1;
    // Initialize all e[][][][] values to 1.

  // Compute slices, one for each example and assignment to the
  //   label of the example.
  for ((k, yk) in {0,...,t+w} x {0,1})
    if (Z[k] in T and yk != Z[k].y) continue;
      // Impossible assignment, so skip it.

    prev[0...t][0...w] := -1;
    next[0...t][0...w] := -1;

    // Initialize prev[0][0].
```

```
  if (Z[k] in T) prev[0][0] :=  0;
  if (Z[k] in W)
    Z[k].y = yk;
    if (nT(Z[k]) != Z[k].y) prev[0][0] := 1; else prev[0][0] := 0;
  end if

  // Compute the contribution for each child b of k in F.
  for ((b, yb) in B[k] x {0,1})
    if (Z[b] in T and yb != Z[b].y) continue;
      // impossible assignment, so skip it.
    if (Z[b] in W) Z[b].y := yb;
    di := cT(b,k) + dT(b,k);
    dj := cW(b,k) + dW(b,k);

    for ((i, j) in {0,...,t} x {0,...,w} such that e[i][j][b][yb] != -1)
      for ((ii, jj) in {0,...,t} x {0,...,w} such that prev[ii][jj] != -1)
        next[ii + i + di][jj + j + dj] = max(next[ii + i + di][jj + j + dj],
                                             prev[ii][jj] + e[i][j][b][yb]);
      end for (ii, jj)
    end for (i, j)

    // Prepare to compute contribution for next child b of k in F.
    prev := next;
    next[0...t][0...w] := -1;
  end for (b, yb)

  // Copy the slice into e[][][][].
  for ((i, j) in {0,...,t}x{0,...,w}) e[i][j][k][yk] = prev[i][j];
end for (k, yk)

// Combine roots: treat each root as a child of a virtual super-root.
prev[0...t][0...w] := -1;
next[0...t][0...w] := -1;

prev[0][0] := 0; // The virtual super-root introduces no errors.

// Accumulate terms over roots b in R.
for ((b, i, j) in R x {0,...,t} x {0,,w})
  m := max(e[i][j][b][0], e[i][j][b][1]);

  if (m!=-1)
    for ((ii, jj) in {0,...,t} x {0,...,w} such that prev[ii][jj]!=-1)
      next[ii + i][jj + j] := max(next[ii + i][jj + j], prev[ii][jj] + m);
    end for (ii, jj)
  end if
```

```
    prev = next;
    next[0...t][0...w] := -1;
  end for (b, i, j)

  // Maximize v over feasible u to produce a bound on error count.
  v = prev;

  mx := 0;

  for ((i, j) in {0,...,t}x{0,...,w})
    if (u[i][j] >= delta) mx = max(mx, v[i][j])
  end for (i, j)

  return mx;
end procedure
```

Appendix A contains Java code to compute error bounds by this procedure. The code uses $O(t^2w^2(t+w))$ time and O($tw(t+w)$) storage. Appendix A also contains a note on how to use less storage.

7.5 Random Tie Breaking for Ranking

Random tie breaking for ranking, as defined in Section 4 and applied to virtual partitions in Section 6, cannot be applied to the bound returned by the algorithm above. The error count that is the bound corresponds to an assignment that maximizes the error of using training examples to classify working examples, subject to the constraints that there are i leave-one-out errors on training examples, there are j leave-one-out errors on working examples, and the counts i and j make the assignment likely even without random tie breaking. It is valid to apply random tie breaking to the assignment behind the error count that is a candidate for the bound, as explained in Section 6. However, if random tie breaking declares the assignment unlikely, then we are left without a next candidate for the bound.

To use random tie breaking, the algorithm needs to store all candidates rather than just the maximum error count candidate in each variable e_{ijky} and v_{ij}. So instead of storing a single maximum value in each variable, store a vector indexed by error counts, with values indicating how many partial assignments (for variables e_{ijky}) or assignments (for variables v_{ij}) produce each error count. Follow the structure of the maximization algorithm, but replace maximization by accumulating candidates. Compute values u_{ij} as in the maximization algorithm above, and use those values to determine which vectors v_{ij} count potentially likely candidates for the bound. (Only the combinations of i and j that are likely without random tie breaking are potentially likely with random tie breaking.) . Then iterate through possible error counts, in descending order. For each error count, for each candidate counted by a vector of potentially likely candidates, apply random tie breaking to the candidate. If the candidate is likely, then return it as the bound.

8. Tests

This section presents results of tests for bounding methods developed in this paper. First there are tests comparing different bounding methods. Next, there are tests to examine the joint frequencies of errors and error bounds. Then there are tests to explore the effect of working set size on error bounds.

8.1 Comparing Bounding Methods

Here are results of tests to compare different bounding methods on 1-nearest neighbor classification for different types of data. The different bounding methods are:

1. Complete – Use the complete filter.
2. 100 – Use a sample filter with 100 sample partitions.
3. 1000 – Use a sample filter with 1000 sample partitions.
4. 10,000 – Use a sample filter with 10,000 sample partitions.
5. LOO – Use virtual partitioning, with a filter that scores one for each leave-one-out error.
6. Double – Use virtual partitioning, with a filter that scores one for each example that is misclassified by both of its two nearest neighbors in $T \cup W$.

All bounding methods use ranking with random tie breaking, as explained in Sections 4 and 6. The sample filters draw partitions without replacement.

For each type of data, there is a table of results. Each row holds results for a different value of the bound certainty parameter δ. The first column of each table shows errors from using training data as a 1-nearest neighbor classifier on working data. The other columns show differences between the bounds on error and actual error for different bounding methods.

Each cell shows a mean and standard deviation over 1000 trials. The cells in the "Error" column show mean and standard deviation of errors. The cells in subsequent columns show mean and standard deviation of difference between bound and error. For example, suppose the error has mean 0.3 and standard deviation 0.4, and a bounding method has mean 0.1 and standard deviation 0.0. This indicates that the error averages 0.3 over the 1000 trials, and the error varies quite a bit, but the bound is always exactly 0.1 greater than the actual error.

Note that the standard deviations displayed in cells are standard deviations of the values over 1000 trials. They are not standard deviations of the estimates of the means of values over 1000 trials, that is, their large sizes do not indicate uncertainty about the accuracy of the means. Since there are 1000 trials, those standard deviations are about 1/33 of the ones shown, indicating that most differences in means for different bounding methods in the tests below are statistically significant.

Each row of each table is based on the same 1000 trials, but different rows are based on different sets of trials. For each trial, a size $t+w$ subset of examples is selected at random from a data set. A size-t subset is selected at random to form the training set T, and the remaining w examples form the working set W. The error is computed, and each bounding method is applied to (T,W) to compute an error bound. The error is subtracted from each bound, and the differences are accumulated into the statistics for the bounding methods. In each row, we show the least mean among bounding methods in bold print.

δ	Error	Complete	100	1000	10,000	LOO	Double
0.1	0.0±0.0	0.21±0.096	0.23±0.10	0.21±0.097	0.21±0.10	0.25±0.18	$\mathbf{0.11 \pm 0.24}$
0.2	0.0±0.0	$\mathbf{0.068 \pm 0.11}$	0.085±0.12	0.068±0.11	0.072±0.11	0.10±0.18	0.079±0.20
0.3	0.0±0.0	0.0085±0.045	0.011±0.050	$\mathbf{0.0065 \pm 0.04}$	0.007±0.041	0.042±0.14	0.051±0.16

Table 1: Bound Methods Compared on Iris Data

All tests ran on an Apple Macintosh, with dual 1.42 GHz PowerPC processors and 512 MB of RAM. The longest-running tests were for the data set involving diabetes among Pima Indians, with t = 200 training examples and w = 15 working examples. The tests for δ = 0.1, δ = 0.2, and δ = 0.3 ran concurrently, taking about a day and a half for the 3000 trials, or about a minute per trial.

8.1.1 IRIS DATA

Table 1 has results for a data set involving iris classification. The data set is from the repository of data sets for machine learning maintained by the University of California at Irvine, which is available online. The data set contains examples for three types of iris; we use only the examples for the first two types in order to have binary classification problems. This leaves 100 examples, with 50 from each class. Each example has four input dimensions. We use t = 40 training examples and w = 4 working examples for each trial. The iris data are easy to classify, as indicated by the fact that the errors are always zero.

For δ = 0.1, the best method is virtual partitioning with a filter based on whether the two nearest neighbors to an example both misclassify the example. To understand why this filter is effective for data sets that are easy to classify, imagine a lone working example in the midst of many training examples that all have the same label. Suppose an assignment gives the opposite label to the working example. The two nearest neighbors are both training examples with the other label, so the filter recognizes the working example. On the other hand, even if the working example is the nearest neighbor of several training examples, the filter ignores the fact that the working example misclassifies those examples, because their next-nearest neighbors are other training examples with the same label. Contrast this with a filter based on leave-one-out errors. This filter would recognize the incorrectly labeled working example, but it would also recognize any nearby training examples that had the working example as nearest neighbor.

The best method for δ = 0.2 uses a complete filter. The best method for δ = 0.3 uses a sample size of 1000. For all values of δ, methods using sampled filters based on 1000 and 10,000 partitions perform about as well as the method using a complete filter—the differences between them are not statistically significant. The method using a sampled filter based on 100 partitions performs slightly worse, indicating that using fewer samples for ranking allows into the likely set some assignments with high error on the working set that would be rejected by using more samples. In general, more samples give stronger bounds, but at the cost of added computation.

8.1.2 DIABETES DATA

Table 2 has results for data involving diabetes in Pima Indians. This data set is also from the UC Irvine repository. The inputs have different scales, so we normalize the data, translating and scaling each input dimension to make each input dimension have mean zero and standard deviation one. There are 768 examples, with 500 from one class and 268 from another. There are eight input

δ	Error	100	1000	LOO
0.1	0.32 ± 0.12	0.23 ± 0.13	$\mathbf{0.20 \pm 0.13}$	0.26 ± 0.15
0.2	0.31 ± 0.12	0.16 ± 0.13	$\mathbf{0.15 \pm 0.13}$	0.20 ± 0.16
0.3	0.31 ± 0.12	0.12 ± 0.13	$\mathbf{0.11 \pm 0.13}$	0.16 ± 0.15

Table 2: Bound Methods Compared On Diabetes Data

δ	Error	100	1000	LOO	Double
0.1	0.070 ± 0.085	0.22 ± 0.11	$\mathbf{0.20 \pm 0.10}$	0.27 ± 0.14	0.29 ± 0.17
0.2	0.060 ± 0.074	0.13 ± 0.094	$\mathbf{0.12 \pm 0.093}$	0.19 ± 0.14	0.22 ± 0.17
0.3	0.068 ± 0.080	0.084 ± 0.097	$\mathbf{0.074 \pm 0.096}$	0.13 ± 0.14	0.18 ± 0.17

Table 3: Bound Methods Compared on Data with a Linear Class Boundary

δ	Error	100	1000	LOO	Double
0.1	0.18 ± 0.13	0.27 ± 0.15	$\mathbf{0.24 \pm 0.14}$	0.31 ± 0.17	0.39 ± 0.12
0.2	0.18 ± 0.12	0.17 ± 0.14	$\mathbf{0.16 \pm 0.14}$	0.22 ± 0.17	0.32 ± 0.18
0.3	0.16 ± 0.11	0.12 ± 0.13	$\mathbf{0.11 \pm 0.13}$	0.18 ± 0.17	0.28 ± 0.18

Table 4: Bound Methods Compared on Data with a Nonlinear Class Boundary

dimensions. We use $t = 200$ training examples and $w = 15$ working examples for each trial. We use 1-nearest neighbor classification.

The error indicates that this is a challenging data set for 1-nearest neighbor classification; even a classifier that always returns the label of the class with 500 examples of the 768 in the data would have average error about 0.35. The bounding method that uses 1000 partitions in a sampled filter performs best for all three values of δ. On average, that method adds error rate margins of 20% for $\delta = 0.1$, about 15% for $\delta = 0.2$, and about 11% for $\delta = 0.3$.

8.1.3 DATA WITH A LINEAR CLASS BOUNDARY

Table 3 has results for randomly generated data. The data consist of 1100 examples drawn uniformly at random from a three-dimensional input cube with length one on each side. The class label is zero if the input is from the left half of the cube and one if the input is from the right half of the cube. For these tests, there are $t = 100$ training examples and $w = 10$ working examples, using 1-nearest neighbor classification. Once again, the method that uses a sample filter with 1000 partitions in the sample outperforms the other methods in the test. On average, the method adds error rate margins of 20% for $\delta = 0.1$, about 12% for $\delta = 0.2$, and about 7.5% for $\delta = 0.3$.

8.1.4 DATA WITH A NONLINEAR CLASS BOUNDARY

Table 4 shows results for randomly generated data with a nonlinear class boundary. The data have the same characteristics as in the previous test, except that each class label is determined by the XOR of whether the input is in the left half of the cube, the bottom half of the cube, and the front half of the cube. In other words, the cube is cut into eight sub-cubes, and each sub-cube has a different class than the three sub-cubes with which it shares a side. This class scheme adds some

	Bound										
Error	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.0			1	17	95	45	4				
0.1				41	187	88	1				
0.2			1	35	190	66	2				
0.3			1	19	95	36					
0.4				10	34	8					
0.5				1	14	4					
0.6				2	2	1					
0.7											
0.8											
0.9											
1.0											

Table 5: Bounds vs. Errors for $\delta = 0.1$

error. As in the previous test, the method that uses a sample filter with 1000 partitions in the sample outperforms the other methods in the test. The method adds error rate margins that are higher than for the previous test, that is, about 24% for $\delta = 0.1$, about 16% for $\delta = 0.2$, and about 11% for $\delta = 0.3$.

8.1.5 COMPARISON TO BOUNDS BASED ON VC DIMENSION

The test results in tables 1 to 4 show that error bounds based on worst likely assignments can be effective for small data sets. Compare these bounds to error bounds based on VC dimension (Vapnik and Chervonenkis, 1971), as follows. Suppose that we train linear classifiers on the training sets for our tests. To simplify our analysis, assume that all trained classifiers are consistent, that is, they have zero error on their training data. This consistency assumption produces stronger VC bounds, allowing us to use the bound formula (Cristianini and Shawe-Taylor, 2000, p. 56):

$$\frac{2}{t}\left(d\log\frac{2et}{d}+\log\frac{2}{\delta}\right),$$

where t is the number of training examples, d is the VC dimension, which is one more than the number of input dimensions for linear classifiers, and δ is the allowed probability of bound failure. Let $\delta = 0.3$. For the iris problem, $t = 40$ and $d = 5$, producing bound 1.5. For the diabetes problem, $t = 200$, $d = 9$, and the bound is 0.65. For the problems with randomly generated data, $t = 100$, $d = 4$, and the bound is 0.62. Compare these bounds to those for $\delta = 0.3$ in tables 1 to 4.

8.2 Joint Frequencies of Errors and Error Bounds

Tables 5, 6, and 7 show bound versus error for 1000 trials using the same XOR-based random data generator used in the previous test and the same classification method, 1-nearest neighbor. These results are for a sampled filter with 1000 sample partitions. Errors are listed down the left column, and error bounds are listed across the top. The value in each cell is the number of trials that have the error indicated by the row and the bound indicated by the column. Cells with value zero are left blank.

	Bound										
Error	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.0			11	95	52	4					
0.1			22	166	110	4					
0.2			22	168	102	7					
0.3			12	93	61	1					
0.4			4	29	18						
0.5			1	12	4						
0.6											
0.7				1	1						
0.8											
0.9											
1.0											

Table 6: Bounds vs. Errors for $\delta = 0.2$

	Bound										
Error	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.0			43	87	14						
0.1		5	99	200	22	2					
0.2		1	96	164	25						
0.3		2	56	91	10	1					
0.4		2	26	33	4						
0.5			5	7	2						
0.6			2								
0.7				1							
0.8											
0.9											
1.0											

Table 7: Bounds vs. Errors for $\delta = 0.3$

		Bound Margin		
δ	$< 0.0(failure)$	0.0 (exact)	+0.1	+0.2
0.1	3.1%	5.8%	13.8%	26.8%
0.2	6.4%	13.3%	25.1%	28.0%
0.3	11.0%	19.6%	27.3%	26.9%

Table 8: Frequencies of Bound Margins

The diagonal from the top left to the bottom right contains cells for which the error and the bound are the same. Cells below this diagonal indicate bound failure—the bound is less than the actual error. Cells above indicate bounds above actual errors. Note how the cloud of values moves toward the diagonal as δ progresses from 0.1 to 0.3.

w	1000	LOO	Double
3	0.20 ± 0.16	0.23 ± 0.22	$\mathbf{0.073 \pm 0.21}$
4	0.21 ± 0.10	0.27 ± 0.18	$\mathbf{0.13 \pm 0.25}$
5	$\mathbf{0.22 \pm 0.084}$	0.31 ± 0.19	0.23 ± 0.25
6	$\mathbf{0.24 \pm 0.087}$	0.35 ± 0.18	0.30 ± 0.21
7	$\mathbf{0.26 \pm 0.078}$	0.39 ± 0.18	0.34 ± 0.21
8	$\mathbf{0.29 \pm 0.074}$	0.44 ± 0.17	0.38 ± 0.21
9	$\mathbf{0.31 \pm 0.076}$	0.47 ± 0.18	0.40 ± 0.22
10	$\mathbf{0.33 \pm 0.074}$	0.52 ± 0.17	0.45 ± 0.22

Table 9: Bounds vs. Number of Working Examples for $\delta = 0.1$

Table 8 summarizes the frequencies of bound margins, that is, of differences between bound and error. A bound margin less than zero means the bounding method fails, supplying an invalid error bound. From the failure column in Table 8, observe that the actual frequency of bound failure is significantly less than δ, the allowed rate of failure. The subsequent columns indicate differences between bound and actual error: exact match, over by 0.1, and over by 0.2.

Suppose we define bound failure as a negative margin and bound success as a valid bound within 0.2 of actual error. Then for $\delta = 0.1$, we have about 3% failure and about 46% success. For $\delta = 0.2$, we have about 6% failure and about 66% success. For $\delta = 0.3$, we have 11% failure and about 74% success.

8.3 Working Set Sizes and Bounds

The next results are from tests to explore how working set sizes affect error bounds. In general, since the bounding methods rely on training examples to constrain the set of likely assignments and hence to constrain the error bound, having more training examples and fewer working examples produces stronger bounds. These tests illustrate this effect and compare it over some bounding methods.

These tests use the iris classification data. The bounding methods are a sampled filter with 1000 sample partitions (1000), virtual partitioning with a leave-one-out filter (LOO), and virtual partitioning with a filter that scores one for each example misclassified by both of its two nearest neighbors (Double). For $\delta = 0.1$, $\delta = 0.2$, and $\delta = 0.3$, the number of training examples is held constant at $t = 40$ while the number of working examples w varies from three to 10.

Table 9 shows results for $\delta = 0.1$, with the lowest mean score for a bounding method in each row in bold. Note that the double-error method is better than the other two methods for small working sets, but not for larger ones. Recall that the double-error method is most effective when each working example has as nearest neighbors training examples that agree with one another. As working set sizes increase, it becomes more likely that working examples become nearest neighbors of other working examples, weakening the double-error filter.

Table 10 shows results for $\delta = 0.2$, with the lowest mean score for a bounding method in each row in bold. A variety of methods perform well for small working sets, but the sampled filter method works best for larger working sets.

Table 11 shows results for $\delta = 0.3$, with the lowest mean score for a bounding method in each row in bold. For these tests, the sampled filter method performed best for all working set sizes. Compare values in this table to values in the previous two tables. Notice that the performance of

w	1000	LOO	Double
3	$\mathbf{0.027 \pm 0.090}$	0.040 ± 0.15	0.047 ± 0.17
4	$\mathbf{0.063 \pm 0.11}$	0.11 ± 0.18	0.074 ± 0.20
5	0.11 ± 0.10	0.16 ± 0.19	$\mathbf{0.10 \pm 0.21}$
6	$\mathbf{0.14 \pm 0.074}$	0.21 ± 0.18	0.16 ± 0.24
7	$\mathbf{0.15 \pm 0.067}$	0.26 ± 0.19	0.22 ± 0.26
8	$\mathbf{0.17 \pm 0.074}$	0.31 ± 0.19	0.24 ± 0.25
9	$\mathbf{0.19 \pm 0.071}$	0.35 ± 0.19	0.29 ± 0.25
10	$\mathbf{0.22 \pm 0.068}$	0.40 ± 0.18	0.33 ± 0.25

Table 10: Bounds vs. Number of Working Examples for $\delta = 0.2$

w	1000	LOO	Double
3	$\mathbf{0.0017 \pm 0.024}$	0.018 ± 0.11	0.040 ± 0.16
4	$\mathbf{0.0083 \pm 0.045}$	0.043 ± 0.15	0.057 ± 0.17
5	$\mathbf{0.027 \pm 0.069}$	0.068 ± 0.16	0.077 ± 0.17
6	$\mathbf{0.047 \pm 0.075}$	0.11 ± 0.18	0.10 ± 0.19
7	$\mathbf{0.070 \pm 0.075}$	0.16 ± 0.19	0.14 ± 0.22
8	$\mathbf{0.088 \pm 0.065}$	0.20 ± 0.19	0.18 ± 0.23
9	$\mathbf{0.11 \pm 0.061}$	0.24 ± 0.20	0.24 ± 0.24
10	$\mathbf{0.13 \pm 0.064}$	0.29 ± 0.19	0.27 ± 0.23

Table 11: Bounds vs. Number of Working Examples for $\delta = 0.3$

the sampled filter method improves noticeably as δ increases. In contrast, the performance of the double-error filter does not change much with δ, especially for small working sets.

9. Discussion

This paper introduces a new method to compute an error bound for applying a classifier based on training examples to a set of working examples with known inputs. The method uses information from the training examples and inputs of working examples to develop a set of likely assignments to outputs of the working examples. A likely assignment with maximum error determines the bound. The method is very effective for small data sets.

In the bounds, filters translate training examples and inputs of working examples into constraints on the assignments to outputs of the working examples. Several filters are introduced in this paper. The complete filter is simple and direct; it evaluates each assignment by comparing the error caused by the training/working partition at hand to the errors caused by all other training/working partitions, rejecting the assignment as unlikely if the error for the partition at hand is especially large. The complete filter is effective because it optimizes directly over the error on the partition at hand, which is the basis for the bound. The sampled filter is an easier-to-compute variant that uses only a subset of training/working partitions rather than computing error for all of them. With a sufficient number of samples, the sampled filter performs about as well as the complete filter.

Filters based on virtual partitions reduce computation by not computing errors over partitions for each assignment. The tradeoff is that filters for virtual partitions rely on indirect measures of

whether assignments are likely. This paper introduces a filter based on leave-one-out error and presents an algorithm based on that filter to efficiently compute an error bound for 1-nearest neighbor classification. This paper also introduces a filter based on whether the two nearest neighbors both misclassify an example. Tests show that this filter can be more effective than the complete filter for problems with little classification error and small numbers of working examples.

Directions for future research include:

1. Develop and analyze new filters for use with virtual partitions, to compete with the leave-one-out and double-error filters presented here.

2. Analyze how training set size, working set size, and properties of the data influence the bounds, with the goal of developing new filters that target different types of problems.

3. Develop efficient algorithms for nearest neighbor classifiers using virtual partitioning with filters beyond the leave-one-out filter.

4. Develop efficient algorithms to compute error bounds using virtual partitioning for classifiers other than nearest neighbor classifiers, such as condensed and edited nearest neighbors classifiers (Devroye et al., 1996) and support vector machines (Vapnik, 1998).

5. Explore how to use efficient bounds for nearest neighbor classifiers to indirectly produce bounds for other types of classifiers. For example, for support vector machines, first bound nearest neighbor error. Then use the bound to constrain assignments, and bound the support vector machine error by the maximum error over the constrained set of assignments.

6. Consider using alternatives to the set of sister partitions in the complete and sampled filters. For example, with 100 training examples and 10 working examples, partition the training examples into blocks of 10 examples each. Treating the working examples as another block, there are 11 blocks. Each partition that has one block as the working examples and the remaining blocks as the training examples is equally likely. So these partitions can be used in place of the sister partitions. As another example, with 10 training examples and 10 working examples, pair off each working example with a different training example. Each partition that has one of each pair in the working examples and the other in the training examples is equally likely. So these partitions can be used in place of the sister partitions.

Acknowledgments

We thank Joel Franklin for encouraging this research through his example and his teaching. We also thank two anonymous referees for very helpful suggestions on notation and presentation.

Appendix A. Java Code to Compute Error Bounds for 1-Nearest Neighbor Classifiers

The excerpt of Java code below uses the approach described in Section 7.4 to efficiently compute all e_{ijky} in methods computePossibilities and computeSlice. (Array a in the code plays the role of array *prev* in Section 7.4, and array b plays the role of array *next*.) The same technique is used to compute all *v*ij in method combineRoots. Each root is treated as a child of a virtual "super-root". The method named bound is included to give an overview of the computation.

```
public class VirtualPartitionBounder
{
  Problem p; // Handles examples, labels, neighbors, memberships in T and W.
  int[][][][] e; // e[i][j][k][y]'s.
  int[][] a; // a[i][j]'s to accumulate a slice over children.
  int[][] b; // b[i][j]'s to accumulate a slice over children.
  int[] order; // Ordering of examples with children in F before parents
  int[][] children; // children[k][] is a list of children of k in F
  int[] roots; // Examples that are roots in F

  /**
   * Constructor.
   **/
  public VirtualPartitionBounder(Problem p)
  {
    this.p = p;
    this.e = new int[p.sizeT()+1][p.sizeW()+1][p.sizeTuW()][2];
    this.a = new int[p.sizeT()+1][p.sizeW()+1];
    this.b = new int[p.sizeT()+1][p.sizeW()+1];
  }

  /**
   * Returns a bound on the number of errors on W by T, with probability
   * of bound failure at most delta.
   **/
  public int bound(double delta)
  {
    int[][] v = computePossibilities();

    double[][] u = computeTails();

    int m = 0;

    for (int i=0; i<u.length; i++)
      for (int j=0; j<u[i].length; j++)
      {
        if (u[i][j]>=delta) m = max(m, v[i][j]);
      }

    return m;
  }

  /**
   * Compute e-values by the slice, then combine roots to compute v-values.
   **/
```

```
public int[][] computePossibilities()
{
  computeOrderChildrenAndRoots();
    // Compute members order, children, and roots.

  clearE();  // Set all e[i][j][k][y] to -1.

  // Loop through slices.
  for (int i=0; i<order.length; i++)
  {
    int k = order[i];

    for (int yk=0; yk<2; yk++) computeSlice(k, yk, children[k]);
  }

  combineRoots(); // Use slices for roots to compute v-values.

  return a;
}

/**
 * Computes a slice e[][][k][yk] by accumulating terms over children.
 **/
private void computeSlice(int k, int yk, int[] kids)
{
  clearA(); // Set all a[i][j] to -1.
  clearB(); // Set all b[i][j] to -1.

  // Set intial values in a by treating k as a leaf in F.
  if (p.inT(k)) // Example k is in T.
  {
    if (yk!=p.getLabel(k)) return; // Impossible label on k.
    else a[0][0] = 0; // Correct label on k.
  }
  else // Example k is in W.
  {
    p.setLabel(k, yk); // Assign label yk to k.

    if (p.isMisclassifiedByT(k)) a[0][0] = 1;
    else a[0][0] = 0;
  }

  // Accumulate terms over children.
  for (int look=0; look<kids.length; look++)
  {
```

```
    int n = kids[look]; // Get child.

    for (int yn=0; yn<2; yn++) // Label child.
    {
      if (p.inT(n) && yn!=p.getLabel(n)) continue;
      if (p.inW(n)) p.setLabel(n, yn);

      int di = p.cT(n,k) + p.dT(n,k);
      int dj = p.cW(n,k) + p.dW(n,k);

      for (int i=0; i<e.length; i++)
        for (int j=0; j<e[i].length; j++)
          if (e[i][j][n][yn]!=-1)
          {
            for (int ii=0; ii<a.length; ii++)
              for (int jj=0; jj<a[ii].length; jj++)
                if (a[ii][jj]!=-1)
                {
                  b[ii+i+di][jj+j+dj] = max(b[ii+i+di][jj+j+dj],
                      a[ii][jj] + e[i][j][n][yn]);
                }
          }
    }

    copyBToA(); // Copy b to a before handling the next child.
    clearB(); // Set all b[i][j] to -1.
  }

  insertAIntoE(k, yk);
}

/**
 * Combine roots to compute v-values. Treat each root as a child
 * of a virtual super-root.
 **/
private void combineRoots()
{
  clearA(); // Set all a[i][j] to -1.
  clearB(); // Set all b[i][j] to -1.

  a[0][0] = 0; // The super-root introduces no errors.

  // Accumulate terms over roots.
  for (int look=0; look<roots.length; look++)
  {
```

```
      int n = roots[look]; // Get root.

      for (int i=0; i<e.length; i++)
        for (int j=0; j<e[i].length; j++)
        {
          int v = max(e[i][j][n][0], e[i][j][n][1]);

          if (v!=-1)
            for (int ii=0; ii<a.length; ii++)
              for (int jj=0; jj<a[ii].length; jj++)
                if (a[ii][jj]!=-1)
                  b[ii+i][jj+j] = max(b[ii+i][jj+j],
                      a[ii][jj] + v);
        }

      copyBToA(); // Copy b to a before handling the next root.
    }
  }
}
```

Now consider the storage and time requirements for this technique. The storage is dominated by the array *e*, which has size O(*tw(t+w)*). The time is dominated by the nested loops in methods computePossibilities, computeSlice, and combineRoots. Examine the nested loops in computeSlice that run for each child and possible *y*-value for the child. The nesting is four deep, with two loops of size *t*+1 and two of size *w*+1. So, for each child-parent relationship, the time is $O(t^2w^2)$. Note that the loops for each root in combineRoots are similar to those for each child in computeSlice. Since each of the t+w examples is a child of one example in *F* or a root in *F*, and there are at most two possible *y*-values for each example, the total time is $O(t^2w^2(t+w))$.

To reduce storage, discard the slice for each example after using it to compute the slice for the parent of the example in *F*. To reduce storage and time in most cases, store a list of nonnegative values for each slice rather than storing the slice in array form with all the -1's included. Then iterate over those lists rather than all pairs (*i*,*j*) and (*ii*,*jj*) in methods computeSlice and combineRoots.

References

J.-Y. Audibert. *PAC-Bayesian Statistical Learning Theory*. PhD thesis, Laboratoire de Probabilities et Modeles Aleatoires, Universites Paris 6 and Paris 7, 2004. URL `http://cermis.enpc.fr/~audibert/ThesePack.zip`.

E. Bax. Partition-based and sharp uniform error bounds. *IEEE Transactions on Neural Networks*, 10(6):1315–1320, 1999.

A. Blum and J. Langford. Pac-mdl bounds. In *Proceedings of the 16th Annual Conference on Computational Learning Theory (COLT)*, pages 344–357, 2003.

O. Catoni. A pac-bayesian approach to adaptive classification, preprint n.840. Technical report, Laboratoire de Probabilities et Modeles Aleatoires, Universites Paris 6 and Paris 7, 2003. URL http://www.proba.jussieu.fr/users/catoni/homepage/dea2005.pdf.

O. Catoni. Improved vapnik-chervonenkis bounds, preprint n.942. Technical report, Laboratoire de Probabilities et Modeles Aleatoires, Universites Paris 6 and Paris 7, 2004. URL http://www.proba.jussieu.fr/mathdoc/preprints/index.html#2004.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

P. Derbeko, R. El-Yaniv, and R. Meir. Error bounds for transductive learning via compression and clustering. In *Advances in Neural Information Processing Systems (NIPS) 16*, pages 1085–1092, Cambridge, MA, 2003. MIT Press.

L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, 1996.

R. El-Yaniv and L. Gerzon. Effective transductive learning via objective model selection. *Pattern Recognition Letters*, 26(13):2104–2115, 2005.

T. Joachims. *Learning to Classify Text using Support Vector Machines*. Kluwer Academic Publishers, 2002.

N. Littlestone and M. Warmuth. Relating data compression and learnability, 1986. Unpublished manuscript, University of California Santa Cruz.

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

Journal of Machine Learning Research 9 (2008) 893-914 Submitted 10/07; Revised 2/08; Published 5/08

Graphical Methods for Efficient Likelihood Inference in Gaussian Covariance Models

Mathias Drton DRTON@UCHICAGO.EDU
Department of Statistics
University of Chicago
5734 S. University Ave
Chicago, IL 60637, USA

Thomas S. Richardson TSR@STAT.WASHINGTON.EDU
Department of Statistics
University of Washington, Box 354322
Seattle, WA 98195-4322, USA

Editor: Max Chickering

Abstract

In graphical modelling, a bi-directed graph encodes marginal independences among random variables that are identified with the vertices of the graph. We show how to transform a bi-directed graph into a maximal ancestral graph that (i) represents the same independence structure as the original bi-directed graph, and (ii) minimizes the number of arrowheads among all ancestral graphs satisfying (i). Here the number of arrowheads of an ancestral graph is the number of directed edges plus twice the number of bi-directed edges. In Gaussian models, this construction can be used for more efficient iterative maximization of the likelihood function and to determine when maximum likelihood estimates are equal to empirical counterparts.

Keywords: ancestral graph, covariance graph, graphical model, marginal independence, maximum likelihood estimation, multivariate normal distribution

1. Introduction

In graphical modelling, bi-directed graphs encode marginal independences among random variables that are identified with the vertices of the graph (Pearl and Wermuth, 1994; Kauermann, 1996; Richardson, 2003). In particular, if two vertices are not joined by an edge, then the two associated random variables are assumed to be marginally independent. For example, the graph G in Figure 1, whose vertices are to be identified with a random vector (X_1, X_2, X_3, X_4), represents the pairwise marginal independences $X_1 \perp\!\!\!\perp X_3$, $X_1 \perp\!\!\!\perp X_4$, and $X_2 \perp\!\!\!\perp X_4$. While other authors (Cox and Wermuth, 1993, 1996; Edwards, 2000) have used dashed edges to represent marginal independences, the bi-directed graphs we employ here make explicit the connection to path diagrams (Wright, 1934; Koster, 1999).

Gaussian graphical models for marginal independence, also known as covariance graph models, impose zero patterns in the covariance matrix, which are linear hypotheses on the covariance matrix (Anderson, 1973). The graph in Figure 1, for example, imposes $\sigma_{13} = \sigma_{14} = \sigma_{24} = 0$. An estimation procedure designed for covariance graph models is described in Drton and Richardson (2003); see

Figure 1: A bi-directed graph G with (unique) minimally oriented graph $G^{\min}$.

also Chaudhuri et al. (2007). Other recent work involving these models includes Mao et al. (2004) and Wermuth et al. (2006).

In this paper we employ the connection between bi-directed graphs and the more general ancestral graphs with undirected, directed, and bi-directed edges (Section 2). For the statistical motivation of ancestral graphs see Richardson and Spirtes (2002); for causal interpretation see Richardson and Spirtes (2003). We show how to construct a maximal ancestral graph $G^{\min}$, which we call a minimally oriented graph, that is Markov equivalent to a given bi-directed graph G and such that the number of arrowheads is minimal (Sections 3–4). Two ancestral graphs are Markov equivalent if the independence models associated with the two graphs coincide; see for example Roverato (2005) for some recent results on Markov equivalence of different types of graphs. The number of arrowheads is the number of directed edges plus twice the number of bi-directed edges. Minimally oriented graphs provide useful nonparametric information about Markov equivalence of bi-directed, undirected and directed acyclic graphs. For example, the graph G in Figure 1 is not Markov equivalent to an undirected graph because $G^{\min}$ is not an undirected graph, and G is not Markov equivalent to a DAG because $G^{\min}$ contains a bi-directed edge. The graph G in Figure 1 has a unique minimally oriented graph but in general, minimally oriented graphs are not unique. Our construction procedure (Algorithm 14) involves a choice of a total order among the vertices. Varying the order one may obtain all minimally oriented graphs (Theorem 17).

For covariance graph models, minimally oriented graphs allow one to determine when the maximum likelihood estimate of a variance or covariance is available explicitly as its empirical counterpart (Section 5). For example, since no arrowheads appear at the vertices 1 and 4 in the graph $G^{\min}$ in Figure 1, the maximum likelihood estimates of σ_{11} and σ_{44} must be equal to the empirical variances of X_1 and X_4, respectively. The likelihood function for covariance graph models may be multi-modal, though simulations suggest this only occurs at small sample sizes, or under misspecification (Drton and Richardson, 2004a). However, when a minimally oriented graph reveals that a parameter estimate is equal to an empirical quantity (such as σ_{11} and σ_{44} in the above example) then even if the likelihood function is multi-modal this parameter will take the same value at every mode. Perhaps most importantly, minimally oriented graphs allow for computationally more efficient maximum likelihood fitting; see Remark 24 and the example in Section 5.3.

2. Ancestral Graphs and Their Global Markov Property

This paper deals with *simple mixed graphs*, which feature undirected ($v - w$), directed ($v \to w$) and bi-directed edges ($v \leftrightarrow w$) under the constraint that there is at most one edge between two vertices. In this section we give a formal definition of these graphs and discuss their Markov interpretation.

2.1 Simple Mixed Graphs

Let $\mathcal{E} = \{\emptyset, -, \leftarrow, \rightarrow, \leftrightarrow\}$ be the set of possible edges between an ordered pair of vertices; $\emptyset$ denoting that there is no edge. A *simple mixed graph* $G = (V, E)$ is a pair of a finite *vertex set* V and an *edge map* $E : V \times V \to \mathcal{E}$. The edge map E has to satisfy that for all $v, w \in V$,

(i) $E(v,v) = \emptyset$, that is, there is no edge between a vertex and itself,

(ii) $E(v,w) = E(w,v)$ if $E(v,w) \in \{-, \leftrightarrow\}$,

(iii) $E(v,w) = \rightarrow \iff E(w,v) = \leftarrow$.

In the sequel, we write $v - w \in G$, $v \to w \in G$, $v \leftarrow w \in G$ or $v \leftrightarrow w \in G$ if $E(v,w)$ equals $-$, $\rightarrow$, $\leftarrow$ or $\leftrightarrow$, respectively. If $E(v,w) \neq \emptyset$, then v and w are *adjacent*. If there is an edge $v \leftarrow w \in G$ or $v \leftrightarrow w \in G$ then there is an *arrowhead at* v on this edge. If there is an edge $v \to w \in G$ or $v - w \in G$ then there is a *tail at* v on this edge. A vertex w is in the *boundary* of v, denoted by $\mathrm{bd}(v)$, if v and w are adjacent. The boundary of vertex set $A \subseteq V$ is the set $\mathrm{bd}(A) = [\cup_{v \in A} \mathrm{bd}(v)] \setminus A$. We write $\mathrm{Bd}(v) = \mathrm{bd}(v) \cup \{v\}$ and $\mathrm{Bd}(A) = \mathrm{bd}(A) \cup A$. An induced subgraph of G over a vertex set A is the mixed graph $G_A = (A, E_A)$ where E_A is the restriction of the edge map E on $A \times A$. The *skeleton* of a simple mixed graph is obtained by making all edges undirected.

In a simple mixed graph a sequence of adjacent vertices $(v_1, \ldots, v_k)$ uniquely determines the sequence of edges joining consecutive vertices v_i and v_{i+1}, $1 \leq i \leq k-1$. Hence, we can define a *path* π between two vertices v and w as a sequence of distinct vertices $\pi = (v, v_1, \ldots, v_k, w)$ such that each vertex in the sequence is adjacent to its predecessor and its successor. A path $v \to \cdots \to w$ with all edges of the form $\rightarrow$ and pointing toward w is a directed path from v to w. If there is such a directed path from v to $w \neq v$, or if $v = w$, then v is an *ancestor* of w. We denote the set of all ancestors of a vertex v by $\mathrm{An}(v)$ and for a vertex set $A \subseteq V$ we define $\mathrm{An}(A) = \cup_{v \in A} \mathrm{An}(v)$. Finally, a directed path from v to w together with an edge $w \to v \in G$ is called a *directed cycle*.

Important subclasses of simple mixed graphs are illustrated in Figure 2. *Bi-directed*, *undirected* and *directed graphs* contain only one type of edge. *Directed acyclic graphs* (DAGs) are directed graphs without directed cycles. These three types of graph are special cases of *ancestral graphs* (Richardson and Spirtes, 2002).

Definition 1 *A simple mixed graph G is an* ancestral graph *if it holds that*

(i) G does not contain any directed cycles;

(ii) if $v - w \in G$, then there does not exist u such that $u \to v \in G$ or $u \leftrightarrow v \in G$;

(iii) if $v \leftrightarrow w \in G$, then v is not an ancestor of w.

2.2 Global Markov Property for Ancestral Graphs

Ancestral graphs can be given an independence interpretation, known as the global Markov property, by a graphical separation criterion called *m*-separation (Richardson and Spirtes, 2002, §3.4). An extension of Pearl's (1988) *d*-separation for DAGs, *m*-separation uses the notion of *colliders*. A non-endpoint vertex v_i on a path is a *collider on the path* if the edges preceding and succeeding v_i on the path both have an arrowhead at v_i, that is, $v_{i-1} \to v_i \leftarrow v_{i+1}$, $v_{i-1} \to v_i \leftrightarrow v_{i+1}$, $v_{i-1} \leftrightarrow v_i \leftarrow v_{i+1}$ or $v_{i-1} \leftrightarrow v_i \leftrightarrow v_{i+1}$ is part of the path. A non-endpoint vertex that is not a collider is a *non-collider on the path*.

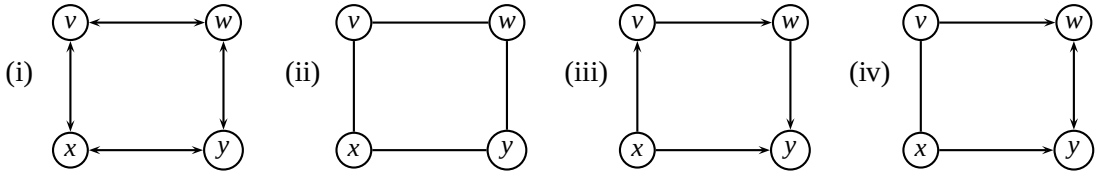


Figure 2: Simple mixed graphs. (i) A bi-directed graph, (ii) an undirected graph, (iii) a DAG, (iv) an ancestral graph.

Definition 2 *A path π between vertices v and w in a simple mixed graph G is m-*connecting *given a possibly empty set $C \subseteq V \setminus \{v,w\}$ if (i) every non-collider on π is not in C, and (ii) every collider on π is in* $\mathrm{An}(C)$. *If no path m-connects v and w given C, then v and w are m-*separated *given C. Two non-empty and disjoint sets A and B are m-separated given $C \subseteq V \setminus (A \cup B)$, if any two vertices $v \in A$ and $w \in B$ are m-separated given C.*

Let $G = (V,E)$ be an ancestral graph whose vertices index a random vector $(X_v \mid v \in V)$. For $A \subseteq V$, let X_A be the subvector $(X_v \mid v \in A)$. The *global Markov property* for G states that X_A is conditionally independent of X_B given X_C whenever A, B and C are pairwise disjoint subsets such that A and B are m-separated given C in G. Subsequently, we denote such conditional independence using the shorthand $A \perp\!\!\!\perp B \mid C$ that avoids making the probabilistic context explicit. The global Markov property, when applied to each of the graphs in Figure 2 in turn, implies (among other independences) that:

(i) $v \perp\!\!\!\perp y$ and $w \perp\!\!\!\perp x$;

(ii) $v \perp\!\!\!\perp y \mid \{w,x\}$ and $w \perp\!\!\!\perp x \mid \{v,y\}$;

(iii) $v \perp\!\!\!\perp y \mid \{w,x\}$ and $w \perp\!\!\!\perp x \mid v$;

(iv) $v \perp\!\!\!\perp y \mid x$ and $w \perp\!\!\!\perp x \mid v$.

If G is a bi-directed graph, then the global Markov property states the marginal independence $v \perp\!\!\!\perp w$ if v and w are not adjacent. In a multivariate normal distribution such pairwise marginal independences hold iff all independences stated by the global Markov property for G hold (Kauermann, 1996). Without any distributional assumption, Richardson (2003, §4) shows that the independences stated by the global Markov property of a bi-directed graph hold iff certain (not only pairwise) marginal independences hold; see also Matúš (1994). Two ancestral graphs G_1 and G_2 are *Markov equivalent* if they have the same vertex set and the global Markov property states the same independences for G_1 as for G_2.

The graphs in Figure 2 have the property that for every pair of non-adjacent vertices v and w there exists some subset C such that the global Markov property states that $v \perp\!\!\!\perp w \mid C$. Ancestral graphs with this property are called *maximal*. If an ancestral graph G is not maximal, then there exists a unique Markov equivalent maximal ancestral graph $\overline{G}$ that contains all the edges present in G. Moreover, any edge in $\overline{G}$ that is not present in G is bi-directed (Richardson and Spirtes, 2002, §3.7).

The following facts are easily established; see also Richardson and Spirtes (2002).

Lemma 3 *(i) Markov equivalent maximal ancestral graphs have the same skeleton.*

(ii) If $\bar{G}$ is an ancestral graph that is Markov equivalent to a maximal ancestral graph G and has the same skeleton as G, then $\bar{G}$ is also a maximal ancestral graph.

(iii) Bi-directed, undirected and directed acyclic graphs are maximal ancestral graphs.

2.3 Boundary Containment

In the subsequent Sections 3 and 4 we will construct maximal ancestral graphs that are Markov equivalent to a given bi-directed graph. Via Theorem 5 below, the following property plays a crucial role in these constructions.

Definition 4 *A simple mixed graph G has the* boundary containment property *if for all distinct vertices $v, w \in V$ the presence of an edge $v - w$ implies that* $\mathrm{Bd}(v) = \mathrm{Bd}(w)$ *and the presence of an edge $v \to w$ in G implies that* $\mathrm{Bd}(v) \subseteq \mathrm{Bd}(w)$.

In the Appendix we present lemmas on the structure of m-connecting paths in graphs with the boundary containment property. These lemmas yield the following key result.

Theorem 5 *If $\bar{G}$ is an ancestral graph that has the same skeleton as a bi-directed graph G, then G and $\bar{G}$ are Markov equivalent iff $\bar{G}$ has the boundary containment property.*

Proof Two vertices are adjacent in G iff they are adjacent in $\bar{G}$. Therefore, G and $\bar{G}$ are Markov equivalent iff it holds that two non-adjacent vertices v and w are m-connected given $C \subseteq V$ in G iff they are m-connected given C in $\bar{G}$.

($\Longrightarrow$:) Suppose $\bar{G}$ does not have the boundary containment property, that is, there exists an edge $v - w \in \bar{G}$ or an edge $v \to w \in \bar{G}$ such that $\mathrm{Bd}(v) \not\subseteq \mathrm{Bd}(w)$. Choose $u \in \mathrm{Bd}(v) \setminus \mathrm{Bd}(w)$. Since u and w are not adjacent, they are m-separated given $C = \emptyset$ in G. In $\bar{G}$, however, the path (u, v, w) m-connects u and w given $C = \emptyset$. Hence, G and $\bar{G}$ are not Markov equivalent.

($\Longleftarrow$:) First, let v and w be non-adjacent vertices that are m-connected given $C \subseteq V$ in $\bar{G}$. By Lemma 29, there is a path $\bar{\pi} = (v, v_1, \ldots, v_k, w)$ that m-connects v and w given C in $\bar{G}$ and is such that $v_1, \ldots, v_k$ are colliders with $\{v_1 \ldots, v_k\} \subseteq C$. Since G is a bi-directed graph, the corresponding path $\pi = (v, v_1, \ldots, v_k, w)$ in G also m-connects v and w given C.

Conversely, let v and w be non-adjacent vertices that are m-connected given $C \subseteq V$ in G. Let $\pi = (v_0, v_1, \ldots, v_k, v_{k+1})$ m-connect $v = v_0$ and $w = v_{k+1}$ given C in G such that no shorter path m-connects v and w given C. Then $v_1, \ldots, v_k$ are colliders, $\{v_1 \ldots, v_k\} \subseteq C$, and v_{i-1} and v_{i+1}, $i = 1, \ldots, k$, are not adjacent in G. (This is a special case of Lemmas 27 and 29 because a bi-directed graph trivially satisfies the boundary containment property.) It follows that, for all $i = 1, \ldots, k-1$, $v_{i-1} \in \mathrm{Bd}(v_i)$ but $v_{i-1} \notin \mathrm{Bd}(v_{i+1})$, and similarly $v_{i+2} \notin \mathrm{Bd}(v_i)$ but $v_{i+2} \in \mathrm{Bd}(v_{i+1})$. This implies that $\mathrm{Bd}(v_i) \not\subseteq \mathrm{Bd}(v_{i+1})$ and $\mathrm{Bd}(v_i) \not\supseteq \mathrm{Bd}(v_{i+1})$ for all $i = 1, \ldots, k-1$. Since $\bar{G}$ has the boundary containment property, it must hold that $v_i \leftrightarrow v_{i+1} \in \bar{G}$ for all $i = 1, \ldots, k-1$. Therefore, $v_2, \ldots, v_{k-1}$ are colliders on the path $\bar{\pi} = (v, v_1, \ldots, v_k, w)$ in $\bar{G}$. Similarly, it follows that $v_2 \in \mathrm{Bd}(v_1) \setminus \mathrm{Bd}(v)$, which entails $\mathrm{Bd}(v_1) \not\subseteq \mathrm{Bd}(v)$. Thus, v_1 is a collider on $\bar{\pi}$. Analogously, we can show that v_k is a collider on $\bar{\pi}$, which yields that $\bar{\pi}$ is a path in $\bar{G}$ that m-connects v and w given C. ■

3. Simplicial Graphs

In this section we show how simplicial vertex sets of a bi-directed graph can be used to construct a Markov equivalent maximal ancestral graph by removing arrowheads from certain bi-directed edges. Simplicial sets are also important in other contexts such as collapsibility (Madigan and Mosurski, 1990; Kauermann, 1996; Lauritzen, 1996, §2.1.3, p.121 and 219) and triangulation of graphs (Jensen, 2001, §5.3).

Definition 6 *A vertex $v \in V$ is* simplicial, *if* $\mathrm{Bd}(v)$ *is complete, that is, every pair of vertices in* $\mathrm{Bd}(v)$ *are adjacent. Similarly, a set $A \subseteq V$ is* simplicial, *if* $\mathrm{Bd}(A)$ *is complete.*

Simplicial vertices can be characterized in terms of boundary containment as follows.

Proposition 7 *A vertex $v \in V$ is simplicial iff* $\mathrm{Bd}(v) \subseteq \mathrm{Bd}(w)$ *for all* $w \in \mathrm{Bd}(v)$.

If an edge between v and w has an arrowhead at v, then we say that we *drop the arrowhead at v* when either $v \leftarrow w$ is replaced by $v - w$ or $v \leftrightarrow w$ is replaced by $v \rightarrow w$.

Definition 8 *Let G be a bi-directed graph. The* simplicial graph G^s *is the simple mixed graph obtained by dropping all the arrowheads at simplicial vertices of G.*

For the graph from Figure 1, G^s is equal to the depicted graph $G^{\min}$; additional examples are given in Figure 3. Parts (i) and (ii) of the next lemma show that simplicial graphs have the boundary containment property.

Lemma 9 *Let v and w be adjacent vertices in a simplicial graph G^s. Then*

(i) if $v - w \in G^s$, then $\mathrm{Bd}(v) = \mathrm{Bd}(w)$*;*

(ii) if $v \rightarrow w \in G^s$, then $\mathrm{Bd}(v) \subsetneq \mathrm{Bd}(w)$*;*

(iii) if $v \leftrightarrow w \in G^s$, then each of $\mathrm{Bd}(v) = \mathrm{Bd}(w)$, $\mathrm{Bd}(v) \subsetneq \mathrm{Bd}(w)$, *and* $\mathrm{Bd}(v) \not\subseteq \mathrm{Bd}(w) \not\subseteq \mathrm{Bd}(v)$ *might be the case.*

Proof (i) and (ii) follow from Proposition 7. For (iii) see, respectively, the graphs G^s_1, G^s_2 in Figure 3, and $G^s = G^{\min}$ in Figure 1. ■

Theorem 10 *The simplicial graph G^s of a bi-directed graph G is a maximal ancestral graph that is Markov equivalent to G.*

Proof By Lemma 3, Theorem 5 and Lemma 9, it suffices to show that G^s is an ancestral graph. This, however, follows from Lemma 11 below. ■

Lemma 11 *If G is an ancestral graph that has the boundary containment property, then dropping all arrowheads at simplicial vertices of G yields an ancestral graph.*

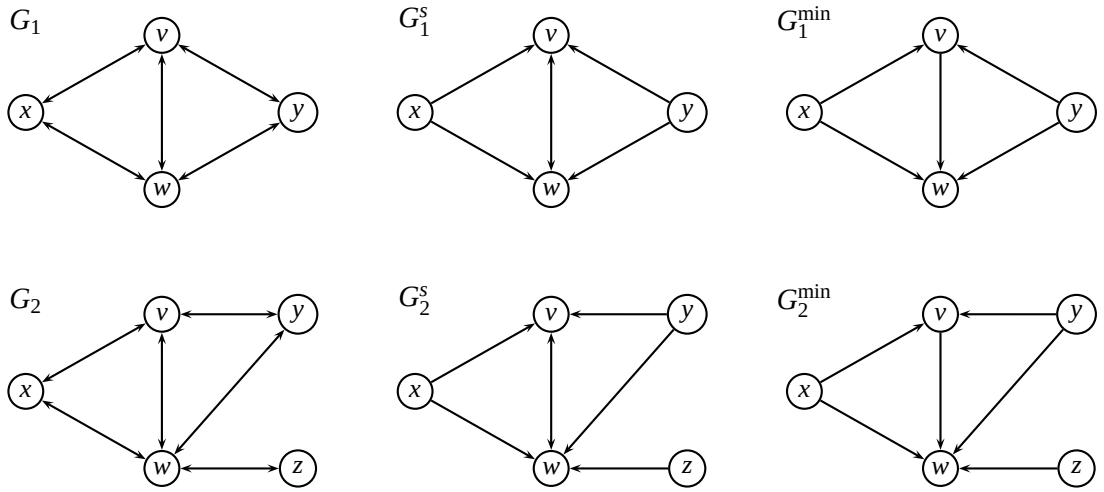


Figure 3: Bi-directed graphs with simplicial and minimally oriented graphs.

Proof Let $\bar{G}$ be the graph obtained by dropping the arrowheads at simplicial vertices. First, suppose $v \to w \in \bar{G}$ or $v \leftrightarrow w \in \bar{G}$ but that there is a path π from w to v that is a directed path in $\bar{G}$. Since there are no arrowheads at simplicial vertices in $\bar{G}$, no vertex on π including the endpoints v and w can be simplicial. This implies that π is a directed path from w to v in G. However, since $v \to w \in G$ or $v \leftrightarrow w \in G$, this is a contradiction to G being ancestral. We conclude that $\bar{G}$ satisfies conditions (i) and (iii) of Definition 1.

Next, suppose $v - w \in \bar{G}$ but that there exists another vertex u such that $u \to v \in \bar{G}$ or $u \leftrightarrow v \in \bar{G}$. It follows that v is not simplicial. Since G is ancestral, this implies that $v \to w \in G$ which in turn implies that $\mathrm{Bd}(v) \subseteq \mathrm{Bd}(w)$ because G has the boundary containment property. The set $\mathrm{Bd}(v)$ is not complete because v is not simplicial. Thus $\mathrm{Bd}(w)$ is not complete, that is, w is not a simplicial vertex. However, this is a contradiction to the fact that $v \to w \in G$ but $v - w \in \bar{G}$. Thus, $\bar{G}$ is indeed an ancestral graph. ■

Proposition 12 *A bi-directed graph G is Markov equivalent to an undirected graph iff the simplicial graph G^s induced by G is an undirected graph iff G is a disjoint union of complete (bi-directed) graphs.*

Proof If G^s is an undirected graph, then by Theorem 10, G is Markov equivalent to an undirected graph, namely G^s. Conversely, assume that there exists an undirected graph U that is Markov equivalent to G. Necessarily, G and U have the same skeleton (recall Lemma 3). By Theorem 5, U has the boundary containment property, which implies that every vertex is simplicial and thus that G^s is an undirected graph (and equal to U).

The simplicial graph G^s is an undirected graph iff the vertex set of the inducing bi-directed graph G can be partitioned into pairwise disjoint sets $A_1, \ldots, A_q$ such that (a) if $v \in A_i$, $1 \leq i \leq q$, and $w \in A_j$, $1 \leq j \leq q$, are adjacent, then $i = j$, and (b) all the induced subgraphs G_{A_i}, $i = 1, \ldots, q$ are complete graphs (Kauermann, 1996). ■

Under multivariate normality, a bi-directed graph that is Markov equivalent to an undirected graph represents a hypothesis that is linear in the covariance matrix as well as in its inverse. The general structure of such models is studied in Jensen (1988).

4. Minimally Oriented Graphs

The simplicial graph G^s sometimes may be a DAG. For example, the graph $u \leftrightarrow v \leftrightarrow w$ has the simplicial graph $u \rightarrow v \leftarrow w$. However, there exist bi-directed graphs that are Markov equivalent to a DAG and yet the simplicial graph contains bi-directed edges. For example, the graph G_1 in Figure 3 is Markov equivalent to the DAG $G_1^{\min}$ in the same Figure. Hence, some arrowheads may be dropped from bi-directed edges in a simplicial graph while preserving Markov equivalence. In this section we construct maximal ancestral graphs from which no arrowheads may be dropped without destroying Markov equivalence.

4.1 Definition and Construction

The following definition introduces the key object of this section.

Definition 13 *Let G be a bi-directed graph. A* minimally oriented graph *of G is a graph $G^{\min}$ that satisfies the following three properties:*

(i) *$G^{\min}$ is a maximal ancestral graph;*

(ii) *G and $G^{\min}$ are Markov equivalent;*

(iii) *$G^{\min}$ has the minimum number of arrowheads of all maximal ancestral graphs that are Markov equivalent to G. Here the number of arrowheads of an ancestral graph G with d directed and b bi-directed edges is defined as* $\mathrm{arr}(G) = d + 2b$.

By Lemma 3, a minimally oriented graph $G^{\min}$ has the same skeleton as the underlying bi-directed graph G. According to Theorem 5, $G^{\min}$ has the boundary containment property. Examples of minimally oriented graphs are shown in Figure 3. Given the small number of vertices of these graphs the claim that these graphs are indeed minimally oriented graphs can be verified directly. The example of graph G_1 in Figure 3 also illustrates that minimally oriented graphs are not unique. By symmetry, reversing the direction of the edge $v \rightarrow w$ in the depicted $G_1^{\min}$ yields a second minimally oriented graph for G_1.

We now turn to the problem of how to construct a minimally oriented graph. Define a relation on the vertex set V of the given bi-directed graph G by letting $v \preccurlyeq_B w$ if $v = w$ or if $\mathrm{Bd}(v) \subsetneq \mathrm{Bd}(w)$ in G. The relation $\preccurlyeq_B$ is a partial order and can thus be extended to a total order $\leq$ on V such that the strict boundary containment $\mathrm{Bd}(v) \subsetneq \mathrm{Bd}(w)$ implies that $v < w$. In general, the choice of such an extension to a total order is not unique.

Algorithm 14 *Let G be a bi-directed graph, and $\leq$ a total order on V that extends the partial order $\preccurlyeq_B$ obtained from strict boundary containment. Create a new graph $G_<^{\min}$ as follows:*

(a) *find the simplicial graph G^s of G;*

(b) set $G^{\min}_{<} = G^s$*;*

(c) replace every bi-directed edge $v \leftrightarrow w \in G^{\min}_{<}$ *with* $\mathrm{Bd}(v) \subseteq \mathrm{Bd}(w)$ *and* $v < w$ *by the directed edge* $v \rightarrow w$*.*

The notation $G^{\min}_{<}$ indicates the dependence of this graph on *both* the bi-directed graph G and the total order $\leq$. Clearly, by Theorem 5, in order for $G^{\min}_{<}$ to be a minimally oriented graph it is necessary that it satisfies the boundary containment property. The next lemma shows that this is true.

Lemma 15 *Let G be a bi-directed graph and $G^{\min}_{<}$ the graph constructed in Algorithm 14. It then holds that*

(i) if $v - w$ *is an undirected edge in* $G^{\min}_{<}$*, then* $\mathrm{Bd}(v) = \mathrm{Bd}(w)$*;*

(ii) if $v \rightarrow w$ *is a directed edge in* $G^{\min}_{<}$*, then* $\mathrm{Bd}(v) \subseteq \mathrm{Bd}(w)$*;*

(iii) $v \leftrightarrow w$ *is a bi-directed edge in* $G^{\min}_{<}$ *iff* $\mathrm{Bd}(v) \not\subseteq \mathrm{Bd}(w) \not\subseteq \mathrm{Bd}(v)$*.*

Proof (i) follows directly from Lemma 9(i) because it follows from Algorithm 14 that $G^{\min}_{<}$ and G^s contain the same undirected edges.

(ii) If the edge $v \rightarrow w$ is already present in G^s, then $\mathrm{Bd}(v) \subsetneq \mathrm{Bd}(w)$ according to Lemma 9(ii). If $v \rightarrow w$ is not already present in G^s, then $v < w$ and $\mathrm{Bd}(v) \subseteq \mathrm{Bd}(w)$.

(iii) Suppose v and w are two adjacent vertices such that $\mathrm{Bd}(v) \not\subseteq \mathrm{Bd}(w) \not\subseteq \mathrm{Bd}(v)$. Then $v \leftrightarrow w$ in G^s and this edge cannot be replaced by a directed edge in step (c) of Algorithm 14. For the converse, consider two adjacent vertices v and w such that $\mathrm{Bd}(v) \subseteq \mathrm{Bd}(w)$. (The other case is symmetric.) If $v < w$, then according to the definition of the simplicial graph and step (c) of Algorithm 14 the edge between v and w in $G^{\min}_{<}$ cannot have an arrowhead at v and thus cannot be bi-directed. If $v > w$, then $\mathrm{Bd}(v) = \mathrm{Bd}(w)$ because $\mathrm{Bd}(v) \subsetneq \mathrm{Bd}(w)$ would imply $v < w$. It follows that the edge between v and w in $G^{\min}_{<}$ cannot be bi-directed as the arrowhead at w would be removed in step (c). ■

By Lemma 15(iii), $v \leftrightarrow w \in G^{\min}_{<}$ iff there exist vertices $x \in \mathrm{bd}(v) \setminus \{w\}$ and $y \in \mathrm{bd}(w) \setminus \{v\}$ such that the induced subgraph $G_{\{x,y,v,w\}}$ equals one of the two graphs shown in Figure 4. Graphs that do not contain the four-cycle from Figure 4(ii) as an induced subgraph are known as chordal or decomposable and play an important role in graphical modelling (Lauritzen, 1996). Graphs not containing the path from Figure 4(i) as an induced subgraph are called cographs and have favorable computational properties (Brandstädt et al., 1999). For instance, cographs can be recognized in linear time (Corneil et al., 1985).

Theorem 16 *The graph $G^{\min}_{<}$ constructed in Algorithm 14 is a minimally oriented graph for the bi-directed graph G.*

Proof We verify the conditions (i) and (iii) of Definition 13. This is sufficient because $G^{\min}_{<}$ has the boundary containment property (Lemma 15) and thus condition (i) implies condition (ii) by Theorem 5.

(i) $G^{\min}_{<}$ is a maximal ancestral graph:
By Lemma 3 it suffices to show that $G^{\min}_{<}$ is an ancestral graph. Let v and w be adjacent vertices

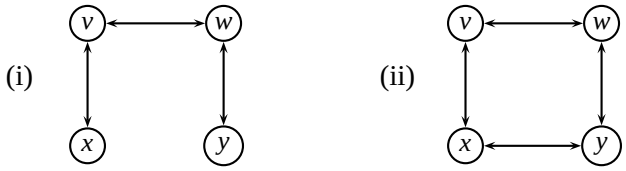


Figure 4: Induced subgraphs for which no arrowhead can be dropped from edge $v \leftrightarrow w$.

such that $v - w \in G^{\min}_{<}$. This is equivalent to $v - w \in G^s$, and it follows that there does not exist an arrowhead at v or w; compare the proof of Theorem 10. Furthermore, $G^{\min}_{<}$ does not contain any directed cycles because Algorithm 14 ensures that the presence of a directed edge $v \to w \in G^{\min}_{<}$ implies $v < w$ in the total order. Finally, assume that there exists $v \leftrightarrow w \in G^{\min}_{<}$. Then there cannot be a directed path from v to w, since by Lemma 15(ii) this would imply $\mathrm{Bd}(v) \subseteq \mathrm{Bd}(w)$, contradicting Lemma 15(iii).

(iii) $G^{\min}_{<}$ has the minimal number of arrowheads:
Let $\bar{G}$ be a maximal ancestral graph that is Markov equivalent to the (bi-directed) graph G, which requires that $\bar{G}$ and G, and thus also $G^{\min}_{<}$ have the same skeleton. Assume that $\mathrm{arr}(\bar{G}) < \mathrm{arr}(G^{\min}_{<})$. Then either (a) there exists $v \to w \in G^{\min}_{<}$ such that $v - w \in \bar{G}$ or (b) there exists $v \leftrightarrow w \in G^{\min}_{<}$ such that $v \to w \in \bar{G}$ or $v - w \in \bar{G}$.

Case (a): If $v \to w \in G^{\min}_{<}$, then w cannot be simplicial. Hence, there exist two vertices $x, y \in \mathrm{bd}(w)$ that are not adjacent in $G^{\min}_{<}$, and thus not adjacent in G; ($v = x$ is possible). The global Markov property of G states that $x \perp\!\!\!\perp y$. Since $\bar{G}$ is an ancestral graph and $v - w \in \bar{G}$, however, there may not be any arrowheads at w on the edges between x and w, and y and w in $\bar{G}$. Therefore, x and y are m-connected given $\emptyset$ in $\bar{G}$, which yields that the global Markov property of $\bar{G}$ does not imply $x \perp\!\!\!\perp y$; a contradiction.

Case (b): Suppose $v \leftrightarrow w \in G^{\min}_{<}$ but there is no arrowhead at v on the edge between v and w in $\bar{G}$. By Lemma 15(iii) there exists $x \in \mathrm{bd}(v) \setminus \mathrm{Bd}(w)$ such that x and w are not adjacent in $G^{\min}_{<}$. Thus x and w are not adjacent in G and $x \perp\!\!\!\perp w$ is stated by the global Markov property for G. In $\bar{G}$, however, v is a non-collider on the path (x, v, w) and thus this path m-connects x and w given $\emptyset$, which yields that the global Markov property of $\bar{G}$ does not imply $x \perp\!\!\!\perp w$; a contradiction. ■

The next result shows that our construction of minimally oriented graphs is complete in the sense that every minimally oriented graph can be obtained as the output of Algorithm 14 by appropriate choice of a total order on the vertex set.

Theorem 17 *If $G^{\min}$ is a minimally oriented graph for a bi-directed graph G, then there exists a total order $\leq$ on the vertex set such that $G^{\min} = G^{\min}_{<}$.*

Proof The graph $G^{\min}$ is an ancestral graph and thus contains no directed cycles. Hence, the directed edges in $G^{\min}$ yield a partial order $\preccurlyeq_D$ on the vertex set V in which $v \preccurlyeq_D w$ if $v = w$ or if there is a directed path from v to w. Define the relation $\preccurlyeq_{BD}$ by letting $v \preccurlyeq_{BD} w$ if $v \preccurlyeq_B w$ or $v \preccurlyeq_D w$. Clearly, $v \preccurlyeq_{BD} v$, that is, the relation is reflexive. We claim that the relation is in fact a partial order.

By Theorem 5, $G^{\min}$ has the boundary containment property such that $\mathrm{Bd}(v) \subseteq \mathrm{Bd}(w)$ if $v \preccurlyeq_D w$. Consequently, if $v \neq w$ then $v \preccurlyeq_D w$ implies $w \not\preccurlyeq_B v$ and $v \preccurlyeq_B w$ implies $w \not\preccurlyeq_D v$. This implies that

$\preccurlyeq_{BD}$ is anti-symmetric. In order to verify transitivity, it suffices to consider three distinct vertices satisfying $v \preccurlyeq_D w \preccurlyeq_B u$ or $v \preccurlyeq_B w \preccurlyeq_D u$. In the former case $\mathrm{Bd}(v) \subseteq \mathrm{Bd}(w) \subsetneq \mathrm{Bd}(u)$, and in the latter case $\mathrm{Bd}(v) \subsetneq \mathrm{Bd}(w) \subseteq \mathrm{Bd}(u)$. In both cases $\mathrm{Bd}(v) \subsetneq \mathrm{Bd}(u)$ such that $v \preccurlyeq_B u$, which implies the required conclusion $v \preccurlyeq_{BD} u$.

We can now choose a total order $\leq$ on V that extends the partial order $\preccurlyeq_{BD}$ and thus extends both $\preccurlyeq_B$ and $\preccurlyeq_D$. Let $G^{\min}_{<}$ be the output of Algorithm 14 when the bi-directed graph G and the chosen total order $\leq$ are given as the input. We claim that $G^{\min} = G^{\min}_{<}$.

First note that if v is a simplicial vertex of G, then there are no arrowheads at v in $G^{\min}$. Otherwise, we could drop all arrowheads at simplicial vertices in $G^{\min}$ to obtain an ancestral graph (Lemma 11) with fewer arrowheads. The new graph would have the boundary containment property and thus be Markov equivalent to G (by Theorem 5). This would contradict the assumed minimality of $G^{\min}$.

The observation about simplicial vertices implies that an undirected edge in the simplicial graph G^s is also an undirected edge in $G^{\min}$. Conversely, if $v - w \in G^{\min}$ then there may not be an arrowhead at v on any other edge, and likewise for w, because $G^{\min}$ is ancestral. Since $G^{\min}$ has the boundary containment property, it follows from Proposition 7 that both v and w are simplicial vertices. This implies that $v - w \in G^s$ and we conclude that $G^{\min}$ and G^s have the same undirected edges. By construction, the same holds for $G^{\min}_{<}$ and G^s. Hence, $G^{\min}$ and $G^{\min}_{<}$ have the same undirected edges.

Suppose $v \to w \in G^{\min}$. Then $\mathrm{Bd}(v) \subseteq \mathrm{Bd}(w)$ because $G^{\min}$ has the boundary containment property. Moreover, $v < w$ because the total order $\leq$ extends $\preccurlyeq_D$. It follows that $v \to w \in G^{\min}_{<}$. In other words, every directed edge in $G^{\min}$ is also in $G^{\min}_{<}$. This together with the fact that $G^{\min}$ and $G^{\min}_{<}$ have the same skeleton and the same number of arrowheads, $\mathrm{arr}(G^{\min}) = \mathrm{arr}(G^{\min}_{<})$, implies that $G^{\min} = G^{\min}_{<}$. ■

4.2 Markov Equivalence Results

The following corollary is an immediate consequence of Proposition 12 because a minimally oriented graph $G^{\min}$ is an undirected graph iff G^s is an undirected graph.

Corollary 18 *Let $G^{\min}$ be a minimally oriented graph for a bi-directed graph G. If G is Markov equivalent to an undirected graph U, then $G^{\min} = U$ is the unique minimally oriented graph of G.*

A minimally oriented graph also reveals whether the original bi-directed graph is Markov equivalent to a DAG.

Theorem 19 *Let $G^{\min}$ be a minimally oriented graph for a bi-directed graph G. Then G is Markov equivalent to a DAG iff $G^{\min}$ contains no bi-directed edges.*

Proof Let G be a bi-directed graph such that $G^{\min}$ contains no bi-directed edges. If $A \subseteq V$ is a simplicial set, then the induced subgraph $(G^{\min})_A$ is undirected and complete (this follows directly from Theorem 17 and Algorithm 14). Let $A_1, \ldots, A_q$ be the inclusion-maximal simplicial sets of G. Let D be a directed graph obtained by replacing each induced subgraph $(G^{\min})_{A_i}$, $i = 1, \ldots, q$, by a complete DAG. Then D itself is acyclic, which can be seen as follows: First, since $G^{\min}$ does not contain any directed cycles, a directed cycle π in D must involve a vertex $v \in \cup_{i=1}^{q} A_i$. Let $v \in A_j$.

Since the induced subgraphs D_{A_i}, $i = 1,\dots,q$, are all acyclic, π must also involve a vertex not in A_j. Therefore, there exists an edge $x \to w$ on π such that $w \in A_j$ and $x \notin A_j$. Since the sets A_i are inclusion-maximal simplicial sets, no vertex in A_i, $i \neq j$, is adjacent to any vertex in A_j. Hence, $x \notin \cup_{i=1}^{q} A_i$, which implies that the edge $x \to w$ is also present in $G^{\min}$. This is a contradiction to w being a simplicial vertex.

Two vertices are adjacent in $G^{\min}$ iff they are adjacent in D. Moreover, D has the boundary containment property because $G^{\min}$ has this property, and if $u \to \bar{u}$ in D then either $u \to \bar{u}$ in $G^{\min}$ or $u - \bar{u}$ in $G^{\min}$. It thus follows from Theorem 5 that D is Markov equivalent to $G^{\min}$ and G.

Conversely, suppose that $v \leftrightarrow w \in G^{\min}$ and for a contradiction, that G is Markov equivalent to a DAG D. Note that D must have the same skeleton as G (and $G^{\min}$). By Lemma 15(iii), there exist two different vertices $x \in \mathrm{bd}(v) \setminus \{w\}$ and $y \in \mathrm{bd}(w) \setminus \{v\}$ such that, by the Markov property of G, $x \perp\!\!\!\perp w$ and $v \perp\!\!\!\perp y$. Hence, v and w must be colliders on the paths (x,v,w) and (v,w,y) in D, respectively. This is impossible in the DAG D. ■

Theorem 19 can be shown to be equivalent to a Markov equivalence result stated without proof in Theorem 1 in Pearl and Wermuth (1994). This latter theorem requires 'no chordless four-chain', which must be read as excluding graphs with induced subgraphs that are either of the graphs in Figure 4. Under this condition, Pearl and Wermuth (1994) also state that a Markov equivalent DAG can be constructed from the (undirected) skeleton of G by introducing directed and bi-directed edges in an operation they term 'sink orientation', and turning remaining undirected edges into directed ones. The sink orientation of the graph G_1 in Figure 3 has the directed edges of G_1^s but an undirected edge $v - w$. Thus sink orientation need not yield an ancestral graph. The bi-directed graphical models considered in Theorem 19 also appear in the construction of generalized Wishart distributions (Letac and Massam, 2007, Theorem 2.2). In that context the models are called *homogeneous* and characterized in terms of Hasse diagrams.

As the next result reveals, bi-directed graphs that are Markov equivalent to DAGs exhibit a structure that corresponds to a multivariate regression model. The graphs can also be termed chordal cographs; compare the paragraph before Theorem 16.

Proposition 20 *Let $G^{\min}$ be a minimally oriented graph for a connected bi-directed graph G. If $G^{\min}$ contains no bi-directed edges, then the set A of all simplicial vertices is non-empty, the induced subgraph $(G^{\min})_A$ is a disjoint union of complete undirected graphs, the induced subgraph $(G^{\min})_{V\setminus A}$ is a complete DAG, and an edge $v \to w$ joins any two vertices $v \in A$ and $w \notin A$ in $G^{\min}$.*

Proof For two adjacent vertices v and w in $G^{\min}$, Lemma 15(i)-(ii) implies that $\mathrm{Bd}(v) \subseteq \mathrm{Bd}(w)$ or $\mathrm{Bd}(w) \subseteq \mathrm{Bd}(v)$. Hence, we can list the vertex set as $V = \{v_1,\dots,v_p\}$ such that $\mathrm{Bd}(v_i) \subseteq \mathrm{Bd}(v_j)$ if v_i and v_j are adjacent and $i \leq j$. It follows that $v_1 \in A$ and thus $A \neq \emptyset$. Let $A_1,\dots,A_q$ be the inclusion-maximal simplicial sets of G. Then $(G^{\min})_A$ equals the union of the disjoint complete undirected graphs $(G^{\min})_{A_1},\dots,(G^{\min})_{A_q}$. Since $G^{\min}$ is an ancestral graph, $(G^{\min})_{V\setminus A}$ is a DAG.

We prove the remaining claims by induction on $|V \setminus A|$. If $|V \setminus A| = 0$, then the connected graph $G^{\min}$ is a complete undirected graph and there is nothing to show. Let $|V \setminus A| \geq 1$. It follows that $v_p \in V \setminus A$. If the shortest path between some vertex v_{i_1} and v_p in G is of the form $v_{i_1} \leftrightarrow \dots \leftrightarrow v_{i_k} \leftrightarrow v_p$, then $i_1 < \dots < i_k < p$ and $\mathrm{Bd}(v_{i_1}) \subseteq \dots \subseteq \mathrm{Bd}(v_{i_k}) \subseteq \mathrm{Bd}(v_p)$, which is easily shown by induction on k. However, since $v_{i_1} \in \mathrm{Bd}(v_{i_1})$ it must in fact hold that v_{i_1} and v_p are adjacent. Hence, there is an edge between every vertex $v \in V \setminus \{v_p\}$ and v_p, which for $v \in A$ is of

the form $v \to v_p$ because clearly $v_p \notin A$. The proof is finished by combining what we learned about v_p with the induction assumption applied to the induced subgraph G_W with $W = \{v_1, \ldots, v_{p-1}\}$. Note that for $v, w \in W$, the inclusion $\mathrm{Bd}_G(v) \subseteq \mathrm{Bd}_G(w)$ implies that $\mathrm{Bd}_{G_W}(v) \subseteq \mathrm{Bd}_{G_W}(w)$. Thus by Lemma 15 and Theorem 16, $(G_W)^{\min}$ does not contain any bi-directed edges. ■

5. Maximum Likelihood Estimation in Gaussian Models

In this section we consider the Gaussian covariance models associated with bi-directed graphs and demonstrate that the graphical constructions from Sections 3 and 4 can be employed for more efficient computation of maximum likelihood estimates.

5.1 Covariance Graphs and Gaussian Ancestral Graph Models

Let G be a bi-directed graph, and

$$\mathbf{P}(G) = \left\{ \Sigma \in \mathbb{R}^{V \times V} \mid \Sigma = (\sigma_{vw}) \text{ sym. pos. def., } \sigma_{vw} = 0 \; \forall (v,w) : v \leftrightarrow w \notin G \right\}$$

be the cone of symmetric positive definite matrices with zero pattern induced by G. The *covariance graph model* associated with G is the family of multivariate normal distributions $\mathbf{N}(G) = \big(\mathcal{N}(0,\Sigma) \mid \Sigma \in \mathbf{P}(G)\big)$. It can be shown that every distribution in $\mathbf{N}(G)$ satisfies all conditional independences stated by the global Markov property for the bi-directed graph G (Kauermann, 1996, Prop. 2.2). Conversely, if a distribution $\mathcal{N}(0,\Sigma)$ satisfies the global Markov property for G, then $\Sigma \in \mathbf{P}(G)$.

Let $S \in \mathbb{R}^{V \times V}$ be the empirical covariance matrix computed from an i.i.d. sample drawn from some unknown distribution $\mathcal{N}(0,\Sigma) \in \mathbf{N}(G)$, that is, the (v,w)-th entry in S is the dot-product of the vectors of observations for the v-th and w-th variables divided by the sample size n. The log-likelihood function $\ell_{S,n} : \mathbf{P}(G) \to \mathbb{R}$ of $\mathbf{N}(G)$ can be written as

$$\ell_{S,n}(\Sigma) = -\frac{n|V|}{2}\log(2\pi) - \frac{n}{2}\log|\Sigma| - \frac{n}{2}\mathrm{tr}\left(\Sigma^{-1}S\right).$$

If S is positive definite then the global maximum of $\ell_{S,n}$ over $\mathbf{P}(G)$ exists. The likelihood equations obtained by setting to zero the partial derivatives of $\ell_{S,n}$ with respect to the non-restricted entries in Σ take on the form

$$(\Sigma^{-1})_{vw} = (\Sigma^{-1}S\Sigma^{-1})_{vw} \quad \forall v,w \in V : \; v = w \;\text{ or }\; v \leftrightarrow w \in G; \tag{1}$$

compare Anderson and Olkin (1985, §2.1.1). A matrix $\hat{\Sigma}(S) \in \mathbf{P}(G)$ that solves (1) is *a solution to the likelihood equations* of $\mathbf{N}(G)$. Since subsequent theorems on the structure of the likelihood equations are obtained via Gaussian ancestral graph models, we briefly review the parametrization of these models.

Let G be an ancestral graph and $\mathrm{un}_G \subseteq V$ the set of vertices v that are such that any edge with endpoint v has a tail at v. By Definition 1(i), $v - w \in G$ implies $v, w \in \mathrm{un}_G$, and $v \leftrightarrow w \in G$ implies that $v, w \notin \mathrm{un}_G$. Let Λ be a symmetric positive definite $\mathrm{un}_G \times \mathrm{un}_G$ matrix such that $\Lambda_{vw} \neq 0$ only if $v = w$ or $v - w \in G$. Let Ω be a symmetric positive definite $(V \setminus \mathrm{un}_G) \times (V \setminus \mathrm{un}_G)$ matrix such that $\Omega_{vw} \neq 0$ only if $v = w$ or $v \leftrightarrow w \in G$. Finally, let B be a $V \times V$ matrix such that $B_{vw} \neq 0$ only if

$w \to v \in G$. Define the symmetric positive definite matrix

$$\Sigma(\Lambda, B, \Omega) = (I - B)^{-1} \begin{pmatrix} \Lambda^{-1} & 0 \\ 0 & \Omega \end{pmatrix} (I - B)^{-\mathrm{T}}, \tag{2}$$

where I is the identity matrix.

Let $\mathbf{N}(G)$ be the Gaussian ancestral graph model associated with G, that is, the family of all centered normal distributions that are globally Markov with respect to G. As shown in Richardson and Spirtes (2002, §8), the normal distribution $\mathcal{N}(0, \Sigma)$ with $\Sigma = \Sigma(\Lambda, B, \Omega)$ defined in (2) is in $\mathbf{N}(G)$. Conversely, if G is *maximal*, then for any $\mathcal{N}(0, \Sigma) \in \mathbf{N}(G)$ there exist unique Λ, Ω, B of the above type such that $\Sigma = \Sigma(\Lambda, B, \Omega)$. (Note that Richardson and Spirtes, 2002, use B for what is here denoted by $I - B$).

Since a bi-directed graph G and a minimally oriented graph $G^{\min}$ are Markov equivalent and maximal, the parametrization map for $G^{\min}$, $(\Lambda, B, \Omega) \mapsto \Sigma(\Lambda, B, \Omega)$, has image equal to $\mathbf{P}(G)$. By Richardson and Spirtes (2002, Theorem 8.14, Lemma 8.22), we obtain the following Lemma.

Lemma 21 *Let G be a bi-directed graph. The covariance matrix $\Sigma(\Lambda, B, \Omega)$ solves the likelihood equations of $\mathbf{N}(G)$ iff (Λ, B, Ω) solves the likelihood equations of $\mathbf{N}(G^{\min})$.*

5.2 Empirical Maximum Likelihood Estimates

Using the graphical results established earlier, we can show that over simplicial sets a solution to the likelihood equations (1) agrees with its empirical counterpart in S.

Theorem 22 *Let G be a bi-directed graph with associated covariance graph model $\mathbf{N}(G)$. If $A \subseteq V$ is simplicial, S is a symmetric positive definite matrix, and $\hat{\Sigma}(S) \in \mathbf{P}(G)$ is a solution to the likelihood equations (1), then $\hat{\Sigma}(S)_{A \times A} = S_{A \times A}$.*

Proof By Theorem 10, the covariance graph model $\mathbf{N}(G)$ and the Gaussian ancestral graph model $\mathbf{N}(G^s)$ based on the simplicial graph G^s are equal. Let $\mathbf{N}(G^s)$ be parametrized by the precision matrix Λ, the matrix of regression coefficients B and the covariance matrix Ω as described in §5.1. In particular, it follows from Richardson and Spirtes (2002, Lemma 8.4) that if $\Sigma = \Sigma(\Lambda, B, \Omega)$, then $(\Lambda^{-1})_{A \times A} = \Sigma_{A \times A}$.

The inclusion-maximal simplicial sets $A_1, \ldots, A_q$ of G form a partition of un_{G^s}. The induced subgraphs $G^s_{A_i}$, $i = 1, \ldots, q$, are complete undirected graphs. It follows that Λ is a block-diagonal matrix such that $\Lambda_{vw} = 0$ if there does not exist an inclusion-maximal simplicial set A_i such that $v, w \in A_i$. Now the discussion in Richardson and Spirtes (2002, §8.5) and Lemma 21 imply that every solution to the likelihood equations for Λ, B, Ω in the Gaussian ancestral graph model $\mathbf{N}(G^s)$ satisfies that $(\hat{\Lambda}^{-1})_{A_i \times A_i} = S_{A_i \times A_i}$ for all $i = 1, \ldots, q$. Since $A \subseteq A_j$ for some j, it holds that $\hat{\Sigma}_{A \times A} = (\hat{\Lambda}^{-1})_{A \times A} = S_{A \times A}$. ■

Our graphical constructions also provide information on when maximum likelihood estimates of conditional parameters are equal to their empirical counterparts. The conditional parameters we consider are the regression coefficients and conditional variance for the conditional distribution of variable v given its *parents* $\mathrm{pa}(v) = \{w \in V \mid w \to v \in G^{\min}\}$ in a minimally oriented graph $G^{\min}$. If $\mathrm{pa}(v) = \emptyset$, then conditioning variable v on $\mathrm{pa}(v)$ is understood to yield the marginal distribution of v.

Theorem 23 *Let $G^{\min}$ be a minimally oriented graph for a bi-directed graph G, S a symmetric positive definite matrix, and $\hat{\Sigma}(S) \in \mathbf{P}(G)$ a solution to the likelihood equations (1). If v is a vertex such that there is no vertex w with $v \leftrightarrow w \in G^{\min}$, then the regression coefficients for v given $\mathrm{pa}(v)$ are*

$$\hat{\Sigma}(S)_{v\times \mathrm{pa}(v)} \left[\hat{\Sigma}(S)_{\mathrm{pa}(v)\times \mathrm{pa}(v)}\right]^{-1} = S_{v\times \mathrm{pa}(v)} \left(S_{\mathrm{pa}(v)\times \mathrm{pa}(v)}\right)^{-1}, \tag{3}$$

and that the conditional variance for v given $\mathrm{pa}(v)$ is

$$\hat{\Sigma}(S)_{vv} - \hat{\Sigma}(S)_{v\times \mathrm{pa}(v)} \left[\hat{\Sigma}(S)_{\mathrm{pa}(v)\times \mathrm{pa}(v)}\right]^{-1} \hat{\Sigma}(S)_{\mathrm{pa}(v)\times v} = S_{vv} - S_{v\times \mathrm{pa}(v)} \left(S_{\mathrm{pa}(v)\times \mathrm{pa}(v)}\right)^{-1} S_{\mathrm{pa}(v)\times v}. \tag{4}$$

Proof If $\mathrm{pa}(v) = \emptyset$, then v is a simplicial vertex, and the claim reduces to $\hat{\Sigma}(S)_{vv} = S_{vv}$, which follows from Theorem 22. Otherwise, using the parametrization of $\mathbf{N}(G^{\min})$, it follows from Richardson and Spirtes (2002, Theorem 8.7) that if $\Sigma = \Sigma(\Lambda, B, \Omega)$, then

$$\Sigma_{v\times \mathrm{pa}(v)} \left[\Sigma_{\mathrm{pa}(v)\times \mathrm{pa}(v)}\right]^{-1} = B_{v\times \mathrm{pa}(v)}$$

and

$$\Sigma_{vv} - \Sigma_{v\times \mathrm{pa}(v)} \left[\Sigma_{\mathrm{pa}(v)\times \mathrm{pa}(v)}\right]^{-1} \Sigma_{\mathrm{pa}(v)\times v} = \Omega_{vv}.$$

If $\hat{\Lambda}$, $\hat{B}$, $\hat{\Omega}$ solve the likelihood equations for $\mathbf{N}(G^{\min})$, then $\hat{B}_{v\times \mathrm{pa}(v)}$ and $\hat{\Omega}_{vv}$ solve the likelihood equations of the model in which all parameters in Λ, B, Ω except for $B_{v\times \mathrm{pa}(v)}$ and Ω_{vv} are held fixed. It follows from Drton and Richardson (2004b, §§5.1-2) that $\hat{B}_{v\times \mathrm{pa}(v)}$ and $\hat{\Omega}_{vv}$ are equal to the empirical expressions on the right hand side of (3) and (4), respectively. Applying Lemma 21 yields the claim. ■

Remark 24 *Iterative Conditional Fitting* is a special purpose algorithm for maximum likelihood estimation in covariance graph models (Drton and Richardson, 2003; Chaudhuri et al., 2007). However, it does not exploit the results of Theorems 22 and 23. On the other hand, if one runs the ancestral graph extension of iterative conditional fitting described in Drton and Richardson (2004b) on a minimally oriented graph, then unnecessary computations are avoided by implicitly exploiting Theorems 22 and 23. This is illustrated in the example in Section 5.3.

If a bi-directed graph G has a minimally oriented graph $G^{\min}$ without bi-directed edges then G is Markov equivalent to a DAG (Theorem 19) and the likelihood equations have a unique solution that is a rational function of the empirical covariance matrix S. However, this is no longer true if there is a bi-directed edge in $G^{\min}$. In this case, G contains one of the two graphs in Figure 4 as a subgraph; compare Lemma 15(iii). Solving the likelihood equations for the bi-directed four-chain in Figure 4(i) is equivalent to computing the roots of a quintic polynomial. There exist data for which this quintic has exactly three real roots (Drton and Richardson, 2004a). Galois theory (Stewart, 1989, Lemma 14.7) implies that for these data the quintic is unsolvable by radicals, that is, the roots of the quintic and thus the solutions to the likelihood equations cannot be computed from the data in finitely many steps involving addition, subtraction, multiplication, division, or taking r-th roots. (Geiger et al., 2006, obtain similar results in the context of undirected graphs). Similarly, solving the likelihood equations of the bi-directed four-cycle in Figure 4(ii) corresponds to solving

	GAL7	GAL10	GAL1	GAL3	GAL2	GAL80	GAL11	GAL4
GAL7	*1.000*	0.91	0.88	0.50	0.81	0.21	-0.07	-0.08
GAL10	0.910	*1.000*	0.92	0.46	0.87	0.26	-0.08	-0.07
GAL1	0.880	0.920	*1.000*	0.39	0.87	0.28	-0.10	-0.10
GAL3	0.489	0.447	0.374	*0.998*	0.44	0.20	-0.18	0.12
GAL2	0.807	0.865	0.865	0.422	*0.991*	0.26	-0.18	-0.03
GAL80	0.224	0.271	0.297	0.191	0.280	*1.001*	0.08	0.23
GAL11	0	0	0	-0.208	-0.103	0	*1.022*	0.24
GAL4	0	0	0	0	0.038	0.209	0.255	*0.987*

Table 1: Gene expression data. Empirical correlation matrix (above diagonal) and maximum likelihood estimate (below diagonal). The italicized diagonal entries are ratios between maximum likelihood and empirical variance estimates.

a polynomial equation system of degree 17. This can be verified in computer algebra systems such as `Singular` (Greuel et al., 2005); see also Drton and Sullivant (2007, §5). It is natural to conjecture that there exist data for which this system is also unsolvable by radicals.

5.3 Example: Gene Expression Measurements

The application of covariance graph models to gene expression data has been promoted in Butte et al. (2000). For illustration, we select data from microarray experiments with yeast strands (Gasch et al., 2000). We focus on eight genes involved in galactose utilization. Expression measurements for all eight genes are available in $n = 134$ experiments, for which the empirical correlation matrix is shown in the upper-diagonal part of Table 1.

For these data, the covariance graph model induced by the graph G in Figure 5(i) has a deviance of 8.87 over 8 degrees of freedom, which indicates a good model fit; the p-value computed using a chi-square distribution is 0.35. Figure 5(ii) shows the unique minimally oriented graph $G^{\min}$. The maximum likelihood estimate obtained by fitting the model to the correlation matrix is shown in the lower-diagonal part of Table 1; note that this estimate is not a correlation matrix (not all the italicized diagonal entries are equal to one). As predicted by Theorem 22, the submatrix over GAL1, GAL7, and GAL10 equals the respective submatrix in the empirical correlation matrix. The regression coefficients for the regression of GAL2 on all remaining variables are identical when computed from the maximum likelihood versus the empirical estimate (Theorem 23).

The use of a minimally oriented graph $G^{\min}$ leads to a considerable gain in computational efficiency in the iterative calculation of the maximum likelihood estimate $\hat{\Sigma}$. With the identity matrix as starting value, iterative conditional fitting (Remark 24) on the original bi-directed graph G performs eight multiple regressions per iteration and converges after 103 iterations. Using the same starting value and termination criterion, iterative conditional fitting on $G^{\min}$ converges after only 5 iterations and requires only five multiple regressions per iteration (for the genes GAL2, GAL3, GAL4, GAL11, and GAL80), of which the one for GAL2 has to be executed only in the first iteration.

As in any application of covariance graph models, one might question the assumption of Gaussianity. Indeed there are 10 experiments in which the measurements for the genes GAL1, GAL7, GAL10 and GAL80 come out to be large negative values, and one in which GAL7 alone takes such

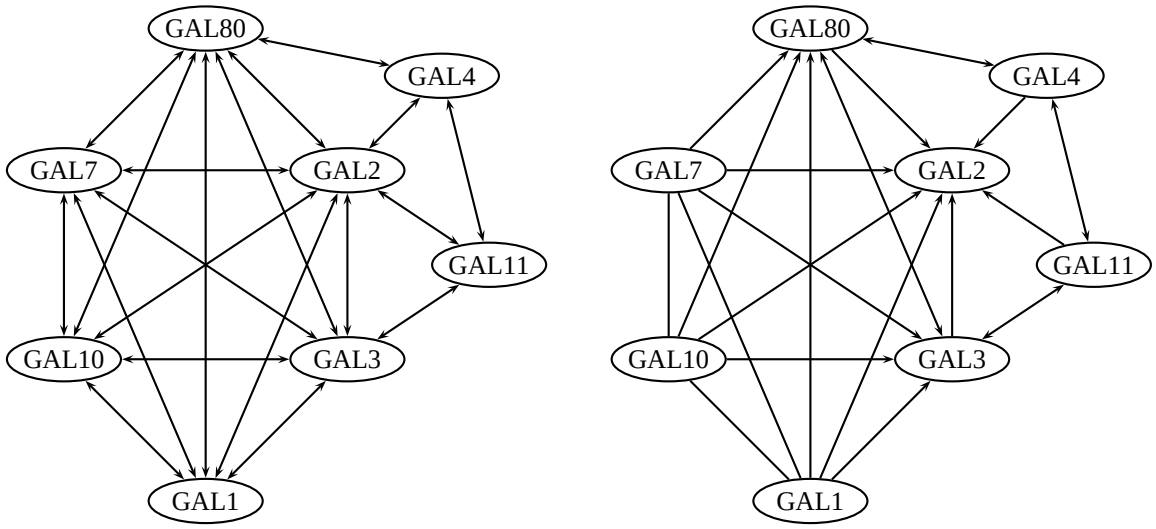


Figure 5: (i) Bi-directed graph G for gene expression measurements, (ii) the unique minimally oriented graph $G^{\min}$.

a value. These appear to be outliers (standardized values between -3 and -5), possibly produced by thresholding, as some values are identical. However, the measurements for the other genes are well within the range of the observations for the remaining 123 experiments. Thus it is unclear whether removing these 11 experiments from consideration is appropriate. If the 11 experiments are removed, then the correlations among GAL1, GAL7 and GAL10 decrease to values between 0.38 and 0.60, the latter value is the maximum of all correlations. Nevertheless the deviance for G only changes slightly to 10.09 (p-value 0.26). The iterative conditional fitting algorithm based on G now converges after only 20 iterations rather than 103. However, this is still four times as many iterations as required in iterative conditional fitting based on the minimally oriented graph $G^{\min}$; recall that in addition each iteration is also simpler.

The original correlation matrix in Table 1 exhibits an apparent similarity of the rows for GAL1, GAL7 and GAL10; this is also reflected in the graph G in which these variables form a complete set and have the same spouses. Such symmetry could be investigated further via a group symmetry model (Andersson and Madsen, 1998).

6. Conclusion

We showed how to remove a maximal number of arrowheads from the edges of a bi-directed graph G such that one obtains a maximal ancestral graph $G^{\min}$ that is Markov equivalent to G. The graph $G^{\min}$, called a minimally oriented graph, reveals whether G is Markov equivalent to an undirected graph, and also whether G is Markov equivalent to a DAG.

For the (Gaussian) covariance graph model associated with G, a minimally oriented graph $G^{\min}$ yields an alternative parametrization that provides insight into likelihood inference. The structure of the arrowheads in $G^{\min}$ allowed us to identify parts of the covariance matrix for which the maximum

likelihood estimates are equal to their empirical counterparts (this applies to all solutions to the likelihood equations if, as occasionally happens, there is more than one solution). This makes it possible to avoid or speed up iterative estimation of the full covariance matrix. We also saw that the maximum likelihood estimator of the covariance matrix in a covariance graph model is a rational function of empirical covariance matrix if $G^{\min}$ contains no bi-directed edge. This is similar to the results that identify decomposable models as the sub-class of all log-linear and all covariance selection models (Dempster, 1972) for which the maximum likelihood estimator is available in closed form.

Drton and Richardson (2008) formulate binary models based on the Markov property of bi-directed graphs. For these models, the maximum likelihood estimator is available in closed form if the model-inducing graph is Markov equivalent to a DAG. Moreover, we verified that in the example of the graph G in Figure 1, the maximum likelihood estimates of the marginal distributions of X_1 and X_4 are equal to the corresponding empirical proportions. We thus believe that analogs to the Gaussian results established here will hold in discrete models, but a general parametrization of discrete ancestral graph models is required to fully access the potential of the results obtained in this paper.

Acknowledgments

This paper is based upon work supported by the U.S. National Science Foundation (DMS-0505612 and 0505865) and the U.S. National Institutes for Health (R01-HG2362-3). We would also like to thank two anonymous referees for very helpful comments on the presentation of our work.

Appendix A. Connecting Paths and Boundary Containment

In this appendix we prove results about graphs that satisfy the *boundary containment property* from Definition 4. These results are used in the proof of Theorem 5.

Let v and w be two fixed distinct vertices that are m-connected given $C \subseteq V \setminus \{v,w\}$ in a simple mixed graph G. Define $\Pi_G(v,w|C)$ to be the set of paths that m-connect v and w given C in G, and let $\Pi_G^{\min}(v,w|C)$ be the set of paths that are of minimal length among the paths in $\Pi_G(v,w|C)$.

Lemma 25 *If a simple mixed graph G satisfies the boundary containment property, v_{i-1}, v_i and v_{i+1} are three consecutive vertices on a path π in G, and v_i is a non-collider on π, then v_{i-1} and v_{i+1} are adjacent.*

Proof If v_i is a non-collider, then the edge between v_i and v_{i-1} or the edge between v_i and v_{i+1} must have a tail at v_i. Suppose, without loss of generality, that the latter is the case. Then $\mathrm{Bd}(v_i) \subseteq \mathrm{Bd}(v_{i+1})$ and thus $v_{i-1} \in \mathrm{Bd}(v_{i+1})$, which is the claim. ■

Lemma 26 *Let G be a simple mixed graph, and $\pi = (v, v_1, \ldots, v_k, w) \in \Pi_G(v,w|C)$. Let $v_0 = v$ and $v_{k+1} = w$. If v_i is a non-endpoint vertex on π and there is an arrowhead at v_i on the edge between v_{i-1} and v_i, then either (i) $v_i \in \mathrm{An}(C)$ or (ii) the path $(v_i, v_{i+1}, \ldots, v_k, w)$ is a directed path from v_i to w.*

Proof Suppose the result is false. Let v_j be the vertex closest to w satisfying the antecedent of the Lemma, but not the conclusion. If v_j is a collider, then by definition of m-connection, $v_j \in \mathrm{An}(C)$, which is a contradiction. If v_j is a non-collider then $v_j \to v_{j+1}$ on π. If $v_{j+1} = w$, if $v_{j+1} \in \mathrm{An}(C)$, or if $(v_{j+1}, \ldots, v_k, w)$ is a directed path from v_{j+1} to w, then clearly v_j satisfies the conclusion of the Lemma, which is a contradiction. But if $v_{j+1} \notin \mathrm{An}(C)$ and $(v_{j+1}, \ldots, v_k, w)$ is not a directed path from v_{j+1} to w then v_{j+1} satisfies the conditions on v_j, but is closer to w, again a contradiction. ■

Lemma 27 *If G is an ancestral graph that satisfies the boundary containment property and $\pi = (v, v_1, \ldots, v_k, w) \in \Pi_G^{\min}(v, w|C)$ then no non-consecutive vertices on π are adjacent.*

Proof Let $v_0 = v$ and $v_{k+1} = w$, and suppose for a contradiction that there are non-consecutive vertices on the path π which are adjacent. Let (v_p, v_q), $p < q$, be a pair of adjacent vertices which are furthest apart on the path, that is, (p, q) maximizes the distance $|r - s|$ among pairs of indices of adjacent vertices v_r and v_s on the path. Since π is of minimal length, either $v \neq v_p$ or $w \neq v_q$.

Suppose that $v \neq v_p$. By definition of (p, q), v_{p-1} is not adjacent to v_q. Consequently, by Lemma 25, v_p is a collider on (v_{p-1}, v_p, v_q), and thus the edge between v_{p-1} and v_p has an arrowhead at v_p. It then follows by Lemma 26 that either $v_p \in \mathrm{An}(C)$ or $(v_p, v_{p+1}, \ldots, v_k, w)$ is a directed path from v_p to w. In the latter case $v_p \in \mathrm{An}(v_q)$, but there is an arrowhead at v_p on the edge between v_p and v_q, which contradicts that G is ancestral. Hence $v_p \in \mathrm{An}(C)$. If $v_q = w$ then the path $(v, v_1, \ldots, v_p, v_q = w)$ is m-connecting given C and shorter than π. Hence $v_q \neq w$. It then follows by the same argument that v_q is a collider on (v_p, v_q, v_{q+1}) and in $\mathrm{An}(C)$. However, this also leads to a contradiction since then the path $(v, v_1, \ldots, v_p, v_q, v_{q+1}, \ldots, v_k, w)$ is both m-connecting given C and shorter than π.

The case where $w \neq v_q$ may be argued symmetrically. ■

Corollary 28 *If G is an ancestral graph that satisfies the boundary containment property and $\pi = (v = v_0, v_1, \ldots, v_k, v_{k+1} = w) \in \Pi_G^{\min}(v, w|C)$, then all the non-endpoint vertices $v_1, \ldots, v_k$ are colliders on π.*

Proof This follows directly from Lemma 27 and Lemma 25. ■

Even though all non-endpoints on a path of the type described in Corollary 28 in $\Pi_G^{\min}(v, w|C)$ are colliders, not all non-endpoints must be in the set C. For example, in the graph $G_2^{\min}$ from Figure 3, the path (x, v, y) m-connects x and y given $\{w\}$ since the collider v is an ancestor of w. However, as the next Lemma shows, there will always exist a path in $\Pi_G^{\min}(v, w|C)$ such that all non-endpoints are colliders in C. In $G_2^{\min}$ from Figure 3, the path (x, w, y) m-connects x and y given $\{w\}$.

Lemma 29 *If G is an ancestral graph that satisfies the boundary containment property, and $\pi = (v = v_0, v_1, \ldots, v_k, v_{k+1} = w) \in \Pi_G^{\min}(v, w|C)$ is such that no other path in $\Pi_G^{\min}(v, w|C)$ has more non-endpoint vertices in C than π, then all non-endpoint vertices $v_1, \ldots, v_k$ on π are colliders that are in C.*

Proof By Corollary 28, all non-endpoints $v_1, \ldots, v_k$ are colliders. Assume that there exists $v_i \notin C$, $1 \leq i \leq k$. Since $\pi \in \Pi_G(v, w|C)$, and thus $v_i \in \mathrm{An}(C)$, there exists $c \in C$ such that $v_i \to \cdots \to c \in G$.

In particular, $c \neq v_{i-1}$ and $c \neq v_{i+1}$ because v_i is ancestral neither to v_{i-1} nor to v_{i+1}. The boundary containment property and the fact that G does not contain directed cycles imply that $v_i \to c \in G$. By Lemma 25, G contains edges between c and both v_{i-1} and v_{i+1}. Since the edge between v_{i-1} and v_i has an arrowhead at v_i and $v_i \to c \in G$, the edge between v_{i-1} and c must have an arrowhead at c because otherwise the fact that G is an ancestral graph would be contradicted. Similarly, the edge between v_{i+1} and c must have an arrowhead at c. If $v_{i-1} \to c$, then $v_i \neq v$, v_{i-2} is adjacent to c and by the same argument as above there must be an arrowhead at c on the edge between v_{i-2} and c. Repeating this argument yields that there exists a vertex v_ℓ, $\ell \leq i-1$, such that either $v_\ell \leftrightarrow c \in G$, or $v_\ell = v$ and $v \to c$. The same arguments also imply that there exists a vertex v_j, $j \geq i+1$, such that either $v_j \leftrightarrow c \in G$, or $v_j = w$ and $w \to c$. Therefore, the path $(v, v_1, \ldots, v_\ell, c, v_j, \ldots, v_k, w)$ is in $\Pi_G(v, w|C)$ and is either shorter than π or of equal length but with more non-endpoint vertices in C. This contradicts the choice of π and therefore the assumption of a non-endpoint on π that is not in C must be false. ■

References

T. W. Anderson. Asymptotically efficient estimation of covariance matrices with linear structure. *Ann. Statist.*, 1:135–141, 1973.

T. W. Anderson and I. Olkin. Maximum-likelihood estimation of the parameters of a multivariate normal distribution. *Linear Algebra Appl.*, 70:147–171, 1985.

S. A. Andersson and J. Madsen. Symmetry and lattice conditional independence in a multivariate normal distribution. *Ann. Statist.*, 26:525–572, 1998.

A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.

A. J. Butte, P. Tamayo, D. Slonim, T. R. Golub, and I. S. Kohane. Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks. *Proc. Nat. Acad. Sci. USA*, 97:12182–12186, 2000.

S. Chaudhuri, M. Drton, and T. S. Richardson. Estimation of a covariance matrix with zeros. *Biometrika*, 94:199–216, 2007.

D. G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14:926–934, 1985.

D. R. Cox and N. Wermuth. *Multivariate Dependencies: Models, Analysis and Interpretation*. Chapman and Hall, London, 1996.

D. R. Cox and N. Wermuth. Linear dependencies represented by chain graphs (with discussion). *Statist. Sci.*, 8:204–218,247–277, 1993.

A. P. Dempster. Covariance selection. *Biometrics*, 28:157–175, 1972.

M. Drton and T. S. Richardson. A new algorithm for maximum likelihood estimation in Gaussian graphical models for marginal independence. In U. Kjærulff and C. Meek, editors, *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, pages 184–191. Morgan Kaufmann, San Francisco, 2003.

M. Drton and T. S. Richardson. Multimodality of the likelihood in the bivariate seemingly unrelated regressions model. *Biometrika*, 91:383–392, 2004a.

M. Drton and T. S. Richardson. Iterative conditional fitting for Gaussian ancestral graph models. In M. Chickering and J. Halpern, editors, *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 130–137. Morgan Kaufmann, San Francisco, 2004b.

M. Drton and T. S. Richardson. Binary models for marginal independence. *J. Roy. Statist. Soc. Ser. B*, 70:287–309, 2008.

M. Drton and S. Sullivant. Algebraic statistical models. *Statist. Sinica*, 17:1273–1297, 2007.

D. M. Edwards. *Introduction to Graphical Modelling*. Springer-Verlag, New York, second edition, 2000.

A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11:4241–4257, 2000.

D. Geiger, C. Meek, and B. Sturmfels. On the toric algebra of graphical models. *Ann. Statist.*, 34: 1463–1492, 2006.

G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3.0. A Computer Algebra System for Polynomial Computations, Centre for Computer Algebra, University of Kaiserslautern, 2005. `http://www.singular.uni-kl.de`.

F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York, 2001.

S. T. Jensen. Covariance hypotheses which are linear in both the covariance and the inverse covariance. *Ann. Statist.*, 16:302–322, 1988.

G. Kauermann. On a dualization of graphical Gaussian models. *Scand. J. Statist.*, 23:105–116, 1996.

J. T. A. Koster. On the validity of the Markov interpretation of path diagrams of Gaussian structural equation systems with correlated errors. *Scand. J. Statist.*, 26:413–431, 1999.

S. L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, UK, 1996.

G. Letac and H. Massam. Wishart distributions for decomposable graphs. *Ann. Statist.*, 35:1278–1323, 2007.

D. Madigan and K. Mosurski. An extension of the results of Asmussen and Edwards on collapsibility in contingency tables. *Biometrika*, 77:315–319, 1990.

Y. Mao, F. R. Kschischang, and B. J. Frey. Convolutional factor graphs as probabilistic models. In U. Kjærulff and C. Meek, editors, *Procceding of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 374–381. Morgan Kaufmann, San Francisco, 2004.

F. Matúš. Stochastic independence, algebraic independence and abstract connectedness. *Theoretical Computer Science*, 134:455–471, 1994.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

J. Pearl and N. Wermuth. When can association graphs admit a causal interpretation? In *Selecting Models from Data: Artificial Intelligence and Statistics IV*, volume 89 of *Lecture Notes in Statistics*, pages 205–214. Springer, New York, 1994.

T. S. Richardson. Markov properties for acyclic directed mixed graphs. *Scand. J. Statist.*, 30: 145–157, 2003.

T. S. Richardson and P. Spirtes. Ancestral graph Markov models. *Ann. Statist.*, 30:962–1030, 2002.

T. S. Richardson and P. Spirtes. Causal inference via ancestral graph models (with discussion). In P. Green, N. Hjort, and S. Richardson, editors, *Highly Structured Stochastic Systems*, chapter 3, pages 83–105. Oxford University Press, Oxford, UK, 2003.

A. Roverato. A unified approach to the characterization of equivalence classes of DAGs, chain graphs with no flags and chain graphs. *Scand. J. Statist.*, 32:295–312, 2005.

I. Stewart. *Galois Theory*. Chapman and Hall, London, second edition, 1989.

N. Wermuth, D. R. Cox, and G. Marchetti. Covariance chains. *Bernoulli*, 12:841–862, 2006.

S. Wright. The method of path coefficients. *Ann. Math. Statist.*, 5:161–215, 1934.

Journal of Machine Learning Research 9 (2008) 915-936 Submitted 5/07; Revised 1/08; Published 5/08

Bouligand Derivatives and Robustness of Support Vector Machines for Regression

Andreas Christmann ANDREAS.CHRISTMANN@UNI-BAYREUTH.DE
Department of Mathematics
University of Bayreuth
D-95440 Bayreuth, GERMANY

Arnout Van Messem AVMESSEM@VUB.AC.BE
Department of Mathematics
Vrije Universiteit Brussel
B-1050 Brussels, BELGIUM

Editor: Peter Bartlett

Abstract

We investigate robustness properties for a broad class of support vector machines with non-smooth loss functions. These kernel methods are inspired by convex risk minimization in infinite dimensional Hilbert spaces. Leading examples are the support vector machine based on the ε-insensitive loss function, and kernel based quantile regression based on the pinball loss function. Firstly, we propose with the Bouligand influence function (BIF) a modification of F.R. Hampel's influence function. The BIF has the advantage of being positive homogeneous which is in general not true for Hampel's influence function. Secondly, we show that many support vector machines based on a Lipschitz continuous loss function and a bounded kernel have a bounded BIF and are thus robust in the sense of robust statistics based on influence functions.

Keywords: Bouligand derivatives, empirical risk minimization, influence function, robustness, support vector machines

1. Introduction

The goal in non-parametric regression is to estimate a functional relationship between an $\mathbb{R}^d$-valued input random variable X and an $\mathbb{R}$-valued output random variable Y, under the assumption that the joint distribution P of (X,Y) is (almost) completely unknown. In order to model this relationship one typically assumes that one has a training data set $D_{train} = \big((x_1,y_1),\ldots,(x_n,y_n)\big)$ from independent and identically distributed (i.i.d.) random variables (X_i,Y_i), $i = 1,\ldots,n$, which all have the distribution P. Informally, the aim is to build a predictor $f : \mathbb{R}^d \to \mathbb{R}$ based on these observations such that $f(X)$ is a good approximation of Y. To formalize this aim one uses a continuous *loss function* $L : Y \times \mathbb{R} \to [0,\infty)$ that assesses the quality of a prediction $f(x)$ for an observed output y by $L(y,f(x))$. We follow the convention that the smaller $L(y,f(x))$ is, the better the prediction is. The quality of a predictor f is measured by the *L-risk* $\mathcal{R}_{L,\mathrm{P}}(f) := \mathbb{E}_{\mathrm{P}} L(Y,f(X))$ which of course is unknown, because P is unknown. One tries to find a predictor whose risk is close to the minimal risk, that is to the Bayes risk $\mathcal{R}^*_{L,\mathrm{P}} := \inf\{\mathcal{R}_{L,\mathrm{P}}(f);\, f : \mathbb{R}^d \to \mathbb{R} \text{ measurable}\}$. One way to build a non-parametric predictor f is to use a support vector machine (SVM) which finds a minimizer $f_{\mathrm{P},\lambda}$

of the regularized risk

$$\mathcal{R}^{reg}_{L,P,\lambda}(f) := \mathbb{E}_P L(Y, f(X)) + \lambda \|f\|^2_{\mathcal{H}}, \tag{1}$$

where $\lambda > 0$ is a regularization parameter to reduce the danger of overfitting, $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) of a measurable kernel $k : X \times X \to \mathbb{R}$, and L is a measurable, *convex* loss function in the sense that $L(y,\cdot) : \mathbb{R} \to [0,\infty)$ is convex for all $y \in Y$, see Vapnik (1998) and Schölkopf and Smola (2002). Since (1) is strictly convex in f, the minimizer $f_{P,\lambda}$ is unique if it exists. We denote the canonical feature map by $\Phi : \mathcal{H} \to \mathcal{H}$, $\Phi(x) := k(\cdot,x)$. The reproducing property gives $f(x) = \langle f, \Phi(x)\rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$ and $x \in X$. A kernel k is bounded, if $\|k\|_\infty := \sup\{\sqrt{k(x,x)} : x \in X\} < \infty$. Using the reproducing property and $\|\Phi(x)\|_{\mathcal{H}} = \sqrt{k(x,x)}$, one obtains the well-known inequalities

$$\|f\|_\infty \le \|k\|_\infty \|f\|_{\mathcal{H}} \quad \text{and} \quad \|\Phi(x)\|_\infty \le \|k\|_\infty \|\Phi(x)\|_{\mathcal{H}} \le \|k\|^2_\infty \tag{2}$$

for $f \in \mathcal{H}$ and $x \in X$. The Gaussian radial basis function kernel defined by $k_{RBF}(x,x') = \exp(-\|x - x'\|^2/\gamma^2)$, $\gamma > 0$, is bounded and universal on every compact subset of $\mathbb{R}^d$ (Steinwart, 2001) which partially explains its popularity. The corresponding RKHS of this kernel has infinite dimension. Of course, $\mathcal{R}^{reg}_{L,P,\lambda}(f)$ is not computable, because P is unknown. However, the empirical distribution $D = \frac{1}{n}\sum_{i=1}^n \delta_{(x_i,y_i)}$ corresponding to the training data set D_{train} can be used as an estimator of P. Here $\delta_{(x_i,y_i)}$ denotes the Dirac distribution in (x_i,y_i). If we replace P by D in (1), we obtain the regularized empirical risk

$$\mathcal{R}^{reg}_{L,D,\lambda}(f) := \mathbb{E}_D L\big(Y, f(X)\big) + \lambda \|f\|^2_{\mathcal{H}}.$$

An empirical SVM f_{D,λ_n} with $\lambda_n > 0$ and $\lambda_n \to 0$ if $n \to \infty$, is called L-risk consistent if $\mathcal{R}_{L,P}(f_{D,\lambda_n}) \to \mathcal{R}^*_{L,P}$ in probability for $n \to \infty$.

Traditionally, research in nonparametric regression is often based on the least squares loss $L_{LS}(y,t) := (y-t)^2$. The least squares loss function is convex in t, is useful to estimate the conditional mean function, and is advantageous from a numerical point of view, but L_{LS} is not Lipschitz continuous. From a practical point of view there are situations in which a different loss function is more appropriate. (i) In some situations one is actually not interested in modeling the conditional mean, but in fitting a conditional quantile function instead. For this purpose the convex pinball loss function $L_{\tau-pin}(y,t) := (\tau-1)(y-t)$, if $y - t < 0$, and $L_{\tau-pin}(y,t) := \tau(y-t)$, if $y - t \ge 0$, is used, where $\tau \in (0,1)$ specifies the desired conditional quantile, see Koenker and Bassett (1978) and Koenker (2005) for parametric quantile regression and Takeuchi et al. (2006) for nonparametric quantile regression. (ii) If the goal is to estimate the conditional median function, then the ε-insensitive loss given by $L_\varepsilon(y,t) := \max\{|y-t| - \varepsilon, 0\}$, $\varepsilon \in (0,\infty)$, promises algorithmic advantages in terms of sparseness compared to the L1-loss function $L_{L1}(y,t) = |y-t|$, see Vapnik (1998) and Schölkopf and Smola (2002). (iii) If the regular conditional distribution of Y given $X = x$ is known to be symmetric, basically all invariant loss functions of the form $L(y,t) = \psi(r)$ with $r = y - t$, where $\psi : \mathbb{R} \to [0,\infty)$ is convex, symmetric and has its only minimum at 0, can be used to estimate the conditional mean, see Steinwart (2007). In this case a less steep loss function such as the Lipschitz continuous Huber loss function given by $L_{c-Huber}(y,t) := \psi(r) = r^2/2$, if $|r| \le c$, and $\psi(r) = c|r| - c^2/2$, if $|r| > c$ for some $c \in (0,\infty)$, may be more suitable if one fears outliers in y-direction, see Huber (1964) and Christmann and Steinwart (2007).

The deeper reason to consider Lipschitz continuous loss functions is the following. One strong argument in favor of SVMs is that they are L-risk consistent under weak assumptions, that is SVMs

are able to "learn", but it is also important to investigate the robustness properties for such statistical learning methods. In almost all cases statistical models are only approximations to the true random process which generated a given data set. Hence the natural question arises what impact such deviations may have on the results. J.W. Tukey, one of the pioneers of robust statistics, mentioned already in 1960 (Hampel et al., 1986, p. 21): *"A tacit hope in ignoring deviations from ideal models was that they would not matter; that statistical procedures which were optimal under the strict model would still be approximately optimal under the approximate model. Unfortunately, it turned out that this hope was often drastically wrong; even mild deviations often have much larger effects than were anticipated by most statisticians."*

Let us consider $T(\mathrm{P}) := \mathcal{R}^{reg}_{L,\mathrm{P},\lambda}(f)$, with P a probability measure, as a mapping $T : \mathrm{P} \mapsto \mathcal{R}^{reg}_{L,\mathrm{P},\lambda}(f)$. In robust statistics we are interested in smooth and bounded functions T, because this will give stable regularized risks within small neighborhoods of P. If an appropriate derivative $\nabla T(\mathrm{P})$ of $T(\mathrm{P})$ is bounded, then the function $T(\mathrm{P})$ cannot increase or decrease unlimited in small neighborhoods of P. Several notions of differentiability have been used for this purpose.

Let us therefore take a look at the following results from Averbukh and Smolyanov (1967, 1968), Fernholz (1983) and Rieder (1994) on various notions of differentiation to clarify the connections between these notions. For every pair of normed real vector spaces (X,Y) let a subset $\mathcal{S}(X,Y)$ of the functions from X to Y be given. The following conditions are imposed on this system $\mathcal{S}$, which will provide the (Landau) o remainder of the first-order Taylor approximation of an $\mathcal{S}$-differentiation: (i) $\rho(0) = 0$, $\rho \in \mathcal{S}(X,Y)$, (ii) $\mathcal{S}(X,Y)$ is a real vector subspace of all functions from X to Y, (iii) $\mathcal{S}(X,Y) \cap \mathcal{L}(X,Y) = \{0\}$ where $\mathcal{L}(X,Y)$ is the space of continuous linear mappings from X to Y, and 0 stands for the zero operator and (iv) moreover, in case $X = \mathbb{R}$, it is required that $\mathcal{S}(\mathbb{R},Y) = \{\rho : \mathbb{R} \to Y | \lim_{t\to 0} \rho(t)/t = 0\}$. If $\mathcal{S}$ fulfills (i) to (iv), then some mapping $T : X \to Y$ is called *$\mathcal{S}$-differentiable* at x if there exists some $A \in \mathcal{L}(X,Y)$ and $\rho \in \mathcal{S}(X,Y)$ such that for all $h \in X$, $T(x+h) = T(x) + Ah + \rho(h)$. The continuous linear mapping $\nabla^{\mathcal{S}} T(x) = A$ is called *$\mathcal{S}$-derivative* of T at x. The set of all functions $T : X \to Y$ which are $\mathcal{S}$-differentiable at x is denoted by $\mathcal{D}_{\mathcal{S}}(X,Y;x)$. From conditions (ii) and (iii) it is seen that the $\mathcal{S}$-derivative $\nabla^{\mathcal{S}} T(x)$ is uniquely defined. Condition (iv) ensures that $\mathcal{S}$-differentiability in case $X = \mathbb{R}$ coincides with the usual notion of differentiability. The function $T \mapsto \nabla^{\mathcal{S}} T(x)$ is a linear mapping from $\mathcal{D}_{\mathcal{S}}(X,Y;x)$ to $\mathcal{L}(X,Y)$.

$\mathcal{S}$-differentiations may be constructed in a special way by means of coverings $\mathcal{C}$, whose elements are naturally assumed to be bounded sets C (so that $th \to 0$ uniformly for $h \in C$ as $t \to 0$). For every normed real vector space X let a covering $\mathcal{C}_X$ of X be given which consists of bounded subsets of X. If Y is another normed real vector space, define $\mathcal{S}_C(X,Y) = \{\rho : X \to Y | \lim_{t\to 0} \sup_{h\in C} \frac{\|\rho(th)\|}{t} = \rho(0) = 0 \; \forall C \in \mathcal{C}_X\}$. Then the class $\mathcal{S}_C$ satisfies the conditions (i) to (iv). With X ranging through all normed real vector spaces, we can then define the following concepts of differentiation by varying the covering $\mathcal{C}_X$. *Gâteaux*-differentiation is defined by the choices $\mathcal{C}_{GX} = \{C \subset X | C \text{ finite}\}$. For *Hadamard*-differentiation, $\mathcal{C}_{HX} = \{C \subset X | C \text{ compact}\}$ and *Fréchet*-differentiation uses the covering $\mathcal{C}_{FX} = \{C \subset X | C \text{ bounded}\}$. The three differentiations will be indicated by the corresponding authors' initials. From these definitions it is clear that ∇^F implies ∇^H which implies ∇^G. It can be shown that ∇^H is actually the weakest $\mathcal{S}$-derivative which fulfills the chain rule.

One general approach to robustness (Hampel, 1968, 1974) is the one based on influence functions which are related to Gâteaux-derivatives. Let $\mathcal{M}_1$ be the set of probability distributions on some measurable space $(Z, \mathcal{B}(Z))$ and let $\mathcal{H}$ be a reproducing kernel Hilbert space. The influence

function (IF) of $T : \mathcal{M}_1 \to \mathcal{H}$ at a point $z \in Z$ for a distribution P is defined as

$$\mathrm{IF}(z;T,\mathrm{P}) = \lim_{\varepsilon \downarrow 0} \frac{T((1-\varepsilon)\mathrm{P} + \varepsilon\delta_z) - T(\mathrm{P})}{\varepsilon}, \tag{3}$$

if the limit exists. Within this approach robust estimators are those which have a bounded influence function.[1] The influence function is neither supposed to be linear nor continuous. If the influence functions exists for all points $z \in Z$ and if it is continuous and linear, then the IF is a special Gâteaux-derivative.

Christmann and Steinwart (2004, 2007) and Steinwart and Christmann (2008b) showed that SVMs have a bounded influence function in binary classification and in regression problems provided that the kernel is bounded and continuous, L is twice Fréchet-differentiable w.r.t. the second argument, and the first and second F-derivative of L is bounded. Hence Lipschitz continuous loss functions are of special interest from a robustness point of view. An example of a loss function with these properties is the logistic loss given by $L_{log}(y,t) := -\log\big(4\Lambda(y-t)(1-\Lambda(y-t))\big)$, $y,t \in \mathbb{R}$, where $\Lambda(y-t) = 1/\big(1+e^{-(y-t)}\big)$. However the important special cases $L_{\underline{\varepsilon}}$, $L_{\tau-pin}$, and $L_{c-Huber}$ are excluded in these results, because these loss functions are not everywhere Fréchet-differentiable.

The present paper tries to fill this gap: we will propose in Definition 1 an alternative to the influence function. This alternative is based on Bouligand-derivatives whereas Hampel's influence function was defined having Gâteaux-derivatives in mind. The second goal of this paper is to use this new notion of robustness to show that SVMs for regression are robust in this sense even if the loss function has no Fréchet-derivative.

Let us now recall some facts on Bouligand-derivatives and strong approximation of functions. For the rest of the introduction let X, Y, W, and Z be normed linear spaces, and we consider neighborhoods $\mathcal{N}(x_0)$ of x_0 in X, $\mathcal{N}(y_0)$ of y_0 in Y, and $\mathcal{N}(w_0)$ of w_0 in W. Let F and G be functions from $\mathcal{N}(x_0) \times \mathcal{N}(y_0)$ to Z, h_1 and h_2 functions from $\mathcal{N}(w_0)$ to Z, f a function from $\mathcal{N}(x_0)$ to Z and g a function from $\mathcal{N}(y_0)$ to Z. A function f *approximates* F in x at (x_0,y_0), written as $f \sim_x F$ at (x_0,y_0), if $F(x,y_0) - f(x) = o(x-x_0)$. Similarly, $g \sim_y F$ at (x_0,y_0) if $F(x_0,y) - g(y) = o(y-y_0)$. A function h_1 *strongly approximates* h_2 at w_0, written as $h_1 \approx h_2$ at w_0, if for each $\varepsilon > 0$ there exists a neighborhood $\mathcal{N}(w_0)$ of w_0 such that whenever w and w' belong to $\mathcal{N}(w_0)$, $\big\|\big(h_1(w) - h_2(w)\big) - \big(h_1(w') - h_2(w')\big)\big\| \le \varepsilon \|w - w'\|$. A function f *strongly approximates* F in x at (x_0,y_0), written as $f \approx_x F$ at (x_0,y_0), if for each $\varepsilon > 0$ there exist neighborhoods $\mathcal{N}(x_0)$ of x_0 and $\mathcal{N}(y_0)$ of y_0 such that whenever x and x' belong to $\mathcal{N}(x_0)$ and y belongs to $\mathcal{N}(y_0)$ we have $\big\|\big(F(x,y) - f(x)\big) - \big(F(x',y) - f(x')\big)\big\| \le \varepsilon \|x - x'\|$. Strong approximation amounts to requiring $h_1 - h_2$ to have a strong Fréchet-derivative of 0 at w_0, though neither h_1 nor h_2 is assumed to be differentiable in any sense. A similar definition is made for strong approximation in y. We define strong approximation for functions of several groups of variables, for example $G \approx_{(x,y)} F$ at (x_0,y_0), by replacing W by $X \times Y$ and making the obvious substitutions. Note that one has both $f \approx_x F$ and $g \approx_y F$ at (x_0,y_0) exactly if $f(x) + g(y) \approx_{(x,y)} F$ at (x_0,y_0).

Recall that a function $f : X \to Z$ is called *positive homogeneous* if

$$f(\alpha x) = \alpha f(x) \quad \forall \alpha \ge 0, \forall x \in X.$$

Following Robinson (1987) we can now define the *Bouligand-derivative*. Given a function f from an open subset $\mathcal{X}$ of a normed linear space X into another normed linear space Z, we say that

1. In the following we use the term "robust" in this sense, unless otherwise stated.

f is *Bouligand-differentiable* at a point $x_0 \in \mathcal{X}$, if there exists a positive homogeneous function $\nabla^B f(x_0) : \mathcal{X} \to Z$ such that

$$f(x_0+h) = f(x_0) + \nabla^B f(x_0)(h) + o(h). \tag{4}$$

We can write (4) also as

$$\lim_{h\to 0} \left\| f(x_0+h) - f(x_0) - \nabla^B f(x_0)(h) \right\|_Z / \|h\|_X = 0. \tag{5}$$

Let $F : X \times Y \to Z$, and suppose that F has a partial B-derivative[2] $\nabla_1^B F(x_0,y_0)$ with respect to x at (x_0,y_0). We say $\nabla_1^B F(x_0,y_0)$ is *strong* if $F(x_0,y_0) + \nabla_1^B F(x_0,y_0)(x-x_0) \approx_x F$ at (x_0,y_0). Robinson (1987) showed that the chain rule holds for Bouligand-derivatives. Let f be a Lipschitzian function from an open set $\Omega \subset \mathbb{R}^m$ to $\mathbb{R}^k$, $x_0 \in \Omega$, and f B-differentiable at x_0. Let g be a Lipschitzian function from an open set $\Gamma \subset \mathbb{R}^k$, with $f(x_0) \in \Gamma$, to $\mathbb{R}^l$ be B-differentiable at $f(x_0)$. Then $g \circ f$ is B-differentiable at x_0 and $\nabla^B(g \circ f)(x_0) = \nabla^B g\big(f(x_0)\big) \circ \nabla^B f(x_0)$. The fact that B-derivatives, just as F- and H-derivatives, fulfill the chain rule is no contradiction to the before mentioned fact that H-differentiability is the *weakest $\mathcal{S}$-differentiation* which fulfills the chain rule (Rieder, 1994, p. 4) because the B-derivative is not necessarily a continuous linear function.

In general Gâteaux- and Bouligand-differentiability are not directly comparable, because B-derivatives are by definition positive homogeneous, but not necessarily linear. We will show that the existence of the BIF implies the existence of the IF and that in that case BIF=IF. Please note that this in general does not imply that the IF is a Gâteaux-derivative.

In this paper, we will prove that many SVMs based on Lipschitz continuous loss functions have a bounded Bouligand influence function. To formulate our results we will use Bouligand-derivatives in the sense of Robinson (1991) as defined above. These directional derivatives were to our best knowledge not used in robust statistics so far, but are successfully applied in approximation theory for non-smooth functions. Section 2 covers our definition of the Bouligand influence function (BIF) and contains the main result which gives the BIF for support vector machines based on a bounded kernel and a B-differentiable Lipschitz continuous convex loss function. In Section 3 it is shown that this result covers the loss functions $L_{\underline{\varepsilon}}$, $L_{\tau-pin}$, $L_{c-Huber}$, and L_{log} as special cases. Section 4 contains the conclusions. All proofs are given in the Appendix.

2. Main Result

This section contains our two main results: the definition of the Bouligand influence function and a theorem which shows that a broad class of support vector machines based on a Lipschitz continuous, but not necessarily Fréchet-differentiable loss function have a bounded Bouligand influence function. We denote the set of all probability distributions on some measurable space $(Z, \mathcal{B}(Z))$ by $\mathcal{M}_1$ and let $\mathcal{H}$ be a Hilbert space.

Definition 1 *The **Bouligand influence function (BIF)** of the function $T : \mathcal{M}_1 \to \mathcal{H}$ for a distribution* P *in the direction of a distribution* $\mathrm{Q} \neq \mathrm{P}$ *is the special Bouligand-derivative (if it exists)*

$$\lim_{\varepsilon \downarrow 0} \frac{\left\| T\big((1-\varepsilon)\mathrm{P} + \varepsilon \mathrm{Q}\big) - T(\mathrm{P}) - \mathrm{BIF}(\mathrm{Q}; T, \mathrm{P}) \right\|_{\mathcal{H}}}{\varepsilon} = 0. \tag{6}$$

2. Throughout the paper we will denote partial B-derivatives of f by $\nabla_1^B f$, $\nabla_2^B f$, $\nabla_{2,2}^B f := \nabla_2^B(\nabla_2^B f)$ etc.

The BIF has the interpretation that it measures the impact of an infinitesimal small amount of contamination of the original distribution P in the direction of Q on the quantity of interest $T(\mathrm{P})$. It is thus desirable that the function T has a *bounded* BIF.

Note that (6) is indeed a special B-derivative, because we consider the directions $h = \varepsilon(\mathrm{Q}-\mathrm{P})$ and $x_0 = \mathrm{P}$. If Q equals the Dirac distribution δ_z in a point $z \in Z$, that is $\delta_z(\{z\}) = 1$, we write $\mathrm{BIF}(z;T,\mathrm{P})$. The choice of the metric on $\mathcal{M}_1$ is not important for the definition of the BIF, because $\|\varepsilon(\mathrm{Q}-\mathrm{P})\| = \varepsilon\,\|\mathrm{Q}-\mathrm{P}\|$ and $\|\mathrm{Q}-\mathrm{P}\|$ is a positive constant. For the norm of total variation we obtain for example,

$$\lim_{\varepsilon(\mathrm{Q}-\mathrm{P})\downarrow 0} \frac{\big\|T\big(\mathrm{P}+\varepsilon(\mathrm{Q}-\mathrm{P})\big) - T(\mathrm{P}) - \mathrm{BIF}(\mathrm{Q};T,\mathrm{P})\big\|_{\mathcal{H}}}{\|\varepsilon(\mathrm{Q}-\mathrm{P})\|_{\mathrm{tv}}} = 0,$$

(cf., Equation 5). Since $\varepsilon(\mathrm{Q}-\mathrm{P}) \to 0$ iff $\varepsilon \to 0$ and by assumption $\mathrm{Q} \neq \mathrm{P}$ we obtain (6).

The Bouligand influence function is a modification of the influence function given by (3). Recall that the Gâteaux-derivative of some mapping f at a point x_0 equals $\nabla^G f(x_0)(h) = \lim_{\varepsilon\downarrow 0}\big(f(x_0+\varepsilon h) - f(x_0)\big)/\varepsilon$ if it exists for every $h \in X$. Hence the influence function is the special Gâteaux-derivative with $\mathrm{Q} = \delta_z$ and $h = \delta_z - \mathrm{P}$, if the IF is continuous and linear. However, the BIF is always positive homogeneous because it is a Bouligand-derivative, which is in general not true for the influence function. As will be shown in (13), this property leads to the result that for $\alpha \geq 0$ and $h := \varepsilon(\mathrm{Q}-\mathrm{P})$ the asymptotic bias $T((1-\alpha\varepsilon)\mathrm{P}+\alpha\varepsilon\mathrm{Q}) - T(\mathrm{P})$ equals $\alpha\,\mathrm{BIF}(\mathrm{Q};T,\mathrm{P}) + o(h)$.

The following simple calculations clarify the connection between the BIF and the IF. In general we have for B-derivatives with $h = \varepsilon\tilde{h}$, where $\varepsilon \in (0,\infty)$ and $\tilde{h} \in X$ with $0 < \|\tilde{h}\| \leq 2$,

$$\begin{aligned}
0 &= \lim_{h\to 0} \frac{\|f(x_0+h) - f(x_0) - \nabla^B f(x_0)(h)\|}{\|h\|} \\
&= \lim_{\varepsilon\downarrow 0} \frac{\|f(x_0+\varepsilon\tilde{h}) - f(x_0) - \varepsilon\nabla^B f(x_0)(\tilde{h})\|}{\varepsilon\|\tilde{h}\|} \\
&= \lim_{\varepsilon\downarrow 0} \Big\| \frac{f(x_0+\varepsilon\tilde{h}) - f(x_0)}{\varepsilon} - \nabla^B f(x_0)(\tilde{h}) \Big\|.
\end{aligned}$$

Hence $\lim_{\varepsilon\downarrow 0}\big(f(x_0+\varepsilon\tilde{h}) - f(x_0)\big)/\varepsilon = \nabla^B f(x_0)(\tilde{h})$. In particular we obtain for $\mathrm{Q} \neq \mathrm{P}$ and taking $0 < \|\mathrm{Q}-\mathrm{P}\| \leq 2$ into account that, if $\mathrm{BIF}(\mathrm{Q};T,\mathrm{P})$ exists, then $\mathrm{BIF}(\mathrm{Q};T,\mathrm{P}) = \lim_{\varepsilon\downarrow 0}\big(T((1-\varepsilon)\mathrm{P}+\varepsilon\mathrm{Q}) - T(P)\big)/\varepsilon$, which is the definition of the IF, if we choose $\mathrm{Q} = \delta_z$.

We can now give a general result on the BIF of the support vector machine $T(\mathrm{P}) := f_{\mathrm{P},\lambda}$. We restrict attention to Lipschitz continuous loss functions, because the growth behavior of L plays an important role to obtain consistency and robustness results as was shown by Christmann and Steinwart (2007). For notational convenience we shall often write $\nabla_2^B L(Y, f(X))$ instead of $\nabla_2^B L(Y,\cdot)(f(X))$, because $f(X) \in \mathbb{R}$. We will sometimes explicitly write "$\cdot$" for multiplication to avoid misunderstandings.

Theorem 2 *Let $X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}$ be closed sets, $\mathcal{H}$ be a RKHS with a bounded, continuous kernel k, $f_{\mathrm{P},\lambda} \in \mathcal{H}$, and $L : Y \times \mathbb{R} \to [0,\infty)$ be a convex loss function which is Lipschitz continuous w.r.t. the second argument with uniform Lipschitz constant $|L|_1 := \sup_{y\in Y} |L(y,\cdot)|_1 \in (0,\infty)$. Further, assume that L has measurable partial B-derivatives w.r.t. to the second argument with*

$$\kappa_1 := \sup_{y\in Y} \big\|\nabla_2^B L(y,\cdot)\big\|_\infty \in (0,\infty)\ ,\ \kappa_2 := \sup_{y\in Y} \big\|\nabla_{2,2}^B L(y,\cdot)\big\|_\infty < \infty. \tag{7}$$

Let $\delta_1 > 0$, $\delta_2 > 0$, $\mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda}) := \{f \in \mathcal{H}; \|f - f_{\mathrm{P},\lambda}\|_{\mathcal{H}} < \delta_1\}$, $\lambda > \frac{1}{2}\kappa_2 \|k\|_\infty^3$, *and* P, Q *be probability measures*[3] *on* $\big(X \times Y, \mathcal{B}(X \times Y)\big)$ *with* $\mathbb{E}_\mathrm{P}|Y| < \infty$ *and* $\mathbb{E}_\mathrm{Q}|Y| < \infty$. *Define* $G : (-\delta_2, \delta_2) \times \mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda}) \to \mathcal{H}$,

$$G(\varepsilon, f) := 2\lambda f + \mathbb{E}_{(1-\varepsilon)\mathrm{P}+\varepsilon\mathrm{Q}} \nabla_2^B L(Y, f(X)) \cdot \Phi(X), \tag{8}$$

and assume that $\nabla_2^B G(0, f_{\mathrm{P},\lambda})$ *is strong. Then the Bouligand influence function of* $T(\mathrm{P}) := f_{\mathrm{P},\lambda}$ *in the direction of* $\mathrm{Q} \neq \mathrm{P}$ *exists,*

$$\begin{aligned} \mathrm{BIF}(\mathrm{Q}; T, \mathrm{P}) &= S^{-1}\big(\mathbb{E}_\mathrm{P} \nabla_2^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot \Phi(X)\big) && (9) \\ &\quad - S^{-1}\big(\mathbb{E}_\mathrm{Q} \nabla_2^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot \Phi(X)\big), && (10) \end{aligned}$$

where $S : \mathcal{H} \to \mathcal{H}$ *with*

$$S(\cdot) := \nabla_2^B G(0, f_{\mathrm{P},\lambda})(\cdot) = 2\lambda \,\mathrm{id}_{\mathcal{H}}(\cdot) + \mathbb{E}_\mathrm{P} \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot \langle \Phi(X), \cdot \rangle_{\mathcal{H}} \Phi(X),$$

and $\mathrm{BIF}(\mathrm{Q}; T, \mathrm{P})$ *is bounded.*

Remark 3 *We additionally show that under the assumptions of Theorem 2 we have:*

1. *For some* χ *and each* $f \in \mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda})$, $G(\cdot, f)$ *is Lipschitz continuous on* $(-\delta_2, \delta_2)$ *with Lipschitz constant* χ.

2. G *has partial B-derivatives with respect to* ε *and* f *at* $(0, f_{\mathrm{P},\lambda})$.

3. $\nabla_2^B G(0, f_{\mathrm{P},\lambda})(h - f_{\mathrm{P},\lambda})$ *lies in a neighborhood of* $0 \in \mathcal{H}$, $\forall h \in \mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda})$.

4. $d_0 := \inf_{h_1, h_2 \in \mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda}) - f_{\mathrm{P},\lambda}; h_1 \neq h_2} \frac{\|\nabla_2^B G(0, f_{\mathrm{P},\lambda})(h_1) - \nabla_2^B G(0, f_{\mathrm{P},\lambda})(h_2)\|_{\mathcal{H}}}{\|h_1 - h_2\|_{\mathcal{H}}} > 0$.

5. *For each* $\xi > d_0^{-1}\chi$ *there exist constants* $\delta_3, \delta_4 > 0$, *a neighborhood* $\mathcal{N}_{\delta_3}(f_{\mathrm{P},\lambda}) := \{f \in \mathcal{H}; \|f - f_{\mathrm{P},\lambda}\|_{\mathcal{H}} < \delta_3\}$, *and a function* $f^* : (-\delta_4, \delta_4) \to \mathcal{N}_{\delta_3}(f_{\mathrm{P},\lambda})$ *satisfying*

   v.1) $f^*(0) = f_{\mathrm{P},\lambda}$.

   v.2) f^* *is Lipschitz continuous on* $(-\delta_4, \delta_4)$ *with Lipschitz constant* $|f^*|_1 = \xi$.

   v.3) For each $\varepsilon \in (-\delta_4, \delta_4)$ *is* $f^*(\varepsilon)$ *the unique solution of* $G(\varepsilon, f) = 0$ *in* $\mathcal{N}_{\delta_3}(f_{\mathrm{P},\lambda})$.

   v.4) $\nabla^B f^*(0)(u) = \big(\nabla_2^B G(0, f_{\mathrm{P},\lambda})\big)^{-1}\big(-\nabla_1^B G(0, f_{\mathrm{P},\lambda})(u)\big)$, $u \in (-\delta_4, \delta_4)$.

   The function f^* *is the same as in the implicit function theorem by Robinson (1991), see Theorem 7.*

Remark 4 *It will be shown that* $\kappa_2 = 0$ *for* $L = L_{\underline{\varepsilon}}$ *and* $L = L_{\tau-pin}$ *and thus the regularization condition only states that* $\lambda > \frac{1}{2}\kappa_2 \|k\|_\infty^3 = 0$.

3. Because X and Y are assumed to be closed, P can be split up into the marginal distribution P_X and the regular conditional probability $\mathrm{P}(\cdot|x)$, $x \in X$, on Y. Same for Q.

Note that S can be interpreted as the (Bouligand-)Hessian of the regularized risk, see (14) and (17). Further the formula in (9) and (10) is similar to the one obtained by Christmann and Steinwart (2007) for the IF of $T(\mathrm{P}) = f_{\mathrm{P},\lambda}$. The difference is that we used B-derivatives instead of F-derivatives, because we allow non-smooth L.

Note that the first summand of the BIF given in (9) does *not* depend on the contaminating distribution Q. In contrast to that, the second summand of the BIF given in (10) depends on Q and consists of two factors. The first factor depends on the partial B-derivative of the loss function, and is hence bounded due to (7). For many loss functions this factor depends only on the residual term $y - f_{\mathrm{P},\lambda}(x)$. The second factor is the feature map $\Phi(x)$ which is bounded, because k is bounded. For the Gaussian RBF kernel we expect that the second factor is not only bounded, but that the impact of $\mathrm{Q} \neq \mathrm{P}$ on the BIF is approximately local, because $k(x,x')$ converges exponentially fast to zero if $||x-x'||_2$ is large.

3. Examples

In this section we show that our main theorem covers some SVMs widely used in practice. The following result treats SVMs based on the ε-insensitive loss function or Huber's loss function for regression, and SVMs based on the pinball loss function for nonparametric quantile regression. These loss functions have uniformly bounded first and second partial B-derivatives w.r.t. the second argument, see the Appendix.

Corollary 5 *Let $X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}$ be closed, and P, Q be distributions on $X \times Y$ with $\mathbb{E}_{\mathrm{P}}|Y| < \infty$ and $\mathbb{E}_{\mathrm{Q}}|Y| < \infty$.*

1. *For $L \in \{L_{\tau-pin}, L_{\varepsilon}\}$, assume that for all $\delta > 0$ there exist positive constants ξ_{P}, ξ_{Q}, c_{P}, and c_{Q} such that for all $t \in \mathbb{R}$ with $|t - f_{\mathrm{P},\lambda}(x)| \leq \delta \|k\|_\infty$ the following inequalities hold for all $a \in [0, 2\delta\|k\|_\infty]$ and $x \in X$:*

$$\mathrm{P}\big(Y \in [t, t+a] \,\big|\, x\big) \leq c_{\mathrm{P}} a^{1+\xi_{\mathrm{P}}} \text{ and } \mathrm{Q}\big(Y \in [t, t+a] \,\big|\, x\big) \leq c_{\mathrm{Q}} a^{1+\xi_{\mathrm{Q}}}. \tag{11}$$

2. *For $L = L_{c-Huber}$, assume for $x \in X$:*

$$\mathrm{P}\big(Y \in \big\{ f_{\mathrm{P},\lambda}(x) - c, f_{\mathrm{P},\lambda}(x) + c \big\} \,\big|\, x\big) = \mathrm{Q}\big(Y \in \big\{ f_{\mathrm{P},\lambda}(x) - c, f_{\mathrm{P},\lambda}(x) + c \big\} \,\big|\, x\big) = 0. \tag{12}$$

Then the assumptions of Theorem 2 are valid: $\mathrm{BIF}(\mathrm{Q}; T, \mathrm{P})$ *of $T(\mathrm{P}) := f_{\mathrm{P},\lambda}$ exists, is given by (9) to (10), and is bounded.*

For the somewhat smoother Huber loss function we only need to exclude by (12) that the conditional probabilities of Y given X with respect to P and Q have no point probabilities at the two points $f_{\mathrm{P},\lambda}(x) - c$ and $f_{\mathrm{P},\lambda}(x) + c$. Therefore, for this loss function Q can be a Dirac distribution and in this case we have BIF = IF.

For the pinball loss function some calculations give

$$\begin{aligned} \mathrm{BIF}(\mathrm{Q}; T, \mathrm{P}) \;&=\; \frac{1}{2\lambda} \int_X \big(\mathrm{P}\big(Y \leq f_{\mathrm{P},\lambda}(x) \,\big|\, x\big) - \tau\big) \Phi(x) \, d\mathrm{P}_X(x) \\ &\quad - \frac{1}{2\lambda} \int_X \big(\mathrm{Q}\big(Y \leq f_{\mathrm{P},\lambda}(x) \,\big|\, x\big) - \tau\big) \Phi(x) \, d\mathrm{Q}_X(x), \end{aligned}$$

if the BIF exists. We expect the first integral to be small, because $f_{\mathrm{P},\lambda}(x)$ approximates the τ-quantile of $\mathrm{P}(\cdot\,|x)$ and even rates of convergence are known (Steinwart and Christmann, 2008a,b). As will become clear from the proof, (11) and (12) guarantee that the regular conditional probabilities $\mathrm{P}(\cdot\,|x)$ and $\mathrm{Q}(\cdot\,|x)$ do not have large point masses at those points where the Lipschitz continuous loss function L is *not* F-differentiable or in small neighborhoods around these points. Even for the case of parametric quantile regression, that is for $L = L_{\tau-pin}$, $\lambda = 0$ and the unbounded linear kernel $k(x,x') := \langle x,x'\rangle$, some assumptions on the distribution P seem to be necessary for the existence of the IF, see Koenker (2005, p. 44). He assumes that P has a continuous density which is strictly positive where needed.

Nevertheless, the question arises whether Theorem 2 and Corollary 5 can be shown without any assumption on the distributions P and Q. This is—at least with the techniques we used—not possible for non-smooth loss functions as the following counterexample shows. Let us consider kernel based quantile regression based on the Gaussian RBF kernel, that is $L = L_{\tau-pin}$, $k = k_{RBF}$, and $\lambda > 0$. Hence the set $\mathfrak{D}$ of discontinuity points of $\nabla_2^B L$ is $\mathfrak{D} = \{0\}$. Fix $x \in X$ and $y, y^* \in Y$ with $y \neq y^*$. Define $\mathrm{P} = \delta_{(x,y)}$ and $\mathrm{Q} = \delta_{(x,y^*)}$. Consider $f_1, f_2 \in \mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda})$ with $f_1(x) \neq f_2(x)$, $y - f_1(x) > 0$, $y - f_2(x) < 0$, $y^* - f_1(x) > 0$, and $y^* - f_2(x) < 0$. Hence, $\nabla_2^B L(y, f_1(x)) = \nabla_2^B L(y^*, f_1(x)) = -\tau$ and $\nabla_2^B L(y, f_2(x)) = \nabla_2^B L(y^*, f_2(x)) = 1-\tau$. Note that $\nabla_{2,2}^B L(y,t) = 0$ for all $y,t \in \mathbb{R}$. We thus obtain for the $\mathcal{H}$-norm in (19) that $\big\|\mathbb{E}_{(1-\varepsilon)\mathrm{P}+\varepsilon\mathrm{Q}}\big(\nabla_2^B L(Y, f_1(X)) - \nabla_2^B L(Y, f_2(X))\big)\cdot\Phi(X)\big\|_{\mathcal{H}} = \|\Phi(x)\|_{\mathcal{H}} > 0$. Hence $\nabla_2^B G(0, f_{\mathrm{P},\lambda})$ is not strong in this special case, because $\|\Phi(x)\|_{\mathcal{H}}$ is in general greater than $\varepsilon^*\,\|f_1 - f_2\|_{\mathcal{H}}$ for arbitrarily small values of ε^*.

Now we shall show for L_{log} that the assumptions (11) or (12) are not needed to obtain a bounded BIF. It is easy to see that L_{log} is strictly convex w.r.t. the second argument and Fréchet-differentiable with $\nabla_2^F L_{log}(y,t) = 1 - 2\Lambda(y-t)$, $\nabla_{2,2}^F L_{log}(y,t) = 2\Lambda(y-t)[1-\Lambda(y-t)]$, and $\nabla_{2,2,2}^F L_{log}(y,t) = -2\Lambda(y-t)[1-\Lambda(y-t)][1-2\Lambda(y-t)]$. Obviously, these partial derivatives are bounded for all $y,t \in \mathbb{R}$. Furthermore, $\kappa_1 = \sup_{y\in\mathbb{R}} |\nabla_2^F L_{log}(y,\cdot)|_1 = 1/2$ and $\kappa_2 = \sup_{y\in\mathbb{R}} |\nabla_{2,2}^F L_{log}(y,\cdot)|_1 \leq 1/2$, because an everywhere F-differentiable function g is Lipschitz continuous with $|g|_1 = ||\nabla^F g||_\infty$ if $\nabla^F g$ is bounded.

Corollary 6 *Let $X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}$ be closed, $L = L_{log}$, and P, Q be distributions on $X \times Y$ with $\mathbb{E}_{\mathrm{P}}|Y| < \infty$ and $\mathbb{E}_{\mathrm{Q}}|Y| < \infty$. Then the assumptions of Theorem 2 are valid, and $\mathrm{BIF}(\mathrm{Q}; T, \mathrm{P})$ of $T(\mathrm{P}) := f_{\mathrm{P},\lambda}$ exists, is given by (9) to (10), and $\mathrm{BIF}(\mathrm{Q}; T, \mathrm{P})$ is bounded.*

Corollary 6 is of course also valid for empirical distributions D_n and Q_m consisting of n and m data points, because no specific assumptions on P and Q are made.

The influence function of $T(\mathrm{P}) = f_{\mathrm{P},\lambda}$ based on L_{log} and error bounds of the type

$$\big\|T\big((1-\varepsilon)\mathrm{P} + \varepsilon\delta_{(x,y)} - T(\mathrm{P})\big)\big\|_{\mathcal{H}} \leq c^*\,\varepsilon$$

where the constant c^* is known and depends only on P, $\mathrm{Q} := \delta_{(x,y)}$, and λ, were recently derived by Christmann and Steinwart (2007). We like to mention that Corollary 6 shows that this influence function is even a Bouligand-derivative, hence *positive homogeneous* in $h = \varepsilon(\mathrm{Q} - \mathrm{P})$. Therefore, we immediately obtain from the existence of the BIF that the asymptotic bias of SVMs has the form

$$\begin{aligned} f_{(1-\alpha\varepsilon)\mathrm{P}+\alpha\varepsilon\mathrm{Q},\lambda} - f_{\mathrm{P},\lambda} &= T(\mathrm{P} + \alpha h) - T(\mathrm{P}) \\ &= \alpha\,\mathrm{BIF}(\mathrm{Q}; T, \mathrm{P}) + o(\alpha h) \\ &= \alpha\big(T(\mathrm{P}+h) - T(\mathrm{P}) + o(h)\big) + o(\alpha h) \\ &= \alpha\big(f_{(1-\varepsilon)\mathrm{P}+\varepsilon\mathrm{Q},\lambda} - f_{\mathrm{P},\lambda}\big) + o(\alpha\varepsilon(\mathrm{Q}-\mathrm{P})), \quad \alpha \geq 0. \end{aligned} \tag{13}$$

This equation nicely describes the behavior of the asymptotic bias term $f_{(1-\varepsilon)P+\varepsilon Q,\lambda} - f_{P,\lambda}$ if we consider the amount $\alpha\varepsilon$ of contamination instead of ε.

4. Discussion

Bouligand-derivatives and strong Bouligand-derivatives were successfully used in approximation theory, see for example Clarke (1983), Robinson (1987, 1991), Ip and Kyparisis (1992), and the references cited therein. To our best knowledge however, these concepts were not used so far to investigate robustness properties of statistical operators.

Therefore, we defined the Bouligand influence function (BIF) as a modification of the influence function (IF), the latter being related to Gâteaux-derivatives and a cornerstone of robust statistics, see Hampel (1974), Hampel et al. (1986), and Maronna et al. (2006). If the BIF exists, then it is identical to the IF. The BIF is a positive homogeneous function by definition. This is in general not true for the IF. We used the BIF to show that support vector machines for regression, which play an important role in modern statistical learning theory, are robust in the sense of influence functions, if a bounded continuous kernel is used and if the convex loss function is Lipschitz continuous and twice Bouligand-differentiable, but not necessarily twice Fréchet-differentiable. The result covers the important special cases of SVMs based on the ε-insensitive, Huber or logistic loss function for regression, and kernel based quantile regression based on the pinball loss function. The IF of SVMs based on the logistic loss was recently derived by Christmann and Steinwart (2007) and Steinwart and Christmann (2008b).

From our point of view, the Bouligand-derivative is a promising concept for robust statistics for the following reason. Many robust estimators proposed in the literature are implicitly defined as solutions of minimization problems where the objective function or loss function is continuous or Lipschitz continuous, but not necessarily twice Fréchet-differentiable. Examples are not only SVMs treated in this paper, but also M-estimators of Huber-type and certain maximum likelihood estimators under non-standard conditions. Bouligand-differentiation nicely fills the gap between Fréchet-differentiation, which is too strong for many robust estimators, and Gâteaux-differentiation which is the basis for the robustness approach based on influence functions. Bouligand-derivatives fulfill a chain rule and a theorem of implicit functions which is in general not true for Gâteaux-derivatives.

Acknowledgments

We thank S.M. Robinson for drawing our attention to his paper from 1987. We also thank the editor and three anonymous referees.

Appendix A. Proofs

This appendix contains all the proofs of the previous sections.

A.1 Proofs for the Results in Section 2

For the proof of Theorem 2 we shall use the following implicit function theorem for B-derivatives, see Robinson (1991, Cor. 3.4). For a function f from a metric space (X, d_X) to another metric space

(Y, d_Y), we define

$$\delta(f, X) = \inf\{d_Y\big(f(x_1), f(x_2)\big) / d_X(x_1, x_2) \mid x_1 \neq x_2; x_1, x_2 \in X\}.$$

Theorem 7 *Let Y be a Banach space and X and Z be normed linear spaces. Let x_0 and y_0 be points of X and Y, respectively, and let $\mathcal{N}(x_0)$ be a neighborhood of x_0 and $\mathcal{N}(y_0)$ be a neighborhood of y_0. Suppose that G is a function from $\mathcal{N}(x_0) \times \mathcal{N}(y_0)$ to Z with $G(x_0, y_0) = 0$. In particular, for some ϕ and each $y \in \mathcal{N}(y_0)$, $G(\cdot, y)$ is assumed to be Lipschitz continuous on $\mathcal{N}(x_0)$ with modulus ϕ. Assume that G has partial B-derivatives with respect to x and y at (x_0, y_0), and that: (i) $\nabla_2^B G(x_0, y_0)(\cdot)$ is strong. (ii) $\nabla_2^B G(x_0, y_0)(y - y_0)$ lies in a neighborhood of $0 \in Z$, $\forall y \in \mathcal{N}(y_0)$. (iii) $\delta(\nabla_2^B G(x_0, y_0), \mathcal{N}(y_0) - y_0) =: d_0 > 0$. Then for each $\xi > d_0^{-1}\phi$ there are neighborhoods U of x_0 and V of y_0, and a function $f^* : U \to V$ satisfying (a) $f^*(x_0) = y_0$. (b) f^* is Lipschitz continuous on $\mathcal{N}(x_0)$ with modulus ξ. (c) For each $x \in U$, $f^*(x)$ is the unique solution in V of $G(x, y) = 0$. (d) The function f^* is B-differentiable at x_0 with $\nabla^B f^*(x_0)(u) = \big(\nabla_2^B G(x_0, y_0)\big)^{-1}\big(-\nabla_1^B G(x_0, y_0)(u)\big)$.*

We will also need the following consequence of the open mapping theorem, see Lax (2002, p. 170).

Theorem 8 *Let X and Y be Banach spaces, $A : X \to Y$ be a bounded, linear, and bijective function. Then the inverse $A^{-1} : Y \to X$ is a bounded linear function.*

The key ingredient of our proof of Theorem 2 is of course the map $G : \mathbb{R} \times \mathcal{H} \to \mathcal{H}$ defined by (8). If $\varepsilon < 0$ the integration is w.r.t. a signed measure. The $\mathcal{H}$-valued expectation used in the definition of G is well-defined for all $\varepsilon \in (\delta_2, \delta_2)$ and all $f \in \mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda})$, because $\kappa_1 \in (0, \infty)$ by (7) and $\|\Phi(x)\|_\infty \leq \|k\|_\infty^2 < \infty$ by (2). For F- and B-derivatives holds a chain rule and F-differentiable functions are also B-differentiable. For $\varepsilon \in [0, 1]$ we thus obtain

$$G(\varepsilon, f) = \frac{\partial \mathcal{R}^{reg}_{L,(1-\varepsilon)\mathrm{P}+\varepsilon\mathrm{Q},\lambda}}{\partial \mathcal{H}}(f) = \nabla_2^B \mathcal{R}^{reg}_{L,(1-\varepsilon)\mathrm{P}+\varepsilon\mathrm{Q},\lambda}(f). \tag{14}$$

Since $f \mapsto \mathcal{R}^{reg}_{L,(1-\varepsilon)\mathrm{P}+\varepsilon\mathrm{Q},\lambda}(f)$ is convex and continuous for all $\varepsilon \in [0, 1]$ equation (14) shows that we have $G(\varepsilon, f) = 0$ if and only if $f = f_{(1-\varepsilon)\mathrm{P}+\varepsilon\mathrm{Q},\lambda}$ for such ε. Hence

$$G(0, f_{\mathrm{P},\lambda}) = 0. \tag{15}$$

We shall show that Theorem 7 is applicable for G and that there exists a B-differentiable function $\varepsilon \mapsto f_\varepsilon$ defined on a small interval $(-\delta_2, \delta_2)$ for some $\delta_2 > 0$ satisfying $G(\varepsilon, f_\varepsilon) = 0$ for all $\varepsilon \in (-\delta_2, \delta_2)$. From the existence of this function we shall obtain $\mathrm{BIF}(\mathrm{Q}; T, \mathrm{P}) = \nabla^B f_\varepsilon(0)$.

Proof of Theorem 2. The existence of $f_{\mathrm{P},\lambda}$ follows from the convexity of L and the penalizing term, see also Christmann and Steinwart (2007, Prop. 8). The assumption that $G(0, f_{\mathrm{P},\lambda}) = 0$ is valid by (15). Let us now prove the results of Remark 3 parts 1 to 5.

Remark 3 part 1. For $f \in \mathcal{H}$ fixed let $\varepsilon_1, \varepsilon_2 \in (-\delta_2, \delta_2)$. Using $\|k\|_\infty < \infty$ and (15) we obtain

$$\begin{aligned}
& \left|\mathbb{E}_{(1-\varepsilon_1)\mathrm{P}+\varepsilon_1\mathrm{Q}} \nabla_2^B L(Y, f(X)) \cdot \Phi(X) - \mathbb{E}_{(1-\varepsilon_2)\mathrm{P}+\varepsilon_2\mathrm{Q}} \nabla_2^B L(Y, f(X)) \cdot \Phi(X)\right| \\
= & \left|(\varepsilon_1 - \varepsilon_2) \mathbb{E}_{\mathrm{Q}-\mathrm{P}} \nabla_2^B L(Y, f(X)) \cdot \Phi(X)\right| \\
\leq & |\varepsilon_1 - \varepsilon_2| \int \left|\nabla_2^B L(y, f(x)) \cdot \Phi(x)\right| d|\mathrm{Q}-\mathrm{P}|(x,y) \\
\leq & |\varepsilon_1 - \varepsilon_2| \int \sup_{y \in Y} |\nabla_2^B L(y, f(x))| \sup_{x \in X} |\Phi(x)| \, d|\mathrm{Q}-\mathrm{P}|(x,y) \\
\leq & |\varepsilon_1 - \varepsilon_2| \, \|\Phi(x)\|_\infty \sup_{y \in Y} \left\|\nabla_2^B L(y, \cdot)\right\|_\infty \int d|\mathrm{Q}-\mathrm{P}|(x,y) \\
\leq & 2 \, \|k\|_\infty^2 \sup_{y \in Y} \left\|\nabla_2^B L(y, \cdot)\right\|_\infty |\varepsilon_1 - \varepsilon_2| \\
= & 2 \, \|k\|_\infty^2 \, \kappa_1 \, |\varepsilon_1 - \varepsilon_2| < \infty .
\end{aligned}$$

Remark 3 part 2. We have

$$\begin{aligned}
\nabla_1^B G(\varepsilon, f) &= \nabla_1^B \Big(\mathbb{E}_{(1-\varepsilon)\mathrm{P}+\varepsilon\mathrm{Q}} \nabla_2^B L(Y, f(X)) \cdot \Phi(X) \Big) \\
&= \nabla_1^B \Big(\mathbb{E}_{\mathrm{P}} \nabla_2^B L(Y, f(X)) \cdot \Phi(X) + \varepsilon \mathbb{E}_{\mathrm{Q}-\mathrm{P}} \nabla_2^B L(Y, f(X)) \cdot \Phi(X) \Big) \\
&= \mathbb{E}_{\mathrm{Q}-\mathrm{P}} \nabla_2^B L(Y, f(X)) \cdot \Phi(X) \\
&= \mathbb{E}_{\mathrm{Q}} \nabla_2^B L(Y, f(X)) \cdot \Phi(X) - \mathbb{E}_{\mathrm{P}} \nabla_2^B L(Y, f(X)) \cdot \Phi(X) . \qquad (16)
\end{aligned}$$

This expectation exists due to (2) and (7). Furthermore, we obtain

$$\begin{aligned}
& \nabla_2^B G(0, f_{\mathrm{P},\lambda})(h) + o(h) \\
= & G(0, f_{\mathrm{P},\lambda} + h) - G(0, f_{\mathrm{P},\lambda}) \\
= & 2\lambda h + \mathbb{E}_{\mathrm{P}} \nabla_2^B L(Y, (f_{\mathrm{P},\lambda}(X) + h(X))) \cdot \Phi(X) - \mathbb{E}_{\mathrm{P}} \nabla_2^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot \Phi(X) \\
= & 2\lambda h + \mathbb{E}_{\mathrm{P}} \Big(\nabla_2^B L\big(Y, (f_{\mathrm{P},\lambda}(X) + h(X))\big) - \nabla_2^B L\big(Y, f_{\mathrm{P},\lambda}(X)\big) \Big) \cdot \Phi(X) .
\end{aligned}$$

This expectation exists, as the term $\nabla_2^B L\big(Y, (f_{\mathrm{P},\lambda}(X) + h(X))\big) - \nabla_2^B L\big(Y, f_{\mathrm{P},\lambda}(X)\big)$ is bounded due to (2), (7), and $\|k\|_\infty < \infty$. Using $\langle \Phi(X), \cdot \rangle_{\mathcal{H}} \in \mathcal{H}$, we get

$$\nabla_2^B G(0, f_{\mathrm{P},\lambda})(\cdot) = 2\lambda \mathrm{id}_{\mathcal{H}}(\cdot) + \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot \langle \Phi(X), \cdot \rangle_{\mathcal{H}} \Phi(X) . \qquad (17)$$

Note that $\mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L\big(Y, f(X)\big) = \nabla_2^B \mathbb{E}_{\mathrm{P}} \nabla_2^B L\big(Y, f(X)\big)$, because

$$\begin{aligned}
& \nabla_2^B \mathbb{E}_{\mathrm{P}} \nabla_2^B L\big(Y, f(X)\big) - \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L\big(Y, f(X)\big) \\
= & \mathbb{E}_{\mathrm{P}} \big(\nabla_2^B L(Y, (f(X) + h(X))) - \nabla_2^B L(Y, f(X)) \big) - \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L(Y, f(X)) + o(h) \\
= & \mathbb{E}_{\mathrm{P}} \big(\nabla_2^B L(Y, (f(X) + h(X))) - \nabla_2^B L(Y, f(X)) - \nabla_{2,2}^B L(Y, f(X)) \big) + o(h) = o(h)
\end{aligned}$$

by definition of the B-derivative.

Remark 3 part 3. Let $\mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda})$ be a δ_1-neighborhood of $f_{\mathrm{P},\lambda}$. Because $\mathcal{H}$ is a RKHS and hence a vector space it follows for all $h \in \mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda})$ that $\left\| f_{\mathrm{P},\lambda} - h - 0 \right\|_{\mathcal{H}} \leq \delta_1$ and hence $h - f_{\mathrm{P},\lambda} \in \mathcal{N}_{\delta_1}(0) \subset$

$\mathcal{H}$. Note that $\nabla_2^B G(0, f_{\mathrm{P},\lambda})(\cdot)$ computed by (17) is a mapping from $\mathcal{H}$ to $\mathcal{H}$. For $\xi := h - f_{\mathrm{P},\lambda}$ we have $\|\xi\|_{\mathcal{H}} \le \delta_1$ and the reproducing property yields

$$\nabla_2^B G(0, f_{\mathrm{P},\lambda})(\xi) = 2\lambda\xi + \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot \xi\Phi(X).$$

Using (2) and (7) we obtain

$$\begin{aligned}
& \left\| 2\lambda\xi + \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot \xi\Phi(X) - 0 \right\|_{\mathcal{H}} \\
\le\ & 2\lambda \|\xi\|_{\mathcal{H}} + \left\| \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot \xi\Phi(X) \right\|_{\mathcal{H}} \\
\le\ & 2\lambda \|\xi\|_{\mathcal{H}} + \sup_{y \in Y} \left\| \nabla_{2,2}^B L(y, \cdot) \right\|_\infty \|\xi\|_\infty \|\Phi(x)\|_\infty \\
\le\ & 2\lambda \|\xi\|_{\mathcal{H}} + \kappa_2 \|\xi\|_{\mathcal{H}} \|k\|_\infty^3 \\
\le\ & \left(2\lambda + \kappa_2 \|k\|_\infty^3\right) \delta_1 ,
\end{aligned}$$

which shows that $\nabla_2^B G(0, f_{\mathrm{P},\lambda})(h - f_{\mathrm{P},\lambda})$ lies in a neighborhood of $0 \in \mathcal{H}$, for all $h \in \mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda})$.

Remark 3 part 4. Due to (17) we have to prove that

$$d_0 := \inf_{f_1 \ne f_2} \frac{\left\| 2\lambda(f_1 - f_2) + \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L\big(Y, f_{\mathrm{P},\lambda}(X)\big) \cdot (f_1 - f_2)\Phi(X) \right\|_{\mathcal{H}}}{\|f_1 - f_2\|_{\mathcal{H}}} > 0 .$$

If $f_1 \ne f_2$, then (2), (7), and $\lambda > \frac{1}{2}\kappa_2 \|k\|_\infty^3$ yield that

$$\begin{aligned}
& \left\| 2\lambda(f_1 - f_2) + \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L\big(Y, f_{\mathrm{P},\lambda}(X)\big) \cdot (f_1 - f_2)\Phi(X) \right\|_{\mathcal{H}} / \|f_1 - f_2\|_{\mathcal{H}} \\
\ge\ & \left(\|2\lambda(f_1 - f_2)\|_{\mathcal{H}} - \left\| \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot (f_1 - f_2)\Phi(X) \right\|_{\mathcal{H}} \right) / \|f_1 - f_2\|_{\mathcal{H}} \\
\ge\ & 2\lambda - \kappa_2 \|k\|_\infty^3 > 0
\end{aligned}$$

by our assumption, which gives the assertion.

Remark 3 part 5. The assumptions of Robinson's implicit function theorem, see Theorem 7, are valid for G due to the results of Remark 3 parts 1 to 4 and the assumption that $\nabla_2^B G(0, f_{\mathrm{P},\lambda})$ is strong. This gives part 5.

The result of Theorem 2 now follows from inserting (16) and (17) into Remark 3 part 5(v.4). Using (7) we see that S is bounded. The linearity of S follows from its definition and the inverse of S does exist by Theorem 7. If necessary we can restrict the range of S to $S(\mathcal{H})$ to obtain a bijective function $S_* : \mathcal{H} \to S(\mathcal{H})$ with $S_*(f) = S(f)$ for all $f \in \mathcal{H}$. Hence S^{-1} is also bounded and linear by Theorem 8. This gives the existence of a bounded BIF specified by (9) and (10). ■

A.2 Calculations for the Results in Section 3

For the proof of Corollary 5 we need the partial B-derivatives for the three loss functions and also have to check that $\nabla_2^B G(0, f_{\mathrm{P},\lambda})$ is strong. We shall compute the partial B-derivatives for these loss functions in advance.

A.2.1 $\underline{\varepsilon}$-INSENSITIVE LOSS

We shall show for the $\underline{\varepsilon}$-insensitive loss $L = L_{\underline{\varepsilon}}$ that

$$\nabla_2^B L(y,t)(h) = \begin{cases} -h & \text{if} \quad \{t < y - \underline{\varepsilon}\} \text{ or } \{y - t = \underline{\varepsilon}, h < 0\} \\ 0 & \text{if} \quad \{y - \underline{\varepsilon} < t < y + \underline{\varepsilon}\} \text{ or } \{y - t = \underline{\varepsilon}, h \geq 0\} \\ & \quad \text{or } \{y - t = -\underline{\varepsilon}, h < 0\} \\ h & \text{if} \quad \{t > y + \underline{\varepsilon}\} \text{ or } \{y - t = -\underline{\varepsilon}, h \geq 0\} \end{cases}$$

and $\nabla_{2,2}^B L(y,t)(h) = 0$.

For the derivation of $\nabla_2^B L(y,t)$ we need to consider 5 cases.

1. If $t > y + \underline{\varepsilon}$, we have $t + h > y + \underline{\varepsilon}$ as long as h is small enough. Therefore,

$$\nabla_2^B L(y,t)(h) + o(h) = L(y,t+h) - L(y,t) = t + h - y - \underline{\varepsilon} - (t - y - \underline{\varepsilon}) = h.$$

2. If $t < y - \underline{\varepsilon}$, we have $t + h < y + \underline{\varepsilon}$ if h is sufficiently small. Thus

$$\nabla_2^B L(y,t)(h) + o(h) = y - t - h - \underline{\varepsilon} - (y - t - \underline{\varepsilon}) = -h.$$

3. If $y - t \in (-\underline{\varepsilon}, \underline{\varepsilon})$ we have $y - t - h \in (-\underline{\varepsilon}, \underline{\varepsilon})$ for $h \to 0$. This yields $\nabla_2^B L(y,t)(h) + o(h) = 0 - 0 = 0$.

4. If $y - t = \underline{\varepsilon}$ we have to consider 2 cases. If $h \geq 0$ and small, then $-\underline{\varepsilon} < y - t - h < \underline{\varepsilon}$ and hence $\nabla_2^B L(y,t)(h) + o(h) = 0 - 0 = 0$.
   If $h < 0$, we have $y - t - h > \underline{\varepsilon}$ and thus

$$\nabla_2^B L(y,t)(h) + o(h) = y - t - h - \underline{\varepsilon} - 0 = -h.$$

5. If $y - t = -\underline{\varepsilon}$ we have again to consider 2 cases. If $h \geq 0$, we have $y - t - h < -\underline{\varepsilon}$. Hence

$$\nabla_2^B L(y,t)(h) + o(h) = t + h - y - \underline{\varepsilon} - 0 = h.$$

   If $h < 0$, we get $-\underline{\varepsilon} < y - t - h < \underline{\varepsilon}$ which gives $\nabla_2^B L(y,t)(h) + o(h) = 0 - 0 = 0$.

This gives the assertion for the first partial B-derivative. Using the same reasoning we obtain $\nabla_{2,2}^B L(y,t)(h) = 0$.

A.2.2 PINBALL-LOSS

It will be shown that for the pinball loss $L = L_{\tau-pin}$ we get

$$\nabla_2^B L(y,t)(h) = \begin{cases} (1-\tau)h & \text{if} \quad \{y - t < 0\} \text{ or } \{y - t = 0, h \geq 0\} \\ -\tau h & \text{if} \quad \{y - t > 0\} \text{ or } \{y - t = 0, h < 0\} \end{cases}$$

and $\nabla_{2,2}^B L(y,t)(h) = 0$.

For the calculation of $\nabla_2^B L(y,t)$ we consider 3 cases.

1. If $y-t<0$ we have $y-t-h<0$ for sufficiently small values of $|h|$. Hence

$$\begin{aligned}\nabla_2^B L(y,t)(h)+o(h) &= L(y,t+h)-L(y,t)\\ &= (\tau-1)(y-t-h)-(\tau-1)(y-t)=(1-\tau)h.\end{aligned}$$

2. If $y-t>0$ we have $y-t-h>0$ for sufficiently small values of $|h|$ which yields

$$\nabla_2^B L(y,t)(h)+o(h)=\tau(y-t-h)-\tau(y-t)=-\tau h.$$

3. Assume $y-t=0$. If $y-t-h<0$ we have

$$\nabla_2^B L(y,t)(h)+o(h)=(1-\tau)h.$$

If $y-t-h>0$ it follows

$$\nabla_2^B L(y,t)(h)+o(h)=\tau(y-t-h)-\tau(y-t)=-\tau h.$$

Together this gives the assertion for $\nabla_2^B L(y,t)(h)$. In the same way we get $\nabla_{2,2}^B L(y,t)(h)=0$.

A.2.3 HUBER LOSS

It will be shown that for the Huber loss $L=L_{c-Huber}$ we have

$$\nabla_2^B L(y,t)(h)=\begin{cases}-c\,\mathrm{sign}(y-t)h & \text{if} \quad |y-t|>c\\ -(y-t)h & \text{if} \quad |y-t|\le c\end{cases}$$

and

$$\nabla_{2,2}^B L(y,t)(h) = \begin{cases} h & \text{if} \quad \{y-t=c, h\ge 0\} \text{ or } \{y-t=-c, h<0\} \\ & \quad \text{or } \{|y-t|<c\} \\ 0 & \text{if} \quad \text{else}. \end{cases}$$

For the derivation of $\nabla_2^B L(y,t)$ we consider the following 5 cases.

1. Let $y-t=c$. If $h\ge 0$ or $y-t-h\le c$ then

$$\begin{aligned}\nabla_2^B L(y,t)(h)+o(h) &= L(y,t+h)-L(y,t)\\ &= \frac{1}{2}(y-t-h)^2-\frac{1}{2}(y-t)^2=-(y-t)h+\frac{h^2}{2}.\end{aligned}$$

If $h<0$ or $y-t-h>c>0$ we have

$$\begin{aligned}\nabla_2^B L(y,t)(h)+o(h) &= c|y-t-h|-\frac{c^2}{2}-\frac{1}{2}(y-t)^2\\ &= c(y-t-h)-\frac{c^2}{2}-\frac{c^2}{2}\\ &= c(c-h)-c^2=-(y-t)h.\end{aligned}$$

2. Now we consider the case $y-t=-c$. If $h \geq 0$ or $y-t-h \leq -c < 0$ we obtain

$$\begin{aligned} \nabla_2^B L(y,t)(h)+o(h) &= c|y-t-h| - \frac{c^2}{2} - \frac{1}{2}(y-t)^2 \\ &= c(c+h) - \frac{c^2}{2} - \frac{c^2}{2} = -(y-t)h. \end{aligned}$$

If $h < 0$ or $y-t-h > -c$ we get

$$\nabla_2^B L(y,t)(h)+o(h) = \frac{1}{2}(y-t-h)^2 - \frac{1}{2}(y-t)^2 = -(y-t)h + \frac{h^2}{2}.$$

3. If $y-t > c$, we have $y-t-h > c$ and thus

$$\begin{aligned} \nabla_2^B L(y,t)(h)+o(h) &= c|y-t-h| - \frac{c^2}{2} - c|y-t| + \frac{c^2}{2} \\ &= c(y-t-h) - c(y-t) = -ch = -c\,\mathrm{sign}(y-t)h. \end{aligned}$$

4. If $y-t < -c$, we have $y-t-h < -c$ and obtain analogously to (iii) that

$$\begin{aligned} \nabla_2^B L(y,t)(h)+o(h) &= c|y-t-h| - \frac{c^2}{2} - c|y-t| + \frac{c^2}{2} \\ &= c(-y+t+h) - c(-y+t) = ch = -c\,\mathrm{sign}(y-t)h. \end{aligned}$$

5. If $-c < y-t < c$, then $-c < y-t-h < c$ and

$$\nabla_2^B L(y,t)(h)+o(h) = \frac{1}{2}(y-t-h)^2 - \frac{1}{2}(y-t)^2 = -(y-t)h + \frac{h^2}{2}.$$

This gives the assertion for $\nabla_2^B L(y,t)(h)$. Only the first two cases, where $y-t=\pm c$, were necessary to compute, since in the other 3 parts the function is already F-differentiable, and thus also B-differentiable. For the second partial B-derivative we consider 3 cases.

1. Assume $y-t=c$. If $y-t-h < c$ then

$$\nabla_{2,2}^B L(y,t)(h)+o(h) = \nabla_2^B L(y,t+h) - \nabla_2^B L(y,t) = -(y-t-h) - (-(y-t)) = h.$$

If $y-t-h > c$ then $\nabla_{2,2}^B L(y,t)(h)+o(h) = -c - (-(y-t)) = 0$.

2. Assume $y-t=-c$. If $y-t-h < -c$ we obtain $\nabla_{2,2}^B L(y,t)(h)+o(h) = c - (-(y-t)) = 0$. If $y-t-h > -c$ then

$$\nabla_{2,2}^B L(y,t)(h)+o(h) = -(y-t-h) - (-(y-t)) = h.$$

3. Assume that $|y-t| \neq c$. Then $\nabla_2^B L(y,t+h) = \nabla_2^B L(y,t)$. The difference, and consequently $\nabla_{2,2}^B L(y,t)(h) = 0$.

This gives the assertion for Huber's loss function.

Proof of Corollary 5. Now that we have shown that these loss functions have bounded first and second partial B-derivatives, we are ready to check if $\nabla_2^B G(0, f_{\mathrm{P},\lambda})$ is strong in these cases. Recall that $\nabla_2^B G(0, f_{\mathrm{P},\lambda})$ is strong, if for all $\varepsilon^* > 0$ there exist a neighborhood $\mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda})$ and an interval $(-\delta_2, \delta_2)$ with $\delta_1, \delta_2 > 0$ such that for all $f_1, f_2 \in \mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda})$ and for all $\varepsilon \in (-\delta_2, \delta_2)$ we have

$$\big\| \big(G(\varepsilon, f_1) - g(f_1)\big) - \big(G(\varepsilon, f_2) - g(f_2)\big) \big\|_{\mathcal{H}} \leq \varepsilon^* \, \| f_1 - f_2 \|_{\mathcal{H}} \,, \tag{18}$$

where

$$\begin{aligned} g(f) \;=\; & 2\lambda f_{\mathrm{P},\lambda}(X) + \mathbb{E}_{\mathrm{P}} \nabla_2^B L\big(Y, f_{\mathrm{P},\lambda}(X)\big) \cdot \Phi(X) + 2\lambda \, \mathrm{id}_{\mathcal{H}}(f(X) - f_{\mathrm{P},\lambda}(X)) \\ & + \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L\big(Y, f_{\mathrm{P},\lambda}(X)\big) \cdot \langle (f(X) - f_{\mathrm{P},\lambda}(X)), \Phi(X) \rangle_{\mathcal{H}} \Phi(X) \,, \;\; f \in \mathcal{H}. \end{aligned}$$

Fix $\varepsilon^* > 0$. Obviously, (18) is valid for $f_1 = f_2$. For the rest of the proof we therefore fix arbitrary functions $f_1, f_2 \in \mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda})$ with $f_1 \neq f_2$. We obtain for the term on the left hand side of (18) that

$$\begin{aligned} & \Big\| \Big(2\lambda f_1(X) + \mathbb{E}_{(1-\varepsilon)\mathrm{P}+\varepsilon\mathrm{Q}} \nabla_2^B L(Y, f_1(X)) \cdot \Phi(X) \\ & \quad - 2\lambda f_{\mathrm{P},\lambda}(X) - \mathbb{E}_{\mathrm{P}} \nabla_2^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot \Phi(X) \\ & \quad - 2\lambda (f_1(X) - f_{\mathrm{P},\lambda}(X)) - \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot (f_1(X) - f_{\mathrm{P},\lambda}(X)) \Phi(X) \Big) \\ & \quad - \Big(2\lambda f_2(X) + \mathbb{E}_{(1-\varepsilon)\mathrm{P}+\varepsilon\mathrm{Q}} \nabla_2^B L(Y, f_2(X)) \cdot \Phi(X) \\ & \quad\quad - 2\lambda f_{\mathrm{P},\lambda}(X) - \mathbb{E}_{\mathrm{P}} \nabla_2^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot \Phi(X) \\ & \quad\quad - 2\lambda (f_2(X) - f_{\mathrm{P},\lambda}(X)) - \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot (f_2(X) - f_{\mathrm{P},\lambda}(X)) \Phi(X) \Big) \Big\|_{\mathcal{H}} \\ = \;\; & \Big\| \mathbb{E}_{(1-\varepsilon)\mathrm{P}+\varepsilon\mathrm{Q}} \Big(\nabla_2^B L(Y, f_1(X)) - \nabla_2^B L(Y, f_2(X)) \Big) \cdot \Phi(X) \\ & \quad - \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot (f_1(X) - f_2(X)) \Phi(X) \Big\|_{\mathcal{H}} \qquad (19) \\ \leq \;\; & |1-\varepsilon| \, \Big\| \mathbb{E}_{\mathrm{P}} \Big(\nabla_2^B L(Y, f_1(X)) - \nabla_2^B L(Y, f_2(X)) \\ & \quad\quad - \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot (f_1(X) - f_2(X)) \Big) \Phi(X) \Big\|_{\mathcal{H}} \\ & + |\varepsilon| \, \Big\| \mathbb{E}_{\mathrm{Q}} \Big(\nabla_2^B L(Y, f_1(X)) - \nabla_2^B L(Y, f_2(X)) \Big) \cdot \Phi(X) \Big\|_{\mathcal{H}} \\ & + |\varepsilon| \, \Big\| \mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot (f_1(X) - f_2(X)) \Phi(X) \Big\|_{\mathcal{H}} \\ =: \;\; & |1-\varepsilon| A + |\varepsilon| B + |\varepsilon| C \,. \qquad (20) \end{aligned}$$

We shall show that (20) is bounded from above by $\varepsilon^* \| f_1 - f_2 \|_{\mathcal{H}}$. When we look at the first partial B-derivatives of our loss functions, we see that we can separate them in 2 cases: for $L_{\underline{\varepsilon}}$ and $L_{\tau-pin}$ there are one or more discontinuities in $\nabla_2^B L$, whereas $\nabla_2^B L$ is continuous for $L_{c-Huber}$. Recall that the set $\mathfrak{D}$ of points where Lipschitz continuous functions are *not* Fréchet-differentiable, has Lebesgue measure zero by Rademacher's theorem (Rademacher, 1919). Define the function $h\big(y, f_1(x), f_2(x)\big) := \nabla_2^B L\big(y, f_1(x)\big) - \nabla_2^B L\big(y, f_2(x)\big)$. For $L \in \{L_{\underline{\varepsilon}}, L_{\tau-pin}\}$, denote the set of discontinuity points of $\nabla_2^B L$ by $\mathfrak{D}$. Take $f_1,\ f_2 \in \mathcal{N}_{\delta_1}(f_{\mathrm{P},\lambda})$. For $\nabla_2^B L(Y, f_{\mathrm{P},\lambda}(x)) \notin \mathfrak{D}$ we obtain $\nabla_2^B L(Y, f_1(x)) = \nabla_2^B L(Y, f_2(x))$ for sufficiently small δ_1 and hence $h(y, f_1(x), f_2(x)) = 0$. If, on the other hand, $\nabla_2^B L(Y, f_{\mathrm{P},\lambda}(x)) \in \mathfrak{D}$ and $f_1(x) < f_{\mathrm{P},\lambda}(x) < f_2(x)$ or $f_2(x) < f_{\mathrm{P},\lambda}(x) < f_1(x)$, then $\nabla_2^B L(Y, f_1(x)) \neq \nabla_2^B L(Y, f_2(x))$ and hence $h(y, f_1(x), f_2(x)) \neq 0$. Define $m = 2|\mathfrak{D}|$.

A.2.4 PINBALL LOSS

Using the first part of this proof we see that for the pinball loss $L = L_{\tau-pin}$ we obtain $|h(y, f_1(x), f_2(x))| \le c_1$, with $c_1 = 1$, $\mathfrak{D} = \{0\}$, $m = 2$, and $\nabla^B_{2,2} L(y,t) = 0$, for all $t \in \mathbb{R}$. For all $f \in \mathcal{N}_{\delta_1}(f_{P,\lambda})$ we get

$$|f(x) - f_{P,\lambda}(x)| \le \left\| f - f_{P,\lambda} \right\|_\infty \le \|k\|_\infty \left\| f - f_{P,\lambda} \right\|_{\mathcal{H}} \le \|k\|_\infty \delta_1 . \tag{21}$$

Further

$$|f_1(x) - f_2(x)| \le \|f_1 - f_2\|_\infty \le \|k\|_\infty \|f_1 - f_2\|_{\mathcal{H}} \le 2\|k\|_\infty \delta_1 . \tag{22}$$

Using (21), (22), and (11) we obtain

$$\begin{aligned}
A &= \left\| \mathbb{E}_P(\nabla^B_2 L(Y, f_1(X)) - \nabla^B_2 L(Y, f_2(X))) \cdot \Phi(X) \right\|_{\mathcal{H}} \\
&\le \mathbb{E}_P |h(Y, f_1(X), f_2(X))| \, |\Phi(X)| \\
&\le \|k\|^2_\infty \, \mathbb{E}_P |h(Y, f_1(X), f_2(X))| \mathbf{1}_{\{h \neq 0\}} \\
&\le \|k\|^2_\infty c_1 P\big(\nabla^B_2 L(Y, f_1(X)) \neq \nabla^B_2 L(Y, f_2(X))\big) \\
&= \|k\|^2_\infty \Big(P\big(\{Y - f_1(X) < 0\} \wedge \{Y - f_2(X) > 0\}\big) \\
&\qquad + P\big(\{Y - f_2(X) < 0\} \wedge \{Y - f_1(X) > 0\}\big) \Big) \\
&= \|k\|^2_\infty \int_X P\big(Y \in (f_2(x), f_1(x)) \,|x\big) + P\big(Y \in (f_1(x), f_2(x)) \,|x\big) dP_X(x) \\
&= \|k\|^2_\infty \int_X P\big(Y \in (f_2(x), f_2(x) + [f_1(x) - f_2(x)]) \,|x\big) \\
&\qquad + P\big(Y \in (f_1(x), f_1(x) + [f_2(x) - f_1(x)]) \,|x\big) dP_X(x) \\
&\le m \|k\|^2_\infty \int_X c_P |f_1(x) - f_2(x)|^{1+\xi_P} dP_X(x) \\
&\le m \|k\|^2_\infty c_P \|f_1 - f_2\|^{1+\xi_P}_\infty \\
&\le m c_P \|k\|^{3+\xi_P}_\infty \|f_1 - f_2\|^{1+\xi_P}_{\mathcal{H}} ,
\end{aligned}$$

where P_X denotes the marginal distribution of X. Similar calculations give that $B \le m c_Q \|k\|^{3+\xi_Q}_\infty \|f_1 - f_2\|^{1+\xi_Q}_{\mathcal{H}}$. We obtain $C = 0$, because $\nabla^B_{2,2} L(Y, f_{P,\lambda}(X)) = 0$. Hence, the term in (20) is less than or equal to

$$\begin{aligned}
& |1 - \varepsilon| m c_P \|k\|^{3+\xi_P}_\infty \|f_1 - f_2\|^{1+\xi_P}_{\mathcal{H}} + |\varepsilon| m c_Q \|k\|^{3+\xi_Q}_\infty \|f_1 - f_2\|^{1+\xi_Q}_{\mathcal{H}} \\
= \; & \big(|1 - \varepsilon| c_P \|k\|^{\xi_P}_\infty \|f_1 - f_2\|^{\xi_P}_{\mathcal{H}} + |\varepsilon| c_Q \|k\|^{\xi_Q}_\infty \|f_1 - f_2\|^{\xi_Q}_{\mathcal{H}} \big) m \|k\|^3_\infty \|f_1 - f_2\|_{\mathcal{H}} \\
\le \; & \varepsilon^* \|f_1 - f_2\|_{\mathcal{H}} ,
\end{aligned}$$

where $\varepsilon^* = (|1 - \varepsilon| c_P \|k\|^{\xi_P}_\infty 2^{\xi_P} \delta_1^{\xi_P} + |\varepsilon| c_Q \|k\|^{\xi_Q}_\infty 2^{\xi_Q} \delta_1^{\xi_Q}) m \|k\|^3_\infty$.

A.2.5 $\underline{\varepsilon}$-INSENSITIVE LOSS

The proof for the $\underline{\varepsilon}$-insensitive loss $L = L_{\underline{\varepsilon}}$ is analogous to the proof for $L_{\tau-pin}$, but with $c_1 = 2$, $\mathfrak{D} = \{-\underline{\varepsilon}, +\underline{\varepsilon}\}$, $m = 4$ and thus we must consider 4 cases instead of 2 where $h(y, f_1(x), f_2(x)) \neq 0$.

A.2.6 HUBER LOSS

For Huber's loss function $L = L_{c-Huber}$ we have $|\nabla^B_{2,2}L(y,t)| \leq 1 := c_2$ and $h(y, f_1(x), f_2(x))$ is bounded by $c_1 = 2c$. Let us define

$$\begin{aligned} h^*(y, f_{\mathrm{P},\lambda}(x), f_1(x), f_2(x)) &:= \nabla^B_2 L(y, f_1(x)) - \nabla^B_2 L(y, f_2(x)) \\ &\quad -\nabla^B_{2,2} L(y, f_{\mathrm{P},\lambda}(x)) \cdot (f_1(x) - f_2(x)). \end{aligned}$$

Somewhat tedious calculations show that there are 8 cases where $h^*(y, f_{\mathrm{P},\lambda}(x), f_1(x), f_2(x)) \neq 0$ and 6 cases where $h^*(y, f_{\mathrm{P},\lambda}(x), f_1(x), f_2(x)) = 0$. In each of the 8 cases, $y - f_{\mathrm{P},\lambda}(x) \in \{-c, c\}$ and $|h^*(y, f_{\mathrm{P},\lambda}(x), f_1(x), f_2(x))| \leq |f_1(x) - f_2(x)|$. Due to symmetry of the Huber loss function, the calculations are quite similar, therefore we only consider here some cases.

If $-c < Y - f_{\mathrm{P},\lambda}(x) < c$, then $\nabla^B_{2,2}L(Y, f_{\mathrm{P},\lambda}(x)) \cdot (f_1(x) - f_2(x)) = f_1(x) - f_2(x)$ and for sufficiently small δ_1, $\nabla^B_2 L(Y, f_1(x)) = -(Y - f_1(x))$ and $\nabla^B_2 L(Y, f_2(x)) = -(Y - f_2(x))$. A small calculation shows that $h^*(Y, f_{\mathrm{P},\lambda}(x), f_1(x), f_2(x)) = 0$.

By straightforward calculations we also obtain that $h^*(Y, f_{\mathrm{P},\lambda}(x), f_1(x), f_2(x)) = 0$ for the following 5 cases:

1. $Y - f_{\mathrm{P},\lambda}(x) < -c$ or $Y - f_{\mathrm{P},\lambda}(x) > c$,
2. $Y - f_{\mathrm{P},\lambda}(x) = -c$ and $f_{\mathrm{P},\lambda}(x) > f_2(x) > f_1(x)$,
3. $Y - f_{\mathrm{P},\lambda}(x) = -c$ and $f_1(x) > f_2(x) > f_{\mathrm{P},\lambda}(x)$,
4. $Y - f_{\mathrm{P},\lambda}(x) = c$ and $f_{\mathrm{P},\lambda}(x) > f_2(x) > f_1(x)$,
5. $Y - f_{\mathrm{P},\lambda}(x) = c$ and $f_1(x) > f_2(x) > f_{\mathrm{P},\lambda}(x)$.

If $Y - f_{\mathrm{P},\lambda}(x) = -c$ and $f_1(x) > f_{\mathrm{P},\lambda}(x) > f_2(x)$, we get $\nabla^B_2 L(Y, f_1(X)) = c$, $\nabla^B_2 L(Y, f_2(x)) = -(Y - f_2(x))$ and $\nabla^B_{2,2}L(Y, f_{\mathrm{P},\lambda}(x)) \cdot (f_1(x) - f_2(x)) = 0$. Hence,

$$h^*(Y, f_{\mathrm{P},\lambda}(x), f_1(x), f_2(x)) = c + Y - f_2(x) = f_{\mathrm{P},\lambda}(x) - f_2(x) \neq 0,$$

since $f_2(x) < f_{\mathrm{P},\lambda}(x)$.

Analogously, some calculations show that $h^*(Y, f_{\mathrm{P},\lambda}(x), f_1(x), f_2(x)) \neq 0$ for the following 7 cases:

1. $Y - f_{\mathrm{P},\lambda}(x) = -c$ and $f_2(x) > f_{\mathrm{P},\lambda}(x) > f_1(x)$,
2. $Y - f_{\mathrm{P},\lambda}(x) = -c$ and $f_{\mathrm{P},\lambda}(x) > f_1(x) > f_2(x)$,
3. $Y - f_{\mathrm{P},\lambda}(x) = -c$ and $f_2(x) > f_1(x) > f_{\mathrm{P},\lambda}(x)$,
4. $Y - f_{\mathrm{P},\lambda}(x) = c$ and $f_1(x) > f_{\mathrm{P},\lambda}(x) > f_2(x)$,
5. $Y - f_{\mathrm{P},\lambda}(x) = c$ and $f_2(x) > f_{\mathrm{P},\lambda}(x) > f_1(x)$,
6. $Y - f_{\mathrm{P},\lambda}(x) = c$ and $f_{\mathrm{P},\lambda}(x) > f_1(x) > f_2(x)$,
7. $Y - f_{\mathrm{P},\lambda}(x) = c$ and $f_2(x) > f_1(x) > f_{\mathrm{P},\lambda}(x)$.

Using (12) in (20) we get for the term A in (20) that

$$\begin{aligned} A &= \left\|\mathbb{E}_{\mathrm{P}} h^*(Y, f_{\mathrm{P},\lambda}(X), f_1(X), f_2(X))\Phi(X)\right\|_{\mathcal{H}} \\ &\leq \|k\|_\infty^2 \int |h^*(y, f_{\mathrm{P},\lambda}(x), f_1(x), f_2(x))| \mathbf{1}_{\{h^*\neq 0\}} d\mathrm{P}(x,y) \\ &\leq \|k\|_\infty^2 \int |f_1(x) - f_2(x)| \mathrm{P}\big(Y \in \{-c + f_{\mathrm{P},\lambda}(x), c + f_{\mathrm{P},\lambda}(x)\} \big| x\big) d\mathrm{P}_X(x) = 0. \end{aligned}$$

Also $C = \left\|\mathbb{E}_{\mathrm{P}} \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot (f_1(X) - f_2(X))\Phi(X)\right\|_{\mathcal{H}} \leq \kappa_2 \|k\|_\infty^3 \|f_1 - f_2\|_{\mathcal{H}}$. One can compute the analogous terms to A and C, say $A(\mathrm{Q})$ and $C(\mathrm{Q})$, respectively, where the integration is with respect to Q instead of P. Combining these expressions we obtain

$$\begin{aligned} B &= \left\|\mathbb{E}_{\mathrm{Q}}(\nabla_2^B L(Y, f_1(X)) - \nabla_2^B L(Y, f_2(X))) \cdot \Phi(X)\right\|_{\mathcal{H}} \\ &\leq \mathbb{E}_{\mathrm{Q}} \big|\nabla_2^B L(Y, f_1(X)) - \nabla_2^B L(Y, f_2(X)) - \\ &\qquad \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot (f_1(X) - f_2(X))\big| |\Phi(X)| \\ &\quad + \mathbb{E}_{\mathrm{Q}} \big|\nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}(X)) \cdot (f_1(X) - f_2(X))\big| |\Phi(X)| \\ &= A(\mathrm{Q}) + C(\mathrm{Q}) \leq \kappa_2 \|k\|_\infty^3 \|f_1 - f_2\|_{\mathcal{H}}. \end{aligned}$$

Hence, the term in (20) is less than or equal to $\varepsilon^* \|f_1 - f_2\|_{\mathcal{H}}$ where $\varepsilon^* = 2|\varepsilon|\kappa_2 \|k\|_\infty^3$. This gives the assertion, because $|\varepsilon|$ can be chosen arbitrarily small. ■

Proof of Corollary 6. Both partial F-derivatives $\nabla_2^F L_{log}(y,t) = 1 - 2\Lambda(y-t)$ and $\nabla_{2,2}^F L_{log}(y,t) = 2\Lambda(y-t)[1-\Lambda(y-t)]$ are clearly bounded, because $\Lambda(z) \in (0,1)$, $z \in \mathbb{R}$. We only have to show that $\nabla_2^B G(0, f_{\mathrm{P},\lambda})$ is strong for $L = L_{log}$, that is that the term in (19) is bounded by $\varepsilon^* \|f_1 - f_2\|_{\mathcal{H}}$ for arbitrary chosen $\varepsilon^* > 0$. A Taylor expansion gives for arbitrary $y, t_1, t_2 \in \mathbb{R}$ that

$$\Lambda(y - t_2) = \Lambda(y - t_1) + (t_1 - t_2)\Lambda(y - t_1)\big(1 - \Lambda(y - t_1)\big) + O((t_1 - t_2)^2). \tag{23}$$

Combining (2), (21), (22), and (23) we obtain

$$\begin{aligned} &\big|\mathbb{E}_{\mathrm{P}}\big(\nabla_2^B L(Y, f_1(X)) - \nabla_2^B L(Y, f_2(X)) - \nabla_{2,2}^B L(Y, f_{\mathrm{P},\lambda}) \cdot (f_1(X) - f_2(X))\big)\Phi(X)\big| \\ \leq\ & 2\|k\|_\infty^2 \mathbb{E}_{\mathrm{P}} \big|\Lambda(Y - f_2(X)) - \Lambda(Y - f_1(X)) \\ &\qquad -\Lambda(Y - f_{\mathrm{P},\lambda}(X))(1 - \Lambda(Y - f_{\mathrm{P},\lambda}(X))\big(f_1(X) - f_2(X)\big)\big| \\ \leq\ & 2\|k\|_\infty^2 \mathbb{E}_{\mathrm{P}} \big|\big(f_1(X) - f_2(X)\big)\big[\Lambda(Y - f_1(X))(1 - \Lambda(Y - f_1(X))) \\ &\qquad -\Lambda(Y - f_{\mathrm{P},\lambda}(X))(1 - \Lambda(Y - f_{\mathrm{P},\lambda}(X)))\big] + O((f_1(X) - f_2(X))^2)\big| \\ \leq\ & 2\|k\|_\infty^2 \mathbb{E}_{\mathrm{P}} \big(\|f_1 - f_2\|_\infty \big|\Lambda(Y - f_1(X))(1 - \Lambda(Y - f_1(X))) \\ &\qquad -\Lambda(Y - f_{\mathrm{P},\lambda}(X))(1 - \Lambda(Y - f_{\mathrm{P},\lambda}(X)))\big| + c_3 \|f_1 - f_2\|_\infty^2\big). \end{aligned} \tag{24}$$

A Taylor expansion around $f_{\mathrm{P},\lambda}(x)$ shows that $\Lambda(y - f_1(x))(1 - \Lambda(y - f_1(x)))$ equals

$$\begin{aligned} &\Lambda(y - f_{\mathrm{P},\lambda}(x))(1 - \Lambda(y - f_{\mathrm{P},\lambda}(x))) \\ +\ &\big(f_{\mathrm{P},\lambda}(x) - f_1(x)\big)\Lambda(y - f_{\mathrm{P},\lambda}(x))(1 - \Lambda(y - f_{\mathrm{P},\lambda}(x)))(1 - 2\Lambda(y - f_{\mathrm{P},\lambda}(x))) \\ +\ &O((f_1(x) - f_{\mathrm{P},\lambda}(x))^2). \end{aligned}$$

Using this expansion and (2), (21), and (22) it follows that the term in (24) is bounded by

$$\begin{aligned} & 2\|k\|_\infty^2 \mathbb{E}_P\big(\|f_1 - f_2\|_\infty \big(\big\|f_1 - f_{P,\lambda}\big\|_\infty/4 + c_4\delta_1^2\|k\|_\infty^2\big) + c_3\|f_1 - f_2\|_\infty^2\big) \\ \leq \quad & \|k\|_\infty^4\big(\delta_1/2 + 2c_4\delta_1^2\|k\|_\infty + 4c_3\delta_1\big)\|f_1 - f_2\|_{\mathcal{H}}\,. \end{aligned} \tag{25}$$

Using the Lipschitz continuity of $\nabla_2^B L(y,\cdot)$, (2), and (23) we obtain

$$\begin{aligned} & |\varepsilon|\,\mathbb{E}_{Q-P}\big|\big(\nabla_2^B L(Y, f_1(X)) - \nabla_2^B L(Y, f_2(X))\big)\cdot\Phi(X)\big| \\ \leq \quad & |\varepsilon|\,\|k\|_\infty^2\,\mathbb{E}_{|Q-P|}\big|\nabla_2^B L(Y, f_1(X)) - \nabla_2^B L(Y, f_2(X))\big| \\ \leq \quad & |\varepsilon|\,\|k\|_\infty^3\,\|f_1 - f_2\|_{\mathcal{H}}\,. \end{aligned} \tag{26}$$

Combining (25) and (26) shows that the term in (19) is bounded by $\varepsilon^*\|f_1 - f_2\|_{\mathcal{H}}$ with the positive constant $\varepsilon^* = \|k\|_\infty^3\big(\delta_1\|k\|_\infty/2 + 2c_4\delta_1^2\|k\|_\infty^2 + 4c_3\delta_1\|k\|_\infty + |\varepsilon|\big)$, where $\delta_1 > 0$ and $\varepsilon > 0$ can be chosen as small as necessary. ■

References

V.I. Averbukh and O.G. Smolyanov. The theory of differentiation in linear topological spaces. *Russian Mathematical Surveys*, 22:201–258, 1967.

V.I. Averbukh and O.G. Smolyanov. The various definitions of the derivative in linear topological spaces. *Russian Mathematical Surveys*, 23:67–113, 1968.

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.

A. Christmann and I. Steinwart. Consistency and robustness of kernel based regression in convex minimization. *Bernoulli*, 13:799–819, 2007.

F.H. Clarke. *Optimization and Nonsmooth Analysis*. Wiley & Sons, New York, 1983.

L.T. Fernholz. *Von Mises Calculus for Statistical Functionals*, volume 19 of *Lecture Notes in Statistics*. Springer, New York, 1983.

F.R. Hampel. Contributions to the theory of robust estimation. Unpublished Ph.D. thesis, Dept. of Statistics, University of California, Berkeley, 1968.

F.R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69:383–393, 1974.

F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, and W.A. Stahel. *Robust statistics: The Approach Based on Influence Functions*. Wiley & Sons, New York, 1986.

P.J. Huber. Robust estimation of a location parameter. *Ann. Math. Statist.*, 35:73–101, 1964.

C. Ip and J. Kyparisis. Local convergence of quasi-Newton methods for B-differentiable equations. *Mathematical Programming*, 56:71–89, 1992.

R. Koenker. *Quantile Regression*. Cambridge University Press, New York, 2005.

R.W. Koenker and G.W. Bassett. Regression quantiles. *Econometrica*, 46:33–50, 1978.

P.D. Lax. *Functional Analysis*. Wiley & Sons, New York, 2002.

R.A. Maronna, R.D. Martin, and V.J. Yohai. *Robust Statistics. Theory and Methods*. Wiley & Sons, New York, 2006.

H. Rademacher. Über partielle und totale Differenzierbarkeit. *Math. Ann.*, 79:254–269, 1919.

H. Rieder. *Robust Asymptotic Statistics*. Springer, New York, 1994.

S.M. Robinson. Local structure of feasible sets in nonlinear programming, Part III: Stability and sensitivity. *Mathematical Programming Study*, 30:45–66, 1987.

S.M. Robinson. An implicit-function theorem for a class of nonsmooth functions. *Mathematics of Operations Research*, 16:292–309, 1991.

B. Schölkopf and A.J. Smola. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, Massachusetts, 2002.

I. Steinwart. How to compare different loss functions. *Constrained Approximation*, 26:225–287, 2007.

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 and A. Christmann. How SVMs can estimate quantiles and the median. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, Massachusetts, 2008a.

I. Steinwart and A. Christmann. *Support Vector Machines*. Springer, New York, 2008b.

I. Takeuchi, Q.V. Le, T.D. Sears, and A.J. Smola. Nonparametric quantile estimation. *Journal of Machine Learning Research*, 7:1231–1264, 2006.

V.N. Vapnik. *Statistical Learning Theory*. Wiley & Sons, New York, 1998.

Journal of Machine Learning Research 9 (2008) 937-965 Submitted 8/07; Revised 2/08; Published 5/08

Accelerated Neural Evolution through Cooperatively Coevolved Synapses

Faustino Gomez TINO@IDSIA.CH
Jürgen Schmidhuber[*] JUERGEN@IDSIA.CH
Dalle Molle Institute for Artificial Intelligence (IDSIA)
Galleria 2, Manno (Lugano), Switzerland

Risto Miikkulainen RISTO@CS.UTEXAS.EDU
Department of Computer Sciences
University of Texas, Austin, TX 78712 USA

Editor: Melanie Mitchell

Abstract

Many complex control problems require sophisticated solutions that are not amenable to traditional controller design. Not only is it difficult to model real world systems, but often it is unclear what kind of behavior is required to solve the task. Reinforcement learning (RL) approaches have made progress by using direct interaction with the task environment, but have so far not scaled well to large state spaces and environments that are not fully observable. In recent years, neuroevolution, the artificial evolution of neural networks, has had remarkable success in tasks that exhibit these two properties. In this paper, we compare a neuroevolution method called Cooperative Synapse Neuroevolution (CoSyNE), that uses cooperative coevolution at the level of individual synaptic weights, to a broad range of reinforcement learning algorithms on very difficult versions of the pole balancing problem that involve large (continuous) state spaces and hidden state. CoSyNE is shown to be significantly more efficient and powerful than the other methods on these tasks.

Keywords: coevolution, recurrent neural networks, non-linear control, genetic algorithms, experimental comparison

1. Introduction

In many decision making processes such as manufacturing, aircraft control, and robotics researchers are faced with the problem of controlling systems that are highly complex and unstable. A controller or *agent* must be built that observes the state of the system, or *environment*, and outputs a control signal that affects future states of the environment in some desirable way.

The problem with designing or programming such controllers by direct engineering methods is twofold: (1) The environment is often non-linear and noisy so that it is impossible to obtain the kind of accurate and tractable mathematical model required by these methods. (2) The task is complex enough that there is very little *a priori* knowledge of what constitutes a reasonable, much less optimal, control strategy.

These two problems have compelled researchers to explore methods based on Dynamic Programming, for example Reinforcement Learning (RL; Sutton and Barto, 1998). Instead of trying to pre-program a response to every likely situation, an agent *learns* the utility of being in each state

∗. Also at Technische Universität München Boltzmannstr. 3, 85748 Garching, München, Germany.

(i.e., a value-function) from a reward signal it receives while interacting directly with the environment. In principle, RL methods can solve these problems: they do not require a mathematical model (i.e., the state transition probabilities) of the environment and can solve many problems where examples of correct behavior are not available. However, in practice, they have not scaled well to large state spaces or tasks where the state of the environment is not fully observable to the agent. This is a serious problem because the real world is continuous (i.e., there are an infinite number of states) and artificial agents, like natural organisms, are necessarily constrained in their ability to fully perceive their environment.

More recently, methods for evolving artificial neural networks or *neuroevolution* have shown promising results on continuous, partially observable tasks (Gomez, 2003; Nolfi and Parisi, 1995; Yamauchi and Beer, 1994). Our previous method, Enforced SubPopulations, is a particularly effective neuroevolution algorithm that has been a applied successfully to many domains (Perez-Bergquist, 2001; Lubberts and Miikkulainen, 2001; Greer et al., 2002; Whiteson et al., 2003; Bryant and Miikkulainen, 2003; Gomez et al., 2001; Grasemann and Miikkulainen, 2005), including the real world reinforcement learning task of finless rocket control (Gomez and Miikkulainen, 2003). The goal of this paper is to present a new algorithm that builds on ESP called Cooperative Synapse Neuroevolution (CoSyNE), and compare it to a wide range of other learning systems in a setting that is both challenging and practical. To this end, we have chosen a set of pole balancing tasks ranging from the trivial to versions that are extremely difficult for some of todays most advanced methods.

The paper is organized as follows: in Section 2, we discuss the general neuroevolution paradigm. In Section 3, the underlying approach used by CoSyNE, cooperative coevolution is described. In Section 4, the CoSyNE algorithm is presented. Section 5 presents our experiments comparing CoSyNE with value function, policy search, and other evolutionary methods. Sections 6 and 7 provide some discussion of our overall results, and conclusions.

2. Neuroevolution

The basic idea of Neuroevolution (NE; Yao, 1999) is to search the space of neural network policies directly using a genetic algorithm. In contrast to *ontogenetic* learning involving a single agent that learns incrementally (i.e., value-based RL), NE uses a population of solutions. The individual solutions are not modified during evaluation; instead, adaptation arises through repeatedly recombining the population's most fit individuals in a kind of collective or *phylogenetic* learning. The population gradually improves as a whole until a sufficiently fit individual is found.

In NE, neural network specifications are encoded in string representations or *chromosomes* (see Figure 1). A chromosome can encode any relevant network parameter including synaptic weight values, number of processing units, connectivity (topology), learning rate, etc. These network *genotypes* are then evolved in a sequence of generations. Each generation each genotype is mapped to its network phenotype (i.e., the actual network), and then evaluated in the problem environment and awarded a fitness score that quantifies its performance in some desirable way. After this evaluation phase, genotypes are selected from the population according to fitness through a variety of possible schemes (e.g., fitness proportional, linear ranking, tournament selection, etc.), and then mated through crossover and possibly mutated to form new genotypes that usually replace the least fit members of the population. This cycle repeats until a sufficiently fit network is found, or some other stopping criteria is met.

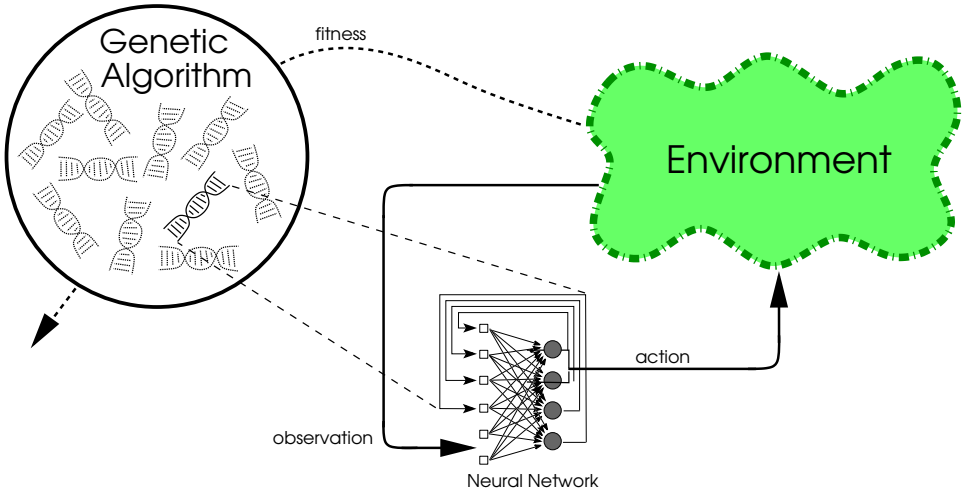


Figure 1: **Neuroevolution.** Each chromosome is transformed into a neural network phenotype and evaluated on the task. The agent receives input from the environment (observation) and propagates it through its neural network to compute an output signal (action) that affects the environment. At the end of the evaluation, the network is assigned a fitness according to its performance. The networks that perform well on the task are mated to generate new networks.

NE approaches differ primarily in how they encode neural network specifications into genetic strings. Direct encoding schemes represent the parameters explicitly on the chromosome as binary or real numbers that are mapped directly to the phenotype (Belew et al., 1991; Jefferson et al., 1991; Moriarty, 1997; Gomez, 2003; Stanley and Miikkulainen, 2002). Indirect encodings operate at a higher level of abstraction. Some simply provide a coarse description such as delineating a neuron's receptive field (Mandischer, 1993) or connective density (Harp et al., 1989), while others are more algorithmic, providing growth rules in the form of graph generating grammars (Kitano, 1990; Voigt et al., 1993; Gruau et al., 1996b). These schemes have the advantage that very large networks can be represented without requiring large chromosomes. Our CoSyNE method is a direct encoding method that does not evolve topology.

By searching the space of policies directly, NE can be applied to reinforcement learning problems without using a value function—neural network controllers map observations from the environment directly to actions. This mapping is potentially powerful: neural networks are universal function approximators that can generalize and tolerate noise. Networks with feedback connections (i.e., recurrent networks) can maintain internal state extracted from a history of inputs, allowing them to solve partially observable tasks. By evolving these networks instead of training them, NE avoids the problem of vanishing error gradients that affect recurrent network learning algorithms (Hochreiter et al., 2001). For NE to work, the environment need not satisfy any particular constraints—it can be continuous and partially observable. All that concerns a NE system is that the network representations be large enough to solve the task and that there is an effective way to evaluate the relative quality of candidate solutions.

Algorithm 1: Cooperative Coevolution (n, m)

```
1  Initialize {P_1, ..., P_n}
2  repeat
3      repeat
4          for j = 1 to n do     // construct complete solution
5              x_ij = Select(P_j)
6              x ⇐ x_ij          // add subgenotype to complete solution
7          end
8          Evaluate(x)
9      until enough solutions evaluated
10     for i = 1 to n do         // each subpopulation reproduces independently
11         Recombine(P_i)
12     end
13 until solution is found
```

3. Cooperative Coevolution

In natural ecosystems, organisms of one species compete and/or cooperate with many other different species in their struggle for resources and survival. The fitness of each individual changes over time because it is coupled to that of other individuals inhabiting the environment. As species evolve they specialize and co-adapt their survival strategies to those of other species. This phenomenon of *coevolution* has been used to encourage complex behaviors in GAs.

Most coevolutionary problem solving systems have concentrated on competition between species (Darwen, 1996; Pollack et al., 1996; Paredis, 1994; Miller and Cliff, 1994; Rosin, 1997). These methods rely on establishing an "arms race" where each species produces stronger and stronger strategies for the others to defeat. This is a natural approach for problems such as game-playing where often an optimal opponent is not available.

A very different kind of coevolutionary model emphasizes cooperation. Cooperative coevolution is motivated, in part, by the recognition that the complexity of difficult problems can be reduced through modularization (e.g., the human brain; Grady, 1993). In cooperative coevolutionary algorithms the species represent solution components. Each individual forms a part of a complete solution but need not represent anything meaningful on its own. The components are evolved by measuring their contribution to complete solutions and recombining those that are most beneficial to solving the task.

Algorithm 1 outlines the basic operation of a generic cooperative coevolutionary algorithm. The first parameter n specifies the number of species (components) that will be coevolved. Each species has its own subpopulation $P_i, i = 1..n$, containing m *subgenotypes*, $x_{ij} \in P_i, j = 1..m$ which are initialized with random values (line 1). Assuming complete solutions of fixed size, n determines the granularity at which the coevolutionary search is conducted.

Next, some number of complete solutions are constructed and evaluated (lines 3-9). A complete solution $\mathbf{x}$ is formed by combining one subgenotype, selected according to some policy, from each of the subpopulations. Usually, the string representations of each subgenotype are simply concatenated in a predefined order to form a single chromosome. Each $\mathbf{x}$ is evaluated in the problem environment and a fitness score is assigned to each constituent subgenotype. Since the number of evaluations per

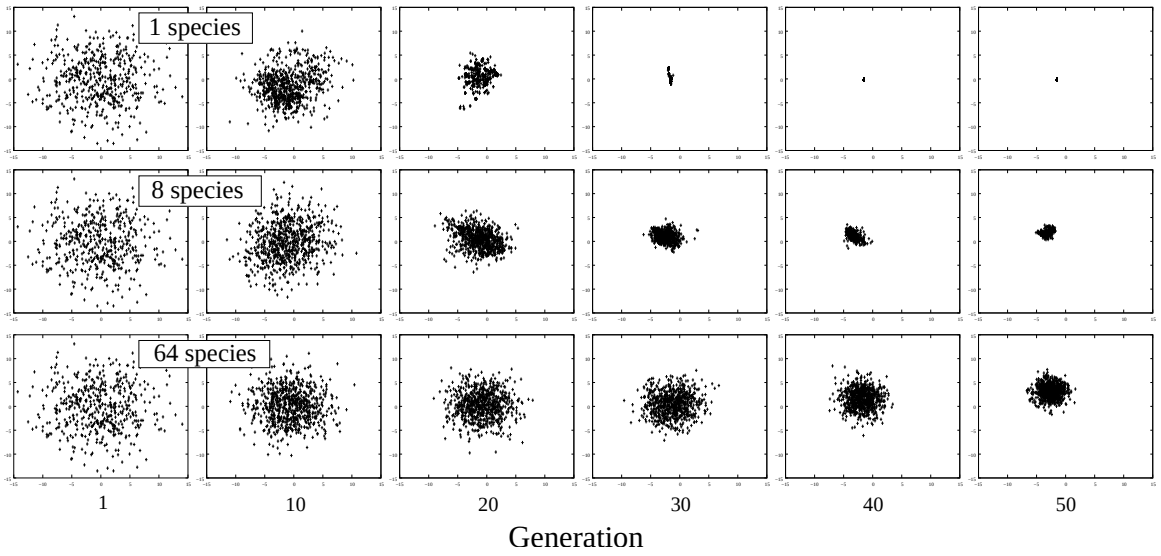


Figure 2: **Convergence speed for varying numbers of species.** Each row shows a PCA projection of 128-dimensional chromosomes at different generations during an evolutionary run optimizing a very simple multi-modal test function. All three runs start with the same set of complete solution (see first column). In row 1, the solutions are not coevolved because each genotype is a complete solution. In row 2, 8 species are coevolved (i.e., have to be combined to form a the complete solutions shown in the plots), and in row 3, 64 species are coevolved. The more species there are to cooperate, the longer it takes for the evolution to converge.

generation can exceed m, subgenotypes can participate in more that one evaluation per generation. Therefore, the fitness score of each x_{ij} at the end of a generation is some function of the raw fitness scores accumulated over multiple evaluations, and is considered a *subjective measure* because it is coupled with that of its *collaborators*, in contrast to an objective measure that only depends on the individual itself (Wiegand, 2003). The exact number of evaluations per subgenotype depends on the collaboration scheme employed by a particular algorithm. One common approach, for example, is simply to evaluate each subgenotype in n trials, and then take the average or best fitness.

Once enough evaluations have been performed, each subpopulation is recombined to form new subgenotypes, as in a normal GA.

Early work in this area was done by Holland and Reitman (1978) in Classifier Systems. A population of rules was evolved by assigning a fitness to each rule based on how well it interacted with other rules. This approach has been used in learning neural network classifiers, in coevolution of cascade correlation networks, and in coevolution of radial basis functions (Eriksson and Olsson, 1997; Horn et al., 1994; Paredis, 1995; Whitehead and Choate, 1995). More recently, Husbands and Mill (1991) and Potter and De Jong (1995) developed a method called Cooperative Coevolutionary GA (CCGA) in which each of the species is evolved independently in its own population. As in Classifier Systems, individuals in CCGA are rewarded for making favorable contributions to complete solutions, but members of different populations (species) are not allowed to mate. A

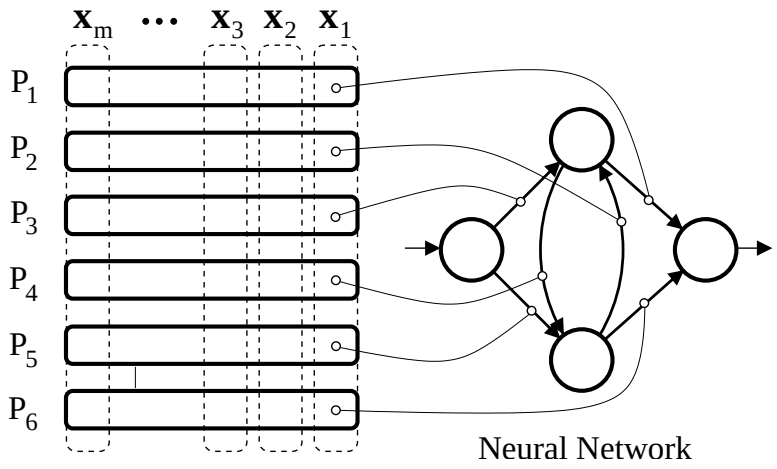


Figure 3: **The CoSyNE method for neuroevolution.** On the left, the figure shows and example population consisting of six subpopulations, $P_1..P_6$, each containing m weight values. To create a network, first the weights at a given index in each subpopulation are collected into a chromosome $\mathbf{x}$, then the weights are mapped to their corresponding synapses in a predefined network architecture with six connections, shown at right.

particularly powerful idea is to combine cooperative coevolution with neuroevolution so that the benefits of evolving neural networks can be enhanced further through improved search efficiency.

Much of the motivation for using the cooperative coevolutionary approach is based on the intuition that many problems may be decomposable into weakly coupled low-dimensional subspaces that can be searched semi-independently by separate species (Wiegand et al., 2001; Jansen and Wiegand, 2003, 2004; Panait et al., 2006). Our experience shows that there may be another, complementary, explanation as to why cooperative coevolution in many cases outperforms single-population algorithms. Figure 2 compares the convergence behavior of the same initial population of complete solutions using different number of species: 1, 8, and 64. Each point represents a 128-dimensional chromosome projected onto 2-D using Principal Component Analysis. The chromosomes are coevolved to optimize a continuous multi-modal test function[1] with 128 randomly distributed maxima that represent valid solutions. As the number of species increases, the selection of subgenotypes for reproduction becomes less greedy, causing the search points that are evaluated each generation to converge more slowly, providing more paths toward better solutions (not shown). In a normal evolutionary algorithm, a subgenotype suffers the fate of the complete solution to which it is attached. If the complete solution performs with high fitness, the subgenotype is retained in the population, even if it is not ultimately beneficial to the search; if it is less fit then this potentially useful component (if combined with other subgenotypes in the population) is lost. This diversity sustaining mechanism is exploited fully in the CoSyNE algorithm, introduced next.

1. The URL is http://www.cs.uwyo.edu/˜wspears/multi.kennedy.html.

Algorithm 2: CoSyNE (n, m, Ψ)

```
1  Initialize P = {P_1, ..., P_n}
2  repeat
3      for j = 1 to m do
4          x_j ⇐ (x_1j, ..., x_nj)     // form complete solution
5          Evaluate(x_j, Ψ)
6      end
7      O ⇐ Recombine(P)
8      for i = 1 to n do
9          Sort(P_i)
10         for k = 1 to l do     // replace least fit weights with
11             x_i,m−k ⇐ o_ik     // weights from offspring nets
12         end
13         for j = 1 to m do
14             prob(x_ij) ⇐ F(P, i, j)     // assign probability to each weight
15             if rand() < prob(x_ij) then
16                 mark(x_ij)     // mark weight for permutation probabilistically
17             end
18         end
19         PermuteMarked(P_i)     // see Figure 4
20     end
21 until solution is found
```

4. Cooperative Synapse Neuroevolution (CoSyNE)

Previous Cooperative Coevolution NE methods decomposed networks at the neuron level (Moriarty, 1997; Potter and De Jong, 1995; Gomez, 2003). This is a natural approach dictated by phenotypic structure: networks consist of multiple processing units that function in parallel. In contrast, CoSyNE evolves at the lowest possible level of granularity, the level of the individual synaptic weight. For each network connection, there is a separate subpopulation consisting of real valued weights. Like neuron-level methods such as ESP, networks are constructed by selecting one member from each subpopulation and plugging them into a predefined network topology.

Algorithm 2 describes the CoSyNE algorithm in pseudocode. First (line 1), a population $\mathcal{P}$ consisting of n subpopulations $P_i, i = 1..n$, is created, where n is the number of synaptic weights in the networks to be evolved, determined by a user-specified network architecture Ψ. Each subpopulation is initialized to contain m real numbers, $x_{ij} = \mathcal{P}_{ij} \in P_i, j = 1..m$, chosen from a uniform probability distribution in the interval $[-\alpha, \alpha]$. The population is thereby represented by an $n \times m$ matrix.

CoSyNE then loops through a sequence of *generations* until a sufficiently good network is found (lines 2-21). Each generation starts by constructing a complete network chromosome $\mathbf{x}_j = (x_{1j}, x_{2j}, \ldots, x_{nj})$ from each row in $\mathcal{P}$. The m resulting chromosomes are transformed into networks by assigning their weights to their corresponding synapses, in Ψ (line 4; see Figure 3).

After all of the networks have been evaluated (line 5) and assigned a fitness, the top quarter with the highest fitness (i.e., the parents) are recombined (line 7) using crossover and mutation. Recombination produces a pool of offspring O consisting of l new network chromosomes $\mathbf{o}_k$, where

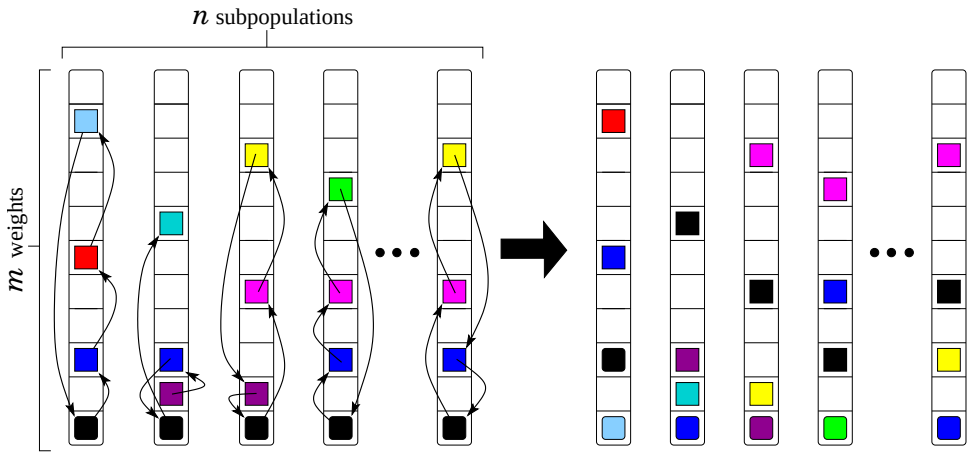


Figure 4: **Probabilistic permutations.** On the left is the set of subpopulations before permutation. The colored boxes are denote those genotypes that have been marked for permutation based on Equation 1. As the individuals are sorted by fitness within each subpopulation, notice that the less fit individuals have a higher probability of being permuted. On the right, the marked individuals have been permuted among themselves with each subpopulation. All unmarked genotypes remain part of the same complete solution.

$o_{ik} = O_{ik} \in O_i, i = 1..n, k = 1..l$, and O_i is the offspring subpopulation corresponding to P_i. The subpopulations are then sorted by fitness (line 9), and the weights from the new networks are added to $\mathcal{P}$ by replacing the least fit weights in their corresponding subpopulation (i.e., the P with the same index i; lines 10-11).

At this point the algorithm functions as a conventional neuroevolution system that evolves complete network chromosomes. In order to *coevolve* the synaptic weights, the subpopulations are permuted so that each weight forms part of a potentially different network in the next generation. Permutation is performed probabilistically. First, weights are marked randomly according to probabilities assigned by a user-defined function $F()$ (lines 14-17). Then the marked weights are permuted amongst themselves (see Figure 4). The function $F()$ can be anything from as simple as $\text{prob}(x_{ij}) = 1.0, \forall i, j$, in which case all weights are permuted, or more sophisticated:

$$\text{prob}(x_{ij}) = 1 - \sqrt[n]{\frac{f(x_{ij}) - f_i^{min}}{f_i^{max} - f_i^{min}}} \tag{1}$$

where $f(x_{ij})$ is the fitness of subgenotype (weight) x_{ij}, and f_j^{min} and f_j^{max} are, respectively, the fitness of the least and most fit individuals in subpopulation i. In this case, the probability of disrupting the network $\mathbf{x}_j$ is inversely proportional to its relative fitness, so that weight combinations that receive high fitness are more likely to be preserved, while those with low fitness are more likely to be disrupted and their constituents used to search for new complete solutions. In the experiments below, the simpler function that permutes all weights, except for the newly inserted offspring weights, was found to work well.

The basic CoSyNE framework does not specify how the weights are grouped in the complete solution chromosomes (i.e., which entry in the chromosome corresponds to which synapse) or which

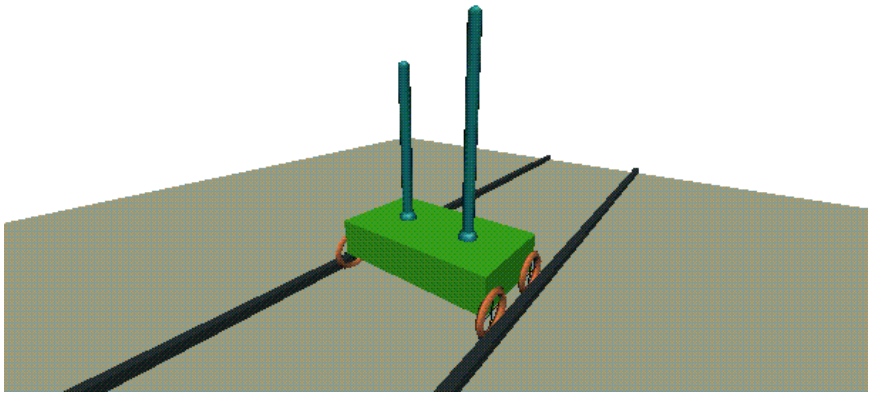


Figure 5: **The double pole balancing system.** Both poles must be balanced simultaneously by applying a continuous force to the cart. The system becomes more difficult to control as the poles assume similar lengths and if the velocities are not provided to the controller. The figure is a snapshot of a 3D real-time simulation.

genetic operators are used. In the implementation used in this paper, the weights of each neuron are grouped together (i.e., form a substring) and are separated into adjacent input, output, and recurrent weight segments, and the neuron substrings are concatenated together in a fixed order. For the genetic operators, we use multi-point crossover where 1-point crossover is applied to each neuron segment of the chromosome to generate two offspring, and mutation where each weight in $\mathcal{P}$ has a probability of being perturbed by Cauchy distributed noise with zero mean $\alpha = 0.3$.

5. Experiments

We compared CoSyNE to a broad range of learning algorithms on a sequence of increasingly difficult versions of the pole balancing task. This scheme allows us to compare methods at different levels of task complexity, exposing the strengths and limitations of each method with respect to specific challenges introduced by each succeeding task.

5.1 The Pole Balancing Problem

The basic pole balancing or inverted pendulum system consists of a pole hinged to a wheeled cart on a finite stretch of track. The objective is to apply a force to the cart at regular intervals such that the pole is balanced indefinitely and the cart stays within the track boundaries. This task has been a popular artificial learning testbed for over 30 years (Michie and Chambers, 1968; Anderson, 1989; Jang, 1992; Lin and Mitchell, 1992; Whitley et al., 1993) because it requires solving the temporal credit assignment problem, and is a good surrogate for a more general class of unstable control problems such as bipedal robot walking, and rocket guidance.

This long history notwithstanding, it turns out that the basic pole balancing problem can be solved easily by random search. To make it challenging for artificial learners, a variety of extensions to the basic pole-balancing task have been suggested. (Wieland, 1991) presented several variations that can be grouped into two categories: (1) modifications to the mechanical system itself, such as

adding a second pole either next to or on top of the other, and (2) restricting the amount of state information that is given to the controller; for example, only providing the cart position and the pole angle. The first category makes the task more difficult by introducing non-linear interactions between the poles. The second makes the task non-Markov, requiring the controller to employ short term memory to disambiguate underlying process states. Together, these extensions represent a family of tasks that can effectively test algorithms designed to learn control policies.

The sequence of comparisons presented below begins with a single pole version and then moves on to progressively more challenging variations culminating in a version where two separate poles of different length must be balanced simultaneously without the benefit of velocity information (see Appendix A for the equations of motion).

5.2 Other Methods

CoSyNE was compared to eight *ontogenetic* methods and seven *phylogenetic* methods in the pole balancing domain:

5.2.1 ONTOGENETIC METHODS

Random Weight Guessing (RWG) where the network weights are chosen at random (i.i.d.) from a uniform distribution. This approach is used to give an idea of how difficult each task is to solve by simply guessing a good set of weights.

Policy Gradient RL (PGRL; Sutton et al., 2000) where sampled Q-values are used to differentiate the performance of a given policy with respect to its parameters. The policy was implemented using a feed-forward neural network with one hidden layer.

Recurrent Policy Gradients (RPG; Wierstra et al., 2007) where a stochastic policy is represented by a Long Short-Term Memory network (LSTM; Hochreiter and Schmidhuber, 1997) trained with BackPropagation Through Time (Werbos, 1990). The gradient of the expected future reward over all possible state trajectories with respect to the policy parameters is calculated by Monte Carlo approximation. To reduce variance in the approximation, a *baseline* representing the expected average reward is used.

Value and Policy Search (VAPS; Meuleau et al., 1999) extends the work of Baird and Moore (1999) to policies that can make use of memory. The algorithm uses stochastic gradient descent to search the space of finite policy graph parameters. A policy graph is a state automaton that consists of nodes labeled with actions that are connected by arcs labeled with observations. When the system is in a particular node, the action associated with that node is taken and the underlying Markov environment transitions to the next observation that determines which arc is followed to the next action node.

Q-learning with MLP (Q-MLP): This method is the basic Q-learning algorithm (Watkins and Dayan, 1992) that uses a Multi-Layer Perceptron (i.e., a feed-forward artificial neural network) to map state-action pairs to values $Q(s,a)$. The input layer of the network has one unit per state variable and one unit per action variable. The output layer consists of a single unit indicating the Q-value. Values are learned through gradient descent on the prediction error using the backpropagation algorithm. This kind of approach has been studied widely with

success in tasks such as pole-balancing (Lin and Mitchell, 1992), pursuit-evasion games (Lin, 1992), and backgammon (Tesauro, 1992).

Sarsa(λ) with Case-Based function approximator (SARSA-CABA; Santamaria et al., 1998): This method consists of the Sarsa on-policy Temporal Difference control algorithm with eligibility traces that uses a case-based memory to approximate the *Q*-function. The memory explicitly records state-action pairs (i.e., cases) that have been experienced by the controller. The value of a new state-action pair not in the memory is calculated by combining the values of the *k*-nearest neighbors. A new case is added to the memory whenever the current query point is further than a specified *density threshold*, t_d away from all cases already in the memory. The case-based memory provides a locally-linear model of the Q-function that concentrates its resources on the regions of the state space that are most relevant to the task and expands its coverage dynamically according to t_d.

Sarsa(λ) with CMAC function approximator (SARSA-CMAC; Santamaria et al., 1998): This is the same as SARSA-CABA except that it uses a Cerebellar Model Articulation Controller (CMAC; Albus, 1975; Sutton, 1996) instead of a case-based memory to represent the Q-function. The CMAC partitions the state-action space with a set of overlapping tilings. Each tiling divides the space into a set of discrete *features* which maintain a value. When a query is made for a particular state-action pair, its *Q*-value is returned as the sum of the value in each tiling corresponding to the feature containing the query point. SARSA-CABA and SARSA-CMAC have both been applied to the pendulum swing-up task and the double-integrator task.

Adaptive Heuristic Critic (AHC; Anderson, 1987): uses a learning agent composed of two components: an *actor* (policy) and a *critic* (value-function), both of which are implemented using a feed-forward neural network trained with a variant of backpropagation.

The three value-function based methods (SARSA-CABA, SARSA-CMAC, and Q-MLP) each use a different kind of function approximator to represent a *Q*-function that can generalize across the continuous space of state-action pairs. Although these approximators can compute a value for any state-action pair, they do not implement true continuous control since the policy is not explicitly stored. Instead, continuous control is approximated by discretizing the action space at a resolution that is adequate for the problem. In order to select the optimal action *a* for a given state *s*, a *one-step* search in the action space is performed. The control agent selects actions according to an ε-*greedy* policy: with probability $1-\varepsilon$, $0 \leq \varepsilon < 1$, the action with the highest value is selected, and with probability ε, the action is random. This policy allows some exploration so that information can be gathered for all actions. In all simulations the controller was tested every 20 trials with ε=0 and learning turned off to determine whether a solution had been found.

5.2.2 PHYLOGENETIC METHODS

Symbiotic, Adaptive Neuro-Evolution (SANE; Moriarty, 1997) is a cooperative coevolutionary method that evolves two different populations simultaneously: a population of neurons and a population of *network blueprints* that specify how the neurons are combined to form complete networks. Each generation of networks is formed both using the blueprints and at random. Neurons that combine to form good networks receive high fitness, and are recombined in a

single population. Blueprints that result in favorable neuron combinations are also recombined to search for even better combinations.

Conventional Neuroevolution (CNE) is our implementation of single-population Neuroevolution similar to the algorithm used in Wieland (1991). In this approach, each chromosome in the population represents a complete neural network. CNE differs from Wieland's algorithm in that (1) the network weights are encoded with real instead of binary numbers, (2) it uses rank selection, and (3) it uses burst mutation. CNE is like ESP except that it evolves at the network level instead of the neuron level, and therefore provides a way to isolate the performance advantage of cooperative coevolution (ESP) over a single population approach (CNE).

Evolutionary Programming (EP; Saravanan and Fogel, 1995) is a general mutation-based evolutionary method that can be used to search the space of neural networks. Individuals are represented by two n-dimensional vectors (where n is the number of weights in the network): $\vec{x}$ contains the synaptic weight values for the network, and $\vec{\delta}$ is a vector of standard deviation values of $\vec{x}$. A network is constructed using the weights in $\vec{x}$, and offspring are produced by applying Gaussian noise to each element $\vec{x}(i)$ with standard deviation $\vec{\delta}(i), i \in \{1..n\}$.

Cellular Encoding (CE; Gruau et al., 1996a,b) uses Genetic Programming (GP; Koza, 1991) to evolve graph-rewriting programs. The programs control how neural networks are constructed out of "cells." A cell represents a neural network processing unit (neuron) with its input and output connections and a set of registers that contain synaptic weight values. A network is built through a sequence of operations that either copy cells or modify the contents of their registers. CE uses the standard GP crossover and mutation to recombine the programs allowing evolution to automatically determine an appropriate architecture for the task and relieve the investigator from this often trial-and-error undertaking.

Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES; Hansen and Ostermeier 2001) evolves the covariance matrix of the mutation operator in evolutionary strategies. The results in the pole-balancing domain were obtained from Igel (2003).

NeuroEvolution of Augmenting Topologies (NEAT; Stanley and Miikkulainen, 2002; Stanley 2004) is another NE method that evolves topology as well as synaptic weights, but unlike CE it uses a direct encoding. NEAT starts with a population of minimal networks (i.e., no hidden units) that can increase in complexity by adding either new connections or units through mutation. Every time a new gene appears, a *global innovation number* is incremented and assigned to that gene. Innovation numbers allow NEAT to keep track of the historical origin of every gene in the population so that (1) crossover can be performed between networks with different topologies, and (2) the networks can be grouped into "species" based on topological similarity.

Whenever two networks are recombined, the genes in both chromosomes with the same innovation numbers are lined up. Those genes that do not match are either *disjoint* or *excess*, depending on whether they occur within or outside the range of the other parent's innovation numbers, and are inherited from the more fit parent.

The number of disjoint and excess genes is used to measure the distance between genomes. Using this distance, the population is divided into species so that individuals compete primarily within their own species instead of with the population at large. This way, topological

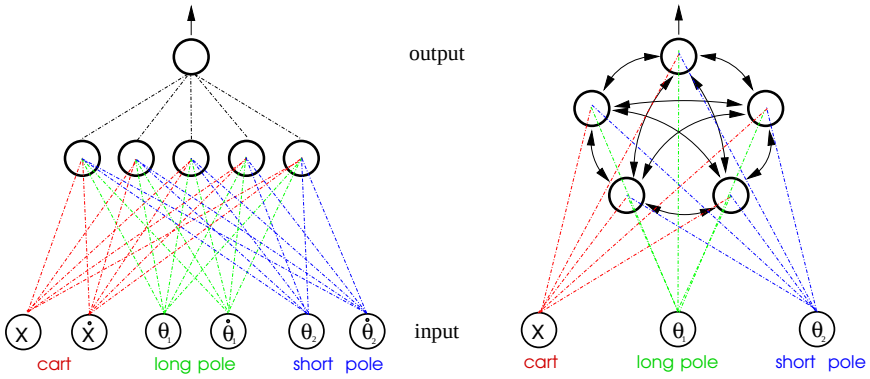


Figure 6: **Neural network control of the pole balancing system.** At each time step the network receives the current state of the cart-pole system $(x, \dot{x}, \theta_1, \dot{\theta_1}, \theta_2, \dot{\theta_2})$ through its input layer. For the feed-forward networks (a) used in the Markov tasks (1a and 2a), the input layer activation is propagated forward through the hidden layer of neurons to the output unit which indicates force to be applied to the cart. For the recurrent networks (b) used in the non-Markov tasks (1b and 2b), the neurons do not receive the velocities $(\dot{x}, \dot{\theta_1}, \dot{\theta_2})$, instead they must use their feedback connections to determine which direction the poles are moving. For the single pole version the network only has inputs for the cart and long pole.

innovations are protected and have time to optimize their structure before they have to compete with other species in the population.

Enforced SubPopulations (ESP; Gomez and Miikkulainen, 1997) is similar to SANE in that it uses cooperative coevolution at the neuron level, but, instead of using blueprints, the neuron population is split into disjoint subpopulations, one for each hidden unit in the network architecture being evolved. Instead of selecting neurons from a single population, as in SANE, to form networks, networks consist of one neuron from each subpopulation. During reproduction, neuron genotypes are only mated with members of their own subpopulation, and offspring remain in their parents' subpopulation.

For Q-MLP, SANE, CNE, ESP, and CoSyNE, experiments were run using our own code. For PGRL, AHC, SARSA, publicly available code from Grudic (2000), Anderson (1987), and Santamaria et al. (1998), was used respectively, modified for the pole-balancing domain. The parameter settings for each of these methods are listed in Appendix B. For VAPS, EP, CMA-ES, NEAT, and CE, the results were taken from the papers cited above. Data was not available for all methods on all tasks: however, in all such cases the method is shown to be significantly weaker already in a previous, easier task.

5.3 Task Setup

The pole balancing environment was implemented using a realistic physical model with friction, and fourth-order Runge-Kutta integration with a step size of 0.01s (see Appendix A for the equations of motion and parameters used). The state variables for the system are the following:

x : position of the cart.
$\dot{x}$: velocity of the cart.
θ_i : angle of the *i*-th pole $(i = 1, 2)$.
$\dot{\theta}_i$: angular velocity of the *i*-th pole.

Figure 6 shows how the network controllers interact with the pole balancing environment. At each time-step (0.02 seconds of simulated time) the network receives the state variable values scaled to [-1.0, 1.0]. This input activation is propagated through the network to produce a signal from the output unit that represents the amount of force used to push the cart. The force is then applied and the system transitions to the next state which becomes the new input to the controller. This cycle is repeated until a pole falls or the cart goes off the end of the track. In keeping with the setup in prior work (e.g., Wieland, 1991; Gruau et al., 1996b) we restrict the force to be no less than $\pm 1/256 \times 10$ Newtons so that the controllers cannot maintain the system in unstable equilibrium by outputting a force of zero when the poles are vertical.

The following four task configurations of increasing difficulty were used:

1. One Pole
   (a) Complete state information
   (b) Incomplete state information
2. Two Poles
   (a) Complete state information
   (b) Incomplete state information

Task 1a is the classic one-pole configuration. In 1b, the controller only has access to two of the four state variables: it does not receive the velocities $(\dot{x}, \dot{\theta})$. In 2a, the system now has a second pole next to the first, making the state-space 6-dimensional. Task 2b, like 1b, is non-Markov with the controller only seeing x, θ_1, and θ_2. Fitness was determined by the number of time steps a network could keep both poles within a specified failure angle from vertical and the cart between the ends of the track. The failure angle was 12° and 36° for the one and two pole tasks, respectively. For the one-pole tasks, the initial pole angle was set to 4.0° from vertical. For the two-pole tasks, the initial angle of the long pole was 4.0°, and the short pole was vertical. A task was considered solved if a network could do this for 100,000 time steps, which is equal to over 30 minutes in simulated time. CoSyNE evolved networks with one hidden unit, 20 weights per subpopulation for the 1-pole tasks, and 30 weights for the 2-pole tasks. Mutation was set to 0.3 for all experiments, which means that each weight in a new network have a 30% chance of being perturbed with Cauchy distributed noise. The initial weight range was $[-10, 10]$. All simulations were run on a 1.5GHz Intel Xeon.

5.4 Results: Balancing One Pole

Balancing one pole is a relatively easy problem that gives us a performance baseline before moving on to the much harder two-pole task. It has also been solved with many other methods and therefore serves to put the results in perspective with prior literature.

5.4.1 COMPLETE STATE INFORMATION

Table 1 shows the results for the single pole balancing task with complete state information. The results show that simply choosing weights at random (RWG) is sufficient to solve this task efficiently. CoSyNE was the only method that solved the task in fewer evaluations.

Method	Evaluations	CPU time (sec)
AHC	189,500	95
PGRL	28,779	1,163
Q-MLP	2,056	53
SARSA-CMAC	540	487
SARSA-CABA	965	1,713
RPG	(863)	—
CMA-ES	283	—
CNE	352	5
SANE	302	5
NEAT	743	7
ESP	289	4
RWG	199	2
CoSyNE	98	1

Table 1: **One pole with complete state information.** Comparison of various learning methods on the basic pole balancing problem with continuous control. Results for all methods are averages of 50 runs.

With the exception of RWG, there is a clear divide between the performance of the ontogenetic and phylogenetic methods, especially in terms of CPU time. For the value-based, ontogenetic methods, evaluating and updating values can be computationally expensive. The value-function approximator must be evaluated $O(|A|)$ times per state transition to determine the best action-value estimate, where A is a finite set of actions. Q-MLP and AHC have a notable CPU time advantage over SARSA because their value functions are represented compactly by neural networks which can be evaluated quickly, while the CMAC and case-based memory are coarse-codings have memory requirements and evaluation cost grow exponentially with the dimensionality of the state space.

In contrast, evolutionary methods do not update any agent parameters during interaction with the environment and only need to evaluate a function approximator once per state transition since the policy is represented explicitly.

PGRL is also quite slow as each update to the policy requires sampling $O(|A|T)$ trajectories, where T is the number of state transitions in the initial trajectory of each update. RPG performed best of the ontogenetic methods, but it must be noted that the criteria for success in the referenced work (Wierstra et al., 2007) was 10K steps instead of the 100K steps used with all the other methods (hence the parentheses in all tables for this method).

This task poses very little difficulty for the NE methods. However, NEAT required more than twice as many evaluations as CNE, SANE, and ESP because it explores different topologies that initially behave poorly and require time to develop. For this task the speciation process is an overkill—the task can be solved more efficiently by devoting resources to searching for weights only. All observed performance differences are statistically significant ($p < 0.01$) except between CNE, SANE and ESP.

Method	Evaluations	CPU time
VAPS	(500,000)	(5days)
SARSA-CABA	15,617	6,754
SARSA-CMAC	13,562	2,034
Q-MLP	11,331	340
RWG	8,557	3
RPG	(1,893)	—
NEAT	1,523	15
SANE	1,212	6
CNE	724	15
ESP	589	11
CoSyNE	127	2

Table 2: **One pole with incomplete state information.** The table shows the number of evaluations, CPU time, and success rate of the various methods. Results are the average of 50 simulations, and all differences are statistically significant ($p < 0.01$). The results for VAPS are in parenthesis since only a single unsuccessful run according to our criteria was reported by Meuleau et al. (1999).

5.4.2 INCOMPLETE STATE INFORMATION

This task is identical to the first task except the controller only senses the cart position x and pole angle θ. Therefore, the underlying states $\{x, \dot{x}, \theta, \dot{\theta}\}$ are hidden and the networks need to be recurrent so that the velocities can be computed internally using feedback connections. This makes the task significantly harder since it is more difficult to control the system when the concomitant problem of velocity calculation must also be solved. We were unable to solve this task with AHC and PGRL.

To allow Q-MLP and the SARSA methods to solve this task, we extended their inputs to include the immediately previous cart position, pole angle, and action $(x_{t-1}, \theta_{t-1}, a_{t-1})$ in addition to x_t, θ_t, and a_t. This *delay window* of depth 1 is sufficient to disambiguate process states (Lin and Mitchell, 1992). For VAPS, the state-space was partitioned into unequal intervals, 8 for x and 6 for θ, with the smaller intervals being near the center of the value ranges (Meuleau et al., 1999).

Table 2 compares the various methods in this task. The table shows the number of evaluations and average CPU time for the successful runs.

The results for VAPS are in parenthesis in the table because only a single run was reported by Meuleau et al. (1999). It is clear, however, that VAPS is the slowest method in this comparison, only being able to balance the pole for around 1 minute of simulated time after several days of computation (Meuleau et al., 1999). The evaluations and CPU time for the SARSA methods are the average of the successful runs only (out 29 of 50 for SARSA-CMAC and 35 out of 50 for SARSA-CABA). Of the value-function methods, Q-MLP fared the best, reliably solving the task and doing so much more rapidly than SARSA. Since both the CMAC and the cased-based memory are local function approximators, they require a dense sampling of the state space to obtain good value estimate. The MLP, being a global function approximator, is able to learn values for a whole set of states every time a state is updated. This property has been considered undesirable in some domains because updates at one state can disrupt or unlearn values at distant states. Because the

relatively simple form of the optimal value function for this task, the MLP accelerates learning by providing "useful" information about more of the state space on each update which is especially useful to bootstrap learning at the beginning when there is virtually no information about the value of most states. For more complicated value functions, the potential for instability in the MLP could give local representations the advantage (Boyan and Moore, 1995).

The performance of the five evolutionary methods degrades only slightly compared to the previous task. CoSyNE, CNE, and ESP were two orders of magnitude faster than VAPS and SARSA, one order of magnitude faster than Q-MLP, and approximately twice as fast as SANE and NEAT. CoSyNE was able to balance the pole for over 30 minutes of simulated time usually within 2 seconds of learning CPU time, and do so reliably.

The results on these first two tasks show that the single pole environment is not very challenging. A large part of the search space represents successful solutions, so that simply choosing points at random (i.e., RWG) can compete favorably with other ontogenetic approaches that start at one point and then must make relatively small incremental changes to reach a solution, without not getting stuck in a local minimum.

5.5 Results: Balancing Two Poles

The double pole problem is a better test environment for these methods, representing a significant jump in difficulty. Here the controller must balance two poles of different lengths (1m and 0.1m) simultaneously. The second pole adds two more dimensions to the state-space $(\theta_2, \dot{\theta}_2)$ and non-linear interactions between the poles.

5.5.1 COMPLETE STATE INFORMATION

For this task, CoSyNE was compared with Q-MLP, CNE, SANE, ESP, NEAT, and the published results of RPG, EP, and CMA-ES. Despite extensive experimentation with many different parameter settings, we were unable to get the SARSA methods to solve this task within 12 hours of computation.

Table 3 shows the results for the two-pole configuration with complete state information. Q-MLP compares very well to the NE methods with respect to evaluations, in fact, better than on task 1b, but again lags behind SANE, ESP and NEAT by nearly an order of magnitude in CPU time. ESP and NEAT are statistically even in terms of evaluations, requiring roughly three times fewer evaluations than SANE. In terms of CPU time, ESP has a slight but statistically significant ($p < 0.01$) advantage over NEAT. This is an interesting result because the two methods take such different approaches to evolving neural networks. NEAT is based on searching for an optimal topology, whereas ESP, like CoSyNE, optimizes a single, general topology (i.e., fully recurrent networks). At least in the difficult versions of the pole balancing task, the performance of these two approaches is very similar.

CMA-ES required the fewest number of evaluations, 59 less than CoSyNE on average, although we do not have the CMA-ES run data to test for statistical significance.

5.5.2 INCOMPLETE STATE INFORMATION

Although the previous task is difficult, the controller has the benefit of complete state information. In this task, as in task 1b, the controller does not have access to the velocities, that is, it does not know how fast or in which direction the poles are moving.

Method	Evaluations	CPU time
RWG	474,329	70
EP	307,200	—
CNE	22,100	73
SANE	12,600	37
Q-MLP	10,582	153
RPG	(4,981)	—
NEAT	3,600	31
ESP	3,800	22
CoSyNE	954	4
CMA-ES	895	—

Table 3: **Two poles with complete state information.** The table shows the number of pole balancing attempts (evaluations) and CPU time required by each method to solve the task. Evolutionary Programming data is taken from Saravanan and Fogel (1995), CMA-ES from Igel (2003). Q-MLP, CNE, SANE, NEAT, ESP, CoSyNE data are the average of 50 simulations, and all differences are statistically significant ($p < 0.01$) except the number of evaluations for NEAT and ESP.

Gruau et al. (1996b) were the first to tackle the two-pole problem without velocity information. Although they report the performance for only one simulation, we include their results to put the performance of the other methods in greater perspective. None of the value-function methods we tested made noticeable progress on the task after approximately 12 hours of computation. Therefore, in this task, only the evolutionary methods are compared.

To accommodate a comparison with CE, controllers were evolved using both the standard fitness function used in the previous tasks and also the "damping" fitness function used by Gruau et al. (1996b). The damping fitness is the weighted sum of two separate fitness measurements ($0.1 f_1 + 0.9 f_2$) taken over a simulation of 1000 time steps:

$$f_1 = t/1000,$$

$$f_2 = \begin{cases} 0 & \text{if } t < 100 \\ \left(\dfrac{0.75}{\sum_{i=t-100}^{t} (|x^i| + |\dot{x}^i| + |\theta_1^i| + |\dot{\theta}_1^i|)} \right) & \text{otherwise,} \end{cases}$$

where t is the number of time steps the poles were balanced out of the first 1000 steps. This complex fitness is intended to force the network to compute the pole velocities, and avoid solutions that balance the poles by merely swinging them back and forth (i.e., without calculating the velocities).

Table 4 compares the "surviving" methods for both fitness functions. To determine when the task was solved for the damping fitness function, the best controller from each generation was tested using the standard fitness to see if it could balance the poles for 100K time steps. The results for CE are in parenthesis in the table because only a single run was reported by Gruau et al. (1996b).

Using the damping fitness, CMA-ES, ESP, CNE, NEAT, and CoSyNE required two orders of magnitude fewer evaluations than CE. ESP was three times faster than CNE using either fitness

Method	Evaluations	
	Standard fitness	Damping fitness
RWG	415,209	1,232,296
CE	—	(840,000)
SANE	262,700	451,612
CNE	76,906	87,623
ESP	7,374	26,342
NEAT	—	6,929
RPG	(5,649)	—
CMA-ES	3,521	6,061
CoSyNE	1,249	3,416

Table 4: **Two poles with incomplete state information.** The table shows the number of evaluations for CNE, NEAT, and ESP using the standard fitness function (middle column), and using the damping fitness function (right column). Results are the average of 50 simulations for all methods except CE which is from a single run. All results are statistically significant ($p < 0.01$) .

function, with CNE failing to solve the task about 40% of the time, and NEAT, using small populations of size 16 (Stanley, 2004) performed nearly as well as CMA-ES (damping function). RPG was the only ontogenetic method to make significant progress in this task, again, however, only up to 10K time-steps of balancing.

On this most difficult task CoSyNE outperformed the next best method, CMA-ES, by a factor of two on both fitness functions.

6. Discussion

The results of the comparisons show that the phylogenetic methods (i.e., neuroevolution) are more efficient on this set of tasks than the ontogenetic methods. In the single pole tasks, the value-based ontogenetic methods were outperformed by random search. Our hope is that these results will help put an end to the use of this task for evaluating artificial learning systems. On the more difficult two-pole tasks, only Q-MLP was able to solve the completely observable version (task 2a), and none of the ontogenetic methods could solve the partially observable one (task 2b). In contrast, all of the neuroevolution methods scaled up to the most difficult tasks, with CMA-ES and CoSyNE leading the pack.

The most challenging of the tasks exhibit many of the dimensions of difficulty found in real world control problems: (1) continuous state and action spaces, (2) partial observability, and (3) non-linearity. The first two are problematic for value-based reinforcement learning methods because they either complicate the representation of the value function or the access to it. Neuroevolution deals with them by evolving recurrent networks; the networks can compactly represent arbitrary temporal, non-linear mappings. The success of CoSyNE on tasks of this complexity suggests that it can be applied to the control of real systems that manifest similar properties—specifically, non-linear, continuous systems such as aircraft control, satellite detumbling, and robot bipedal walking.

Other types of environments that are discrete or discontinuous, such as game-playing, job-shop scheduling, and resource allocation may be better served by other learning or optimization strategies.

The CoSyNE implementation used in this paper permuted all members of a subpopulation each generation. This means that it is possible for the networks evaluated in a given generation to not contain any combinations of weights found in the networks of the previous generation. While this maximizes the amount of exploration performed by sampling new networks, good weight combinations may be lost that could lead to a solution more efficiently. This aggressive exploration could become a problem for large networks, such as those that use very high-dimensional vision inputs. Future work will begin by investigating schemes for assigning permutation probabilities to weights (e.g., fitness proportional) in order to retain potential useful building blocks in the system and facilitate search in larger network spaces.

7. Conclusion

Reinforcement learning can in principle be used to control real world systems, but conventional methods scale poorly to large state-spaces and non-Markov environments. In this paper, we have shown that for a set of benchmark tasks that exhibit many of the key dimensions of difficulty found in real world control problems, neuroevolution in general, and CoSyNE in particular, can solve these problems much more reliably and efficiently than non-evolutionary reinforcement learning approaches.

Appendix A. Pole-balancing Equations

The equations of motion for N unjointed poles balanced on a single cart are

$$\ddot{x} = \frac{F - \mu_c sgn(\dot{x}) + \sum_{i=1}^{N} \tilde{F}_i}{M + \sum_{i=1}^{N} \tilde{m}_i},$$

$$\ddot{\theta}_i = -\frac{3}{4l_i}(\ddot{x}\cos\theta_i + g\sin\theta_i + \frac{\mu_{pi}\dot{\theta}_i}{m_i l_i}),$$

where $\tilde{F}_i$ is the effective force from the ith pole on the cart,

$$\tilde{F}_i = m_i l_i \dot{\theta}_i^2 \sin\theta_i + \frac{3}{4} m_i \cos\theta_i (\frac{\mu_{pi}\dot{\theta}_i}{m_i l_i} + g\sin\theta_i),$$

and $\tilde{m}_i$ is the effective mass of the ith pole,

$$\tilde{m}_i = m_i(1 - \frac{3}{4}\cos^2\theta_i).$$

Parameters used for the single pole problem:

Sym.	Description	Value
x	Position of cart on track	[-2.4,2.4] m
θ	Angle of pole from vertical	[-12,12] deg.
F	Force applied to cart	-10,10 N
l	Half length of pole	0.5m
M	Mass of cart	1.0 kg
m	Mass of pole	0.1 kg

Parameters for the double pole problem.

Sym.	Description	Value
x	Position of cart on track	[-2.4,2.4] m
θ	Angle of pole from vertical	[-36,36] deg.
F	Force applied to cart	[-10,10] N
l_i	Half length of ith pole	l_1 = 0.5m l_2 = 0.05m
M	Mass of cart	1.0 kg
m_i	Mass of ith pole	m_1 = 0.1 kg m_2 = 0.01 kg
μ_c	Coefficient of friction of cart on track	0.0005
μ_p	Coefficient of friction if ith pole's hinge	0.000002

Appendix B. Parameter Settings Used in Pole Balancing Comparisons

Below are the parameters used to obtain the results for Q-MLP, SARSA-CABA, SARSA-CMAC, CNE, SANE, ESP, and NEAT. The parameters for VAPS (Meuleau et al., 1999), RPG (Wierstra et al., 2007), CMA-ES (Igel, 2003), EP (Saravanan and Fogel, 1995), and CE2 (Gruau et al., 1996b) along with a detailed description of each method can be found in the cited papers.

Table 5 describes the parameters common to all of the value function methods.

Parameter	Description
ε	greediness of policy
α	learning rate
γ	discount rate
λ	eligibility

Table 5: All parameters have a range of (0,1).

Q-MLP

Parameter	Task		
	1a	1b	2a
ε	0.1	0.1	0.05
α	0.4	0.4	0.2
γ	0.9	0.9	0.9
λ	0	0	0

For all Q-MLP experiments the Q-function network had 10 hidden units and the action space was quantized into 26 possible actions: $\pm 0.1, 0.25, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$.

SARSA-MLP

Parameter	Task		
	1a	1b	2a
ε	0.1	0.1	0.05
α	0.4	0.4	0.1
γ	0.9	0.9	0.9
λ	0	0	0.3

For all Q-MLP experiments the Q-function network had 10 hidden units and the action space was quantized into 26 possible actions: $\pm 0.1, 0.25, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$.

SARSA-CABA

Parameter	Task	
	1a	1b
τ_d	0.03	0.03
τ_k^x	0.05	0.05
τ_k^u	0.1	0.1
ε	0.05	0.05
α	0.4	0.1
γ	0.99	0.99
λ	0.4	0.4

τ_d is the density threshold, τ_k^x and τ_k^u are the smoothing parameters for the input and output spaces, respectively. See Santamaria et al. (1998) for a more detailed description of the Case-Based Memory architecture.

SARSA-CMAC

Parameter	Task	
	1a	1b
ε	0.05	0.05
α	0.4	0.1
γ	0.9	0.9
λ	0.5	0.3
No. of tilings	45: 10 based on $x, \dot{x}, \theta_1$ 5 based on x, θ 5 based on $x, \dot{\theta}$ 5 based on $\dot{x}, \dot{\theta}$ 5 based on x 5 based on $\dot{x}$ 5 based on θ 5 based on $\dot{\theta}$	50 : 10 based on x_t, x_{t-1}, θ_t 10 based on $x, \theta_t, \theta_{t-1}$ 5 based on x_t, θ_t 5 based on x_{t-1}, θ_{t-1} 5 based on x_t 5 based on x_{t-1} 5 based on θ_t 5 based on θ_{t-1}

where x_t and θ_t are the cart position and pole angle at time t. Each variable was divided in to 10 intervals in each tiling. For a more complete explanation of the CMAC architecture see Santamaria et al. (1998).

SANE

Parameter	Task	
	1(a,b)	2(a,b)
no. of neurons	100	200
no. of blueprints	50	100
evals per generation	200	400
size of networks	5	7

The mutation rate for all runs was set to 10%.

CNE

Parameter	Task			
	1a	1b	2a	2b
no. of networks	200	200	400	1000
size of networks	5	5	5	rand [1..9]
burst threshold	10	10	10	15

The mutation rate for all runs was set to 40%. Burst threshold is the number of generations after which burst mutation is activated if the best network found so far is not improved upon. CNE evaluates each of the networks in its population once per generation.

ESP

Parameter	Task			
	1a	1b	2a	2b
network type	FF	FR	FF	FR
initial no. of subpops	5	5	5	5
size of subpopulations	20	20	40	100
evals per generation	200	200	400	1000
burst threshold	10	10	10	5

The mutation rate for all runs was set to 40%. Burst threshold is the number of generations after which burst mutation is activated if the best network found so far is not improved upon. FF denotes a feed-forward network, whereas FR denotes a fully recurrent network.

Acknowledgments

This research was partially funded by the following grants: NSF EIA-0303609, NSF IIS-0083776, THECB (Texas Higher Education Coordinating Board) ARP-003658-476-2001, and CSEM Alpnach and the EU MindRaces project: FP6 511931.

References

J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement, and Control*, 97(3):220–227, 1975.

C. W. Anderson. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, 9:31–37, 1989.

C. W. Anderson. Strategy learning with multilayer connectionist representations. Technical Report TR87-509.3, GTE Labs, Waltham, MA, 1987.

L. C. Baird and Andrew W. Moore. Gradient descent reinforcement learning. In *Advances in Neural Information Processing Systems 12*, 1999.

R. K. Belew, J. McInerney, and N. N. Schraudolph. Evolving networks: Using the genetic algorithm with connectionist learning. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Proceedings of the Workshop on Artificial Life (ALIFE '90)*. Reading, MA: Addison-Wesley, 1991. ISBN 0-201-52570-4.

J. A. Boyan and A. W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems 7*, 1995.

B. Bryant and R. Miikkulainen. Neuroevolution of adaptive teams: Learning heterogeneous behavior in homogeneous multi-agent systems. In *Congress in Evolutionary Computation, Canberra, Australia*, 2003.

P. J. Darwen. *Co-Evolutionary Learning by Automatic Modularization with Speciation*. PhD thesis, University College, University of New South Wales, November 1996.

R. Eriksson and B. Olsson. Cooperative coevolution in inventory control optimization. In *Proceedings of 3rd International Conference on Artificial Neural Networks and Genetic Algorithms*, 1997.

F. Gomez and R. Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342, 1997.

F. Gomez, D. Burger, and R. Miikkulainen. A neuroevolution method for dynamic resource allocation on a chip multiprocessor. In *Proceedings of the INNS-IEEE International Joint Conference on Neural Networks*, pages 2355–2361, Piscataway, NJ, 2001. IEEE.

F. J. Gomez. *Robust Nonlinear Control through Neuroevolution*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin, August 2003. Technical Report AI-TR-03-303.

F. J. Gomez and R. Miikkulainen. Active guidance for a finless rocket using neuroevolution. In E. Cant-Paz et al., editor, *Proceedings of the Genetic Evolutionary Computation Conference (GECCO-03)*. Springer-VerlagBerlin; New York, 2003.

D. Grady. The vision thing: Mainly in the brain. *Discover*, 14:57–66, June 1993.

U. Grasemann and R. Miikkulainen. Effective image compression using evolved wavelets. In *Proceedings of the Genetic Evolutionary Computation Conference (GECCO-05)*, New York, 2005. ACM. ISBN 1-59593-010-8.

B. Greer, H. Hakonen, R. Lahdelma, and R. Miikkulainen. Numerical optimization with neuroevolution. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*, 2002.

F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 81–89, Cambridge, MA, 1996a. MIT Press.

F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. Technical Report NC-TR-96-048, NeuroCOLT, 1996b.

G. Grudic. Simulation code for policy gradient reinforcement learning. http://www.cis.upenn.edu/ grudic/PGRLSim/, 2000.

N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

S. A. Harp, T. Samad, and A. Guha. Towards the genetic synthesis of neural networks. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 360–369, 1989.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern-Directed Inference Systems*. Academic Press, New York, 1978.

J. Horn, D. E. Goldberg, and K. Deb. Implicit niching in a learning classifier system: Nature's way. *Evolutionary Computation*, 2(1):37–66, 1994.

P. Husbands and F. Mill. Simulated co-evolution as the mechanism for emergent planning and scheduling. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 264–270. San Francisco, CA: Morgan Kaufmann, 1991. ISBN 1-55860-208-9.

C. Igel. Neuroevolution for reinforcement learning using evolution strategies. In R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon, editors, *Congress on Evolutionary Computation (CEC 2003)*, volume 4, pages 2588–2595. IEEE, 2003.

J.-S. R. Jang. Self-learning fuzzy controllers based on temporal backpropagation. *IEEE Transactions on Neural Networks*, 3(5):714–723, September 1992.

T. Jansen and R. P. Wiegand. The cooperative coevolutionary (1+1) ea. *Evolutionary Computation*, 12(4), 2004.

T. Jansen and R. P. Wiegand. Exploring the explorative advantage of the CC (1+1) ea. In E. Cant-Paz et al., editor, *Proceedings of the Genetic Evolutionary Computation Conference (GECCO-03)*. Springer-VerlagBerlin; New York, 2003.

D. Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, R. Korf, C. Taylor, and A. Wang. Evolution as a theme in artificial life: The Genesys/Tracker system. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Proceedings of the Workshop on Artificial Life (ALIFE '90)*. Reading, MA: Addison-Wesley, 1991. ISBN 0-201-52570-4.

H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476, 1990.

J. R. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1991.

L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning, and teaching. *Machine Learning*, 8(3):293–321, 1992.

L.-J. Lin and T. M. Mitchell. Memory approaches to reinforcement learning in non-Markovian domains. Technical Report CMU-CS-92-138, Carnegie Mellon University, School of Computer Science, May 1992.

A. Lubberts and R. Miikkulainen. Co-evolving a go-playing neural network. In *Coevolution: Turning Adaptive Algorithms Upon Themselves, Birds-of-a-Feather Workshop, Genetic and Evolutionary Computation Conference (GECCO-2001)*, 2001.

M. Mandischer. Representation and evolution of neural networks. In R.F. Albrecht, C.R. Reeves, and N.C. Steele, editors, *Proceedings of the Conference on Artificial Neural Nets and Genetic Algorithms at Innsbruck, Austria*, pages 643–649. Springer-Verlag, 1993.

N. Meuleau, L. Peshkin, K.-E. Kim, and L. P. Kaelbling. Learning finite state controllers for partially observable environments. In *15th International Conference of Uncertainty in AI*, 1999.

D. Michie and R. A. Chambers. BOXES: An experiment in adaptive control. In E. Dale and D. Michie, editors, *Machine Intelligence*. Oliver and Boyd, Edinburgh, UK, 1968.

G. Miller and D. Cliff. Co-evolution of pursuit and evasion i: Biological and game-theoretic foundations. Technical Report CSRP311, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 1994.

D. E. Moriarty. *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin, 1997. Technical Report UT-AI97-257.

S. Nolfi and D. Parisi. Learning to adapt to changing environments in evolving neural networks. Technical Report 95-15, Institute of Psychology, National Research Council, Rome, Italy, 1995.

L. Panait, S. Luke, and J. F. Harrison. Archive-based cooperative coevolutionary algorithms. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 345–352, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-186-4. doi: http://doi.acm.org/10.1145/1143997.1144060.

J. Paredis. Steps towards co-evolutionary classification neural networks. In R. A. Brooks and P. Maes, editors, *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV)*, pages 102–108. Cambridge, MA: MIT Press, 1994. ISBN 0-262-52190-3.

J. Paredis. Coevolutionary computation. *Artificial Life*, 2:355–375, 1995.

A. S. Perez-Bergquist. Applying ESP and region specialists to neuro-evolution for Go. Technical Report CSTR01-24, Department of Computer Sciences, The University of Texas at Austin, 2001.

J. B. Pollack, A. D. Blair, and M. Land. Coevolution of a backgammon player. In C. G. Langton and K. Shimohara, editors, *Proceedings of the 5th International Workshop on Artificial Life: Synthesis and Simulation of Living Systems (ALIFE-96)*. Cambridge, MA: MIT Press, 1996. ISBN 0-262-62111-8.

M. A. Potter and K. A. De Jong. Evolving neural networks with collaborative species. In *Proceedings of the 1995 Summer Computer Simulation Conference*, 1995.

C. D. Rosin. *Coevolutionary Search Among Adversaries*. PhD thesis, University of California, San Diego, San Diego, CA, 1997.

J. C. Santamaria, R. S. Sutton, and A. Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, 6(2):163–218, 1998.

N. Saravanan and D. B. Fogel. Evolving neural control systems. *IEEE Expert*, pages 23–27, June 1995.

K. O. Stanley. *Efficient Evolution of Neural Networks Through Complexification*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin, August 2004. Technical Report AI-TR-04-314.

K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127, 2002.

R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 1038–1044. Cambridge, MA: MIT Press, 1996.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998. ISBN 0-262-19398-1.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, volume 12, pages 1057–1063. MIT Press, 2000.

G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8:257–277, 1992.

H. M. Voigt, J. Born, and I. Santibanez-Koref. Evolutionary structuring of artificial neural networks. Technical report, Technical University Berlin, Bio- and Neuroinformatics Research Group, 1993.

C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

P. Werbos. Backpropagation through time: what does it do and how to do it. In *Proceedings of IEEE*, volume 78, pages 1550–1560, 1990.

B. A. Whitehead and T. D. Choate. Cooperative–competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE Transactions on Neural Networks*, 1995.

S. Whiteson, N. Kohl, R. Miikkulainen, and P. Stone. Evolving keepaway soccer players through task decomposition. In E. Cant-Paz et al., editor, *Proceedings of the Genetic Evolutionary Computation Conference (GECCO-03)*. Springer-VerlagBerlin; New York, 2003.

D. Whitley, S. Dominic, R. Das, and Charles W. Anderson. Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13:259–284, 1993.

R. P. Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University, Fall 2003.

R. P. Wiegand, W. C. Liles, and K. A. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In L. Spector et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1235–1242. San Francisco, CA: Morgan Kaufmann, 2001. ISBN 1-55860-774-9. URL `citeseer.ist.psu.edu/481900.html`.

A. Wieland. Evolving neural network controllers for unstable systems. In *Proceedings of the International Joint Conference on Neural Networks* (Seattle, WA), pages 667–673. Piscataway, NJ: IEEE, 1991.

D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber. Solving deep memory pomdps with recurrent policy gradients. In *International Conference on Artificial Neural Networks*, 2007.

B. Yamauchi and R. D. Beer. Integrating reactive, sequential, and learning behavior using dynamical neural networks. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, editors, *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 382–391. Cambridge, MA: MIT Press, 1994. ISBN 0-262-53122-4.

X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

Journal of Machine Learning Research 9 (2008) 967-991 Submitted 9/06; Revised 9/07; Published 5/08

Search for Additive Nonlinear Time Series Causal Models

Tianjiao Chu TIC19@PITT.EDU
Department of Obstetrics, Gynecology & Reproductive Sciences
University of Pittsburgh
204 Craft Ave., Room B409
Pittsburgh, PA 15213, USA

Clark Glymour CG09@ANDREW.CMU.EDU
Department of Philosophy
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Editor: Greg Ridgeway

Abstract

Pointwise consistent, feasible procedures for estimating contemporaneous linear causal structure from time series data have been developed using multiple conditional independence tests, but no such procedures are available for non-linear systems. We describe a feasible procedure for learning a class of non-linear time series structures, which we call additive non-linear time series. We show that for data generated from stationary models of this type, two classes of conditional independence relations among time series variables and their lags can be tested efficiently and consistently using tests based on additive model regression. Combining results of statistical tests for these two classes of conditional independence relations and the temporal structure of time series data, a new consistent model specification procedure is able to extract relatively detailed causal information. We investigate the finite sample behavior of the procedure through simulation, and illustrate the application of this method through analysis of the possible causal connections among four ocean indices. Several variants of the procedure are also discussed.

Keywords: conditional independence test, contemporaneous causation, additive model regression, Granger causality, ocean indices

1. Introduction

For stationary time series of four or more dimensions, Swanson and Granger (1997) proposed to determine contemporaneous causation—causal influences occurring more rapidly than the sampling interval of the time series data—by regressing each time series variable on all lags of all variables considered and using the residuals to test for vanishing partial correlations. Using search procedures for directed acyclic graphical linear models, in particular, the PC algorithm (Spirtes et al., 2000), Bessler et al. (2002), Demiralp and Hoover (2003), and Hoover (2005) generalized Swanson and Granger's procedure to allow specification searches for contemporaneous linear systems among all partial orderings of the dependencies among the variables. Moneta (2003) derived the correction needed for the fact that the correlations are obtained from residuals of a regression, and applied it to a set of cointegrated variables.

All these methods are designed for linear systems with joint Normal distributions, and allow neither unrecorded (latent) common causes nor feedbacks. One source of these limitations is the

search algorithm used by all of these procedures, PC, which is known to be consistent only in the absence of feedback relations and latent common causes. In principle, some of these difficulties can be met by replacing PC with related algorithms: the FCI algorithm (Spirtes et al., 2000), which allows latent variables, or an algorithm due to Richardson and Spirtes (1999) that allows linear feedback relations, though no algorithm is available that is consistent for search for linear causal models when both latent variables *and* feedback may be present.

More fundamentally, PC and related algorithms require conditional independence information about the random variables as input, and are therefore limited to distribution families for which conditional independence tests of arbitrary order are available, such as Multinomial and Normal distributions. (Another group of causal inference algorithms that are based on model scores, such as Bayesian posteriors, are unable to handle either latent variables *or* feedbacks, except under extra constraints (Silva et al., 2006; Drton et al., 2006). For *non-Gaussian* linear models with latent variables, independent component analysis based algorithms (Hoyer et al., 2006) could be more informative than PC and FCI.) Extending the PC and related algorithms based on conditional independence constraints to a larger class of systems that includes nonlinear continuous models requires more general conditional independence tests. We begin by considering some of the difficulties involved with finding such tests.

In theory, using nonparametric density estimation, we can test conditional independence for any set of random variables which have a joint density with respect to the Lebesgue measure. For example, let the joint density of $\{X,Y,Z\}$ be $f_{XYZ}(x,y,z)$, the joint density of $\{X,Z\}$ be $f_{XZ}(x,z)$, the joint density of $\{Y,Z\}$ be $f_{YZ}(y,z)$, and the marginal density of Z be $f_Z(z)$. We could test if X and Y are independent given Z by testing if the Hellinger distance between $f_{XYZ}(x,y,z)f_Z(z)$ and $f_{XZ}(x,z)f_{YZ}(y,z)$ is 0. For example, Su and White (2007) propose a conditional independence test for stationary time series satisfying certain conditions, based on a weighted Hellinger distance between $f_{X|YZ}(x;y,z)$ and $f_{X|Z}(x;z)$, where $f_{X|YZ}(x;y,z)$ and $f_{X|Z}(x;z)$ are densities of the conditional distributions of X given $\{Y,Z\}$ and Z respectively. However, this approach requires nonparametric density estimation of multivariate distributions, which is subject to the curse of dimensionality: as the number of variables increases, the data points become sparse rapidly in the space spanned by the variables.

Baek and Brock (1992) and Hiemstra and Jones (1994) proposed a nonparametric method intended for Granger causality testing of nonlinear time series. Consider a bivariate time series $\{X_t,Y_t\}$, $t = 1,\cdots$, let $\boldsymbol{X}_t^m = (X_t,\cdots,X_{t+m-1})$ for some m, they proposed to test if $\boldsymbol{X}_t^m$ and $\boldsymbol{Y}_{t-b}^b$ are independent given $\boldsymbol{X}_{t-a}^a$ by testing the following null hypothesis:

$$P\left(\|\boldsymbol{X}_t^m - \boldsymbol{X}_s^m\|_\infty < e \mid \|\boldsymbol{X}_{t-a}^a - \boldsymbol{X}_{s-a}^a\|_\infty < e, \|\boldsymbol{Y}_{t-b}^b - \boldsymbol{Y}_{s-b}^b\|_\infty < e\right)$$
$$= P\left(\|\boldsymbol{X}_t^m - \boldsymbol{X}_s^m\|_\infty < e \mid \|\boldsymbol{X}_{t-a}^a - \boldsymbol{X}_{s-a}^a\|_\infty < e\right).$$

Unfortunately, only under some specific conditions is the above null hypothesis equivalent to the hypothesis that $\boldsymbol{X}_t^m$ is independent of $\boldsymbol{Y}_{t-b}^b$ given $\boldsymbol{X}_{t-a}^a$ (Diks and Panchenko, 2006).

Bell et al. (1996) considered additive model regression (Hastie and Tibshirani, 1990) for conditional independence tests in their study of nonlinear Granger causality. An additive model assumes that the response variable Y is a linear combination of univariate smooth functions of predictors $\boldsymbol{X} = \{X_1,\cdots,X_p\}$ plus an independent error term. That is:

$$Y = \sum_{i=1}^{p} f_i(X_i) + \varepsilon \tag{1}$$

where it is possible that $f_i(X_i) = 0$ for some $i \in \{1, \cdots, p\}$. Assuming Equation (1), additive model regression could be used to test if the response variable Y and some predictors $\boldsymbol{X}_a \subseteq \boldsymbol{X}$ are independent conditional on the other predictors $\boldsymbol{X}_b = \boldsymbol{X} \setminus \boldsymbol{X}_a$, because Y is independent of $\boldsymbol{X}_a$ given $\boldsymbol{X}_b$ if and only if $\mathrm{E}[Y|\boldsymbol{X}]$ is constant in $\boldsymbol{X}_a$}.

Additive regression works well as a conditional independence test in the study of Granger causality when no contemporaneous causation is allowed among time series, because the only type of conditional independence relations to be tested is the one described above. For example, in Bell et al. (1996), two additive models were fitted: one model for estimating the conditional expectation of a variable X_{T+1} given its T lags $\{X_1, X_2, \cdots, X_T\}$, another for conditional expectation of X_{T+1} given $\{X_1, X_2, \cdots, X_T\}$ and $\{Y_1, Y_2, \cdots, Y_T\}$. The F test was used to compare these two regression models: if the test failed to reject the first model, X_{T+1} was judged independent of $\{Y_1, Y_2, \cdots, Y_T\}$ given $\{X_1, X_2, \cdots, X_T\}$.

However, the use of additive model regression as a general purpose nonlinear conditional independence test is problematic, even for variables that are known to be related via additive models. Generally speaking, it is not always valid to use additive model regression to test conditional independence relations other than those between the response variable and some predictors given the other predictors. First, in some cases, additive model regression may miss some conditional dependencies. Consider a causal system with two exogenous variables X_1 and X_2, and an endogenous variable Y such that $Y = X_1^2 + X_2^2 + \varepsilon_Y$, where X_1, X_2 and ε_Y are independent Gaussian variables. Although the predictors X_1 and X_2 are dependent given the response variable Y, the conditional expectation of X_1 given Y and X_2 estimated using additive model regression will be constant in X_2. Second, even worse, in some cases additive model regression may miss some conditional independencies. Consider a system with two exogenous variables X_1 and X_2, and five endogenous variables $W = X_1 + X_2 + \varepsilon_W$, $Y = W^2 + \varepsilon_Y$, $U = \log(X_1) + \varepsilon_U$, $V = \log(X_2) + \varepsilon_V$, and $Z = U + V + \varepsilon_Z$. Although the two response variables Y and Z are independent conditional on the predictors X_1 and X_2, Z will be present in the conditional expectation of Y given $\{X_1, X_2, Z\}$ estimated by additive model regression. (Note that Y contains a term $2X_1X_2$, and $e^Z = e^{\varepsilon_U + \varepsilon_V + \varepsilon_Z} X_1 X_2$.)

Nevertheless, additive model regression has some very attractive features. First, and probably most importantly, it is not subject to the curse of dimensionality. In fact, Stone (1985) shows that the rate of convergence for an additive model regression is the same as that for a univariate smoother, which is much faster than a general multidimensional nonparametric regression method. The second major advantage of additive model regression is that it is possible to identify the contribution of each predictor to the response variable, thus allowing an intuitive interpretation of the fitted models.

In the following sections, we define a additive non-linear time series model by imposing linear constraints only among contemporaneous variables. We show that two families of conditional independence relations can be tested consistently among variables in a additive non-linear time series model using additive model regression. That is, asymptotically, additive model regression will neither miss any conditional independence relations nor report any spurious conditional independence relations when applied to data generated from a additive non-linear time series model to test those two families of conditional independence relations. We propose an inference procedure for nonlinear time series data that requires only information about these two families of conditional independence relations.

2. Additive Non-linear Time Series Models

Below we present the definition of a family of nonlinear time series models for which additive model regression based conditional independence test is possible. Here $\boldsymbol{X}_t$ is a p dimensional observed time series, $\boldsymbol{U}_t$ a q dimensional unobserved time series, and $\boldsymbol{\varepsilon}_t$ a p dimensional white noise.
Definition: A p dimensional time series $\{\boldsymbol{X}\}_t = \{\cdots, \boldsymbol{X}_1, \cdots, \boldsymbol{X}_T, \cdots\}$, where $\boldsymbol{X}_t = \{X_{t,1}, \cdots, X_{t,p}\}$, is generated from a *lag T additive non-linear model* if it satisfies the following conditions:

C1 For $i = 1, \cdots, p$,

$$X_{t,i} = \sum_{1 \le j \le p, j \ne i} c_{j,i} X_{t,j} + \sum_{1 \le k \le p, 1 \le l \le T} f_{k,i,l}(X_{t-l,k}) + \sum_{m=1}^{q} b_{m,i} U_{t,m} + \varepsilon_{t,i} \tag{2}$$

where $b_{m,i}$'s and $c_{j,i}$'s are constants, and $f_{k,i,l}$'s are smooth univariate functions

C2 $\cdots, \varepsilon_{1,1}, \cdots, \varepsilon_{1,p}, \varepsilon_{2,1}, \cdots, \varepsilon_{t,i}, \cdots$ and $\cdots, U_{1,1}, \cdots, U_{1,q}, U_{2,1}, \cdots, U_{t,j}, \cdots$ are jointly independent, with $\varepsilon_{t,i} \sim N(0, \sigma^2_{1,i})$ and $U_{t,j} \sim N(0, \sigma^2_{2,j})$.

C3 There is a k and an i such that $f_{k,i,T}(\cdot) \ne 0$

C4 There is no sequence of indices $\{j_1, j_2, \cdots, j_m\}$ such that c_{j_1,j_2}, c_{j_2,j_3}, $\cdots$, c_{j_{m-1},j_m}, c_{j_m,j_1} are all nonzero.

The model is additive because Equation (2) includes both linear terms and arbitrary smooth terms. It is also recursive in the sense that given an initialization of $\boldsymbol{X}_{t-T}, \cdots, \boldsymbol{X}_{t-1}$, all the later points in the time series, starting from $\boldsymbol{X}_t$, can be generated inductively.

A additive non-linear model can be causally interpreted in the following way:

- $X_{t,j}$ is a direct cause of $X_{t,i}$ if and only if $c_{j,i} \ne 0$ in Equation (2), (for the definition of *direct cause*, see Spirtes et al., 2000; Pearl, 2000);
- $X_{t-l,j}$ is a direct cause of $X_{t,i}$ if and only if $f_{j,i,l}(\cdot) \ne 0$ in Equation (2);
- Latent common causes are allowed only for variables in the same time tier, and $X_{t,i}$ and $X_{t,j}$ have a latent common cause $U_{t,m}$ if and only if there is an m such that $b_{m,i} b_{m,j} \ne 0$.

  Note that both $\boldsymbol{U}_t$ and $\boldsymbol{\varepsilon}_t$ are multi-dimensional Gaussian white noise and both are unobserved. However, for $i = 1, \cdots, p$, $\varepsilon_{t,i}$ can only be a direct cause of $X_{t,i}$, where for $m = 1, \cdots, q$, $U_{t,m}$ can be a direct cause of several variables in $\boldsymbol{X}_t$.
- Condition C4 means that no contemporaneous feedback is allowed. If condition C4 is violated, X_{t,j_m} would be a direct cause of X_{t,j_1}, while at the same time X_{t,j_1} would be a (possibly indirect) cause of X_{t,j_m}.

  Note that using results of Richardson and Spirtes (1999) the method described in Section 3 can be modified to allow contemporaneous feedback.

A additive non-linear model can be represented by a directed graph consisting of nodes for $X_{T+1,1}, \cdots, X_{T+1,p}$ and their direct causes, and directed edges between nodes for the direct influences between the corresponding variables. We call this graph a *unit causal graph* for the corresponding

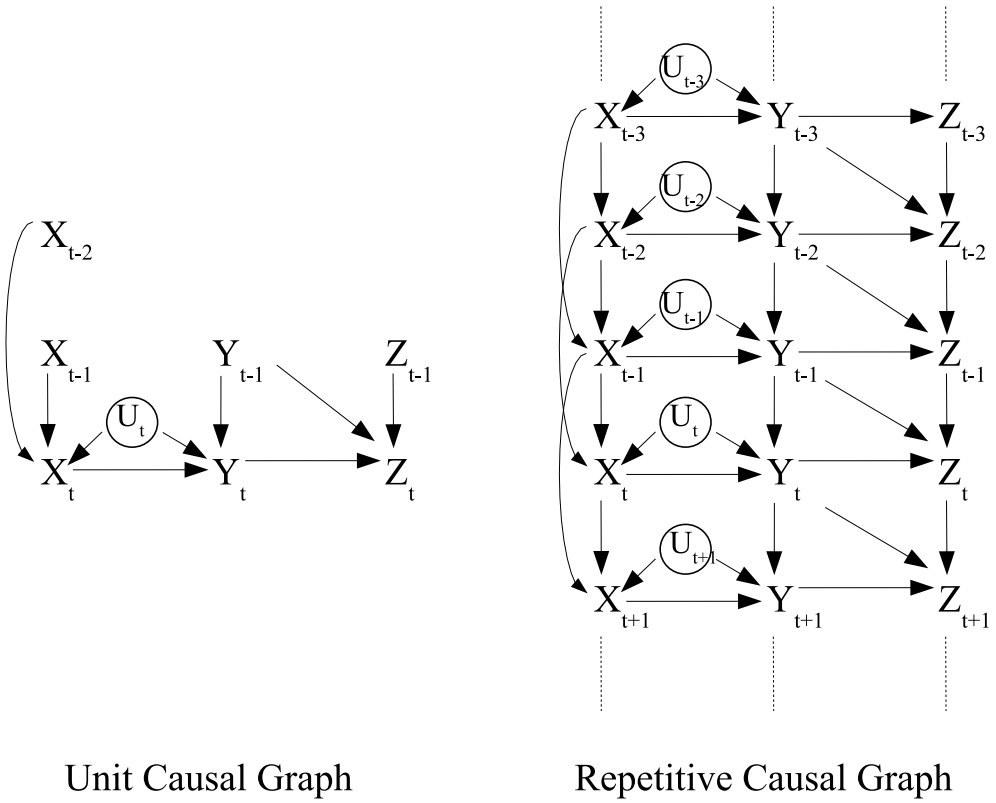


Figure 1: Unit causal graph and repetitive causal graph

time series. A unit causal graph can be extended to a *repetitive causal graph* by including all the variables in $\boldsymbol{X}_1, \cdots, \boldsymbol{X}_{T+1}$. Moreover, if there is an edge between $X_{T+1,i}$ and $X_{t,j}$, where $1 \leq t \leq T+1$, then similar edges will be added between $X_{T+1-l,j}$ and $X_{t-l,i}$ for $1 \leq l \leq t-1$. Figure 1 shows a unit causal graph and a segment of the corresponding repetitive graph. (The circled variables are latent variables.) In the remaining part of this paper, all time series causal models are represented by unit causal graphs.

Additive non-linear time series models make it possible to use the additive regression method, which is not subject to the curse of dimensionality, to test conditional independence for nonlinear time series. For a time series $\{\boldsymbol{X}\}_t$ generated from a lag T additive non-linear model, the following holds:

Proposition 1: Let X_t^1 and X_t^2 be any two distinct entries of random vector $\boldsymbol{X}_t$, $\boldsymbol{X}_t^c$ any subset, possibly empty, of $\boldsymbol{X}_t \setminus \{X_t^1, X_t^2\}$, and $\boldsymbol{X}_t^d$ any subset, possibly empty, of $\boldsymbol{X}_t \setminus \{X_t^1\}$. Let $\boldsymbol{X}^l = \{\boldsymbol{X}_{t-T}, \cdots, \boldsymbol{X}_{t-1}\}$, and $\boldsymbol{X}^e = \boldsymbol{X}^l \setminus \{X_{t-i,j}\}$ for some $X_{t-i,j} \in \boldsymbol{X}^l$.

- For any $\boldsymbol{x}_t^d$ and $\boldsymbol{x}^l$, conditional on $\boldsymbol{X}_t^d = \boldsymbol{x}_t^d$ and $\boldsymbol{X}^l = \boldsymbol{x}^l$, X_t^1 has a normal distribution $N(\mu_{1|a}, \sigma_{1|a}^2)$ such that $\mu_{1|a}$ is a linear combination of $\boldsymbol{x}_t^d$ and smooth univariate functions of entries of $\boldsymbol{x}^l$, and $\sigma_{1|a}$ is independent of t, $\boldsymbol{x}^l$ and $\boldsymbol{x}_t^d$. Thus, X_t^1 is independent of $X_{t-i,j}$ conditional on $\boldsymbol{X}^e$ and $\boldsymbol{X}_t^d$ if and only if $\mu_{1|a}$, the conditional expectation of X_t^1 given $\boldsymbol{X}_t^d = \boldsymbol{x}_t^d$ and $\boldsymbol{X}^l = \boldsymbol{x}^l$, is constant in $x_{l,j}$.

- For any x_t^2, $\boldsymbol{x}_t^c$ and $\boldsymbol{x}^l$, conditional on $X_t^2 = x_t^2$, $\boldsymbol{X}_t^c = \boldsymbol{x}_t^c$, and $\boldsymbol{X}^l = \boldsymbol{x}^l$, X_t^1 has a normal distribution $N(\mu_{1|b}, \sigma_{1|b}^2)$ such that $\mu_{1|b}$ is a linear combination of x_t^2, $\boldsymbol{x}_t^c$, and smooth univariate functions of entries of $\boldsymbol{x}^l$, and $\sigma_{1|b}$ is constant in t, x_t^2, $\boldsymbol{x}_t^c$, and $\boldsymbol{x}^l$. Thus, X_t^1 is independent of X_t^2 conditional on $\boldsymbol{X}_t^c$ and $\boldsymbol{X}^l$ if and only if, $\mu_{1|b}$, the conditional expectation of X_t^1 given $X_t^2 = x_t^2$, $\boldsymbol{X}_t^c = \boldsymbol{x}_t^c$, and $\boldsymbol{X}^l = \boldsymbol{x}^l$, is constant in x_t^2.

Proposition 1 implies that it is possible to use additive model regression to test the following two types of conditional independence relations among variables in a additive non-linear model. First, we can test if X_t^1 and X_t^2 are independent conditional on $\boldsymbol{X}_t^c$ and $\boldsymbol{X}^l$ by estimating the conditional expectation of X_t^1 given $\{X_t^2\} \cup \boldsymbol{X}_t^c \cup \boldsymbol{X}^l$ using additive model regression, and check if X_t^2 is a significant predictor for X_t^1 using statistical tests such as the F test (Bell et al., 1996) or the BIC scores (Huang and Yang, 2004). Similarly, if $X_{t-i,j}$ is not a significant predictor for X_t^1 in the additive model regression of X_t^1 against $\boldsymbol{X}^l$ and $\boldsymbol{X}_t^d$, we would say X_t^1 and $X_{t-i,j}$ are independent conditional on $\boldsymbol{X}_t^d$ and $\boldsymbol{X}^e$.

To make the above tests valid, we also need the assumption that additive model regression is an (asymptotically) consistent estimator of conditional expectations such as $\mathrm{E}[X_t^1|\boldsymbol{X}_t^d, \boldsymbol{X}^l]$ and $\mathrm{E}[X_t^1|X_t^2, \boldsymbol{X}_t^c, \boldsymbol{X}^l]$. Fortunately, it has been shown that, given a stationary nonlinear time series $\{\boldsymbol{X}\}_t$, nonparametric estimation of the conditional mean $\mathrm{E}[X_t|\boldsymbol{X}_{t-1}, \cdots, \boldsymbol{X}_{t-T}]$ is asymptotically consistent and/or asymptotically normal, provided certain conditions are satisfied (Robinson, 1983; Truong and Stone, 1992; Chen and Tsay, 1993; Tjøstheim and Auestad, 1994; Härdle et al., 1997; Cai and Masry, 2000; Huang and Yang, 2004). Generally speaking, besides some regularity conditions on the density of $\boldsymbol{X}_t \cup \boldsymbol{X}^l$ and smoothness condition on $\mathrm{E}[X_t|\boldsymbol{X}^l]$, $\{\boldsymbol{X}\}_t$ should satisfy some form of α mixing condition. $\{\boldsymbol{X}\}_t$ is α mixing if for some $\alpha(n) \to 0$,

$$\sup\{|P(A \cap B) - P(A)P(B)| : A \in \mathcal{F}_t, B \in \mathcal{G}_{n+t}\} \leq \alpha(n)$$

where $\mathcal{F}_t$ is the σ-field generated by $\boldsymbol{X}_t, \boldsymbol{X}_{t-1}, \cdots$, and $\mathcal{G}_{n+t}$ the σ-field generated by $\boldsymbol{X}_{t+n}, \boldsymbol{X}_{t+n+1}, \cdots$.

A concept closely related to α mixing is geometric ergodicity. A stationary time series $\{\boldsymbol{X}\}_t$ is geometrically ergodic if there is a function $M(x) < \infty$ and a constant $\rho < 1$ such that for all x:

$$\sup_A |P(X_n \in A|X_0 = x) - \pi(A)| \leq M(x)\rho^n$$

where π is the stationary distribution of $\{\boldsymbol{X}\}_t$. For stationary time series, geometric ergodicity implies α mixing for an $\alpha(n)$ of exponential rate (Davydov, 1973). Sufficient conditions for a nonlinear time series to be geometrically ergodic can be found in Chan and Tong (1994), An and Huang (1996), and Cline and Pu (1999). In particular, Xia and An (1999) provides a set of sufficient conditions for the geometric ergodicity of time series generated by projection pursuit models, of which our additive non-linear model is a special case.

3. A Causal Inference Algorithm

Consider a time series $\{\boldsymbol{X}\}_t = \{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_t, \cdots\}$ are generated from a lag T additive non-linear model. Let $\boldsymbol{X}^l = \{\boldsymbol{X}_{t-1}, \cdots, \boldsymbol{X}_{t-T}\}$, X_t^1 and X_t^2 be any two entries of $\boldsymbol{X}_t$, $\boldsymbol{X}_t^b$ be any subset, possibly empty, of $\boldsymbol{X}_t \setminus \{X_t^1\}$, $\boldsymbol{X}_t^c$ be any subset, possibly empty, of $\boldsymbol{X}_t \setminus \{X_t^1, X_t^2\}$, $X_{t-i,j}$ any variable in $\boldsymbol{X}^l$, and $\boldsymbol{X}^e = \boldsymbol{X}^l \setminus \{X_{t-i,j}\}$. Using additive model regression, we can test two types of conditional independence relations: 1), if X_t^1 and X_t^2 are independent given $\boldsymbol{X}_t^c$ and $\boldsymbol{X}^l$, and 2), if X_t^1 and $X_{t-i,j}$ are independent given $\boldsymbol{X}_t^b$ and $\boldsymbol{X}^e$. These pieces of information are not generally sufficient for currently available causal inference algorithms, such as the PC and FCI, to be informative: these procedures require (in the worst case) complete conditional independence information. However, starting from the same principle behind the PC and FCI algorithms, we describe a procedure that requires only these two types of conditional independence information. The procedure, which is capable of producing very informative causal structures, takes advantage of the constraints on possible causal relations among the random variables imposed by additive non-linear models, for example, $X_{t_2,k}$ cannot be a cause of $X_{t_1,j}$ if $t_1 < t_2$, no latent common cause exists for $X_{t_2,k}$ and $X_{t_1,j}$ if $t_1 \neq t_2$, etc.

The following propositions are needed to justify our procedure. We assume familiarity with notions from the graphical modeling literature, including the notion of d-separation (Pearl, 2000), and faithfulness (Spirtes et al., 2000). In summary:

Formally a causal graph G is defined as an ordered pair $\langle \boldsymbol{V}, \boldsymbol{E} \rangle$, where $\boldsymbol{V}$ is the set of variables in G, and $\boldsymbol{E}$ the set of edges in G. An edge e in $\boldsymbol{E}$ is again defined as an ordered pair $\langle V_i, V_j \rangle$, where V_i and V_j are two variables in $\boldsymbol{V}$. Given an edge $e = \langle V_i, V_j \rangle$ in graph G, we say that V_i is a direct cause of V_j in G. The *subgraph G_m induced by $\boldsymbol{V}_m$*, where $\boldsymbol{V}_m$ is a subset of $\boldsymbol{V}$, is defined as an ordered pair $\langle \boldsymbol{V}_m, \boldsymbol{E}_n \rangle$ such that an edge $e = \langle V_i, V_j \rangle$ is in $\boldsymbol{E}_n$ if and only if e is in $\boldsymbol{E}$ and the two variables $\{V_i, V_j\}$ are both in $\boldsymbol{V}_m$. A vertex is a collider on an undirected path in a directed acyclic graph (DAG) if and only if it is the second member of both of two edges on the path, that is, two edges on the path are directed into it. Two vertices X, Y (representing random variables) are d-separated with respect to a set $\boldsymbol{Z}$ of vertices if and only if every undirected path between the variables contains a collider having no directed path into a member of $\boldsymbol{Z}$ or contains a non-collider that is a member of $\boldsymbol{Z}$. A joint distribution on the variables (vertices) of a DAG is faithful if and only if all conditional independence relations follow from the d-separation property applied to the DAG.

In the three propositions below, $\{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_t, \cdots\}$ form a time series generated from a lag T additive non-linear model, $\boldsymbol{X}^l = \{\boldsymbol{X}_{t-1}, \cdots, \boldsymbol{X}_{t-T}\}$, X_t^1 and X_t^2 are any two entries of $\boldsymbol{X}_t$, and $\boldsymbol{X}^e = \boldsymbol{X}^l \setminus \{X_{t-i,j}\}$ for some $X_{t-i,j} \in \boldsymbol{X}^l$

Proposition 2: The d-separation relations among the variables in $\boldsymbol{X}_t$ conditional on $\boldsymbol{X}^l$ in a repetitive causal graph G_c are the same as the d-separation relations among the variables in $\boldsymbol{X}_t$ in the subgraph of G_c induced by $\boldsymbol{X}_t$.

Proof: See Moneta (2003), proposition 4. □

Proposition 3: Consider a time series $\{\boldsymbol{X}\}_t = \{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_t, \cdots\}$ generated from a lag T additive non-linear model. Let $\boldsymbol{X}^l = \{\boldsymbol{X}_{t-1}, \cdots, \boldsymbol{X}_{t-T}\}$, X_t^1 and X_t^2 be any two entries of $\boldsymbol{X}_t$. Assuming faithfulness, if there is a variable $X_{t-i,j} \in \boldsymbol{X}^l$ such that X_t^2 and $X_{t-i,j}$ are independent conditional on $\boldsymbol{X}^e = \boldsymbol{X}^l \setminus \{X_{t-i,j}\}$, but $X_{t-i,j}$ and X_t^1 are not independent conditional on $\boldsymbol{X}^e$, then X_t^1 is not a cause of X_t^2.

Proof: Suppose X_t^1 is a cause of X_t^2, then there must be a directed path P' from X_t^1 to X_t^2 such that each vertex on P' is in $\boldsymbol{X}_t$. If $X_{t-i,j}$ and X_t^1 are dependent given $\boldsymbol{X}^e$, there must be a path P d-connecting $X_{t-i,j}$ and X_t^1 given $\boldsymbol{X}^e$. Thus, no variable in $\boldsymbol{X}^e$ is a non-collider on path P, and all the colliders on path P must be observed ancestors of $\boldsymbol{X}^e$, hence must be in $\boldsymbol{X}^e$. (Note that the set of observed ancestors of $\boldsymbol{X}^e$ is either $\boldsymbol{X}^e$ or $\boldsymbol{X}^e \cup \{X_{t-i,j}\}$). This implies that P must be into X_t^1, because otherwise either P would be a direct path from X_t^1 to $X_{t-i,j}$, which is not allowed, or there must be a collider on P that is both a descendant of X_t^1 and an element of $\boldsymbol{X}^e$, which also is impossible. By appending the direct path P' to P, we get a path d-connecting $X_{t-i,j}$ and X_t^2 given $\boldsymbol{X}^e$, which is a contradiction. □

Proposition 4: Consider a time series $\{\boldsymbol{X}\}_t = \{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_t, \cdots\}$ generated from a lag T additive non-linear model. Let $\boldsymbol{X}^l = \{\boldsymbol{X}_{t-1}, \cdots, \boldsymbol{X}_{t-T}\}$, X_t^1 be any entry of $\boldsymbol{X}_t$, $X_{t-i,j}$ be any variable in $\boldsymbol{X}^l$, $\boldsymbol{X}_t^d$ be the set of all observed contemporary direct causes of X_t^1, and $\boldsymbol{X}^e = \boldsymbol{X}^l \setminus \{X_{t-i,j}\}$. Assuming faithfulness, $X_{t-i,j}$ and X_t^1 are dependent conditional on $\boldsymbol{X}_t^d$ and $\boldsymbol{X}^e$ if and only if:

- either $X_{t-i,j}$ is a direct cause of X_t^1,
- or there is a path P between X_t^1 and $X_{t-i,j}$, with $\langle W_1, \cdots, W_m \rangle$ being the set of observed variables on P between X_t^1 and $X_{t-i,j}$ and ordered along the direction from X_t^1 to $X_{t-i,j}$, such that:
  1. $W_i \in \boldsymbol{X}_t$ for $i = 1, \cdots, m$;
  2. X_t^1 and W_1 have a latent common cause;
  3. if $W_i \in \boldsymbol{X}_t^d$ then W_i is a collider on P;
  4. W_i is a (possibly indirect) cause of X_t^1 for $i = 1, \cdots, m$;
  5. $X_{t-i,j}$ is a direct cause of W_m.

Proof: The *if* part of the proposition is trivial, here we only prove the *only if* part.

Suppose $X_{t-i,j}$ is not a direct cause of X_t^1, then there is a path P d-connecting $X_{t-i,j}$ and X_t^1 conditional on $\boldsymbol{X}_t^d$ and $\boldsymbol{X}^e$. Let $\boldsymbol{W} = \langle W_1, \cdots, W_m \rangle$ be the set of observed variables on P between X_t^1 and $X_{t-i,j}$, ordered along the direction from X_t^1 to $X_{t-i,j}$.

To show that $W_i \in \boldsymbol{X}_t$ for $i = 1, \cdots, m$, we note that if W_j is the first element in $\boldsymbol{W}$ such that $W_j \notin \boldsymbol{X}_t$, it must belong to $\boldsymbol{X}^e$, where W_{j-1} is in $\boldsymbol{X}_t$. Because there is no observed variable between W_{j-1} and W_j on P, by the definition of additive non-linear models, there must be a direct edge from W_j to W_{j-1} on P (let $X_t^1 = W_0$ when $j = 1$). This means that W_j is not a collider on P, hence P cannot d-connect X_t^1 and $X_{t-i,j}$ conditional on $\boldsymbol{X}^e$ and $\boldsymbol{X}_t^d$, which contradicts our assumption. Using the same argument, given that $W_m \in \boldsymbol{X}_t$, it is easy to see that $X_{t-i,j}$ must be a direct cause of W_m.

Next we show that W_1 and X_t^1 must have a latent common cause. Assume that there is no latent common cause for W_1 and X_t^1. Because there is no observed variable between W_1 and X_t^1 on P, they must be adjacent on P, hence there must be a direct causal relation between X_t^1 and W_1. Consider the two alternative cases:

- First, suppose that W_1 is a direct cause of X_t^1. Then $W_1 \in \boldsymbol{X}_t^d$, and is a non-collider on P, hence P cannot d-connecting $X_{t-i,j}$ and X_t^1 conditional on $\boldsymbol{X}_t^d$ and $\boldsymbol{X}^e$.

- Second, suppose X_t^1 is a direct cause of W_1. Then there must be a variable W_i for some $i \geq 1$ such that the subpath $\{X_t^1, W_1, \cdots, W_i\}$ of P is a directed path from X_t^1 to W_i, and W_i is a collider on P. This would imply that W_i has to be a cause of X_t^1, for otherwise neither W_i nor any of its descendants belong to $\boldsymbol{X}_t^d$, which means that P cannot d-connect X_t^1 and $X_{t-i,j}$ conditional on $\boldsymbol{X}^e$ and $\boldsymbol{X}_t^d$. But allowing W_i to be a cause of X_t^1 would make the path $X_t^1, W_1, \cdots, W_i, X_t^1$ a directed cycle, which is impossible.

It is obvious that if $W_i \in \boldsymbol{X}_t^d$, then it must be a collider on P. To show that W_i is a cause of X_t^1, we note that if W_j is a collider on P, it must be a cause of X_t^1, for otherwise neither W_j nor any of its descendants belongs to $\boldsymbol{X}_t^d$, hence P cannot d-connect X_t^1 and $X_{t-i,j}$ conditional on $\boldsymbol{X}^e$ and $\boldsymbol{X}_t^d$. Therefore W_i must be a cause of X_t^1, because it is either a collider on P, or a cause of a collider on P. □

Given propositions 2, 3, and 4, we propose a three-step procedure for inference to unit causal graphs from time series data generated by additive non-linear models. The output of this causal inference procedure is a Partial Ancestral Graph (PAG). Roughly speaking, a PAG is a graph consisting of a list of vertices representing observed random variables, and 3 types of end points, $-$, $\circ$, and $>$, which are combined to form the following 4 types of edges representing causal relations between random variables.

- $X \to Y$ means that X is a (possibly indirect) cause of Y.
- $X \leftrightarrow Y$ means that there is a latent variable Z that is a (possibly indirect) cause of both X and Y.
- $X \circ\!\!\to Y$ means either $X \to Y$ or $X \leftrightarrow Y$.
- $X \circ\!\!-\!\!\circ Y$ means either $X \to Y$, or $Y \circ\!\!\to X$. In other words, $X \circ\!\!-\!\!\circ Y$ means that X and Y cannot be d-separated by any other observed variables.

For detailed explanation of PAGs, see Spirtes et al. (2000). Following Spirtes et al. (2000), we also use * as a meta symbol to represent any of the three end points.

Below is a constraint based additive non-linear time series causal inference procedure for non-linear time series with latent common causes. The conditional independence information required by the procedure can be obtained using additive model regression based conditional independence tests mentioned in the previous section. Here we assume that the time series data satisfies various conditions for the asymptotic consistency and normality of the additive model estimator, and that an upper bound T_{max} on the unknown true lag number T of the additive non-linear model has been set, either using the procedures in Tjøstheim and Auestad (1994) or Huang and Yang (2004), or based on background knowledge. So long as T_{max} is no less than T, the following procedure asymptotically obtains a correct PAG. Of course, choosing a T_{max} much higher than T will reduce the efficiency of the procedure.

The symbols in the following procedure are defined in the same way as in the beginning of this section, except that $\boldsymbol{X}^l$ is redefined as $\boldsymbol{X}^l = \{\boldsymbol{X}_{t-1}, \cdots, \boldsymbol{X}_{t-T_{max}}\}$.

1. Identify contemporary causal relations

   (a) For all choices of X_t^1, X_t^2, and $\boldsymbol{X}_t^c$, determine if X_t^1 is independent of X_t^2 conditional on $\boldsymbol{X}_t^c$ and $\boldsymbol{X}^l$.

(b) Treat the above conditional independence relations as if they were conditional independence relations between X_t^1 and X_t^2 given $\boldsymbol{X}_t^c$.

- Feed these conditional independencies to a causal inference algorithm allowing presence of latent common causes, such as the FCI algorithm. Derive the PAG for the contemporary causal structure among variables in $\boldsymbol{X}_t$. Call this PAG π_t.
- For all choices of X_t^1, identify the set of *possible contemporaneous direct causes* of X_t^1, where X_t^2 is a possible contemporaneous direct cause of X_t^1 if in π_t either $X_t^2 \circ\!\!-\!\!\circ X_t^1$, or $X_t^2 \circ\!\!\rightarrow X_t^1$, or $X_t^2 \rightarrow X_t^1$. Denote by $\mathrm{PCDC}(X_t^1)$ the set of possible contemporaneous direct causes of X_t^1.

2. Identify lagged causal relations.

(a) Create a new graph π_f such that the vertices in π_f are $\{\boldsymbol{X}_t, \boldsymbol{X}_{t-1}, \cdots, \boldsymbol{X}_{t-T}\}$, and the edges in π_f are exactly the same as the edges in π_t.

(b) For all choices of X_t^1, $X_{t-i,j}$, and $\boldsymbol{X}_t^b$, determine if X_t^1 and $X_{t-i,j}$ are independent given $\boldsymbol{X}^e$ and $\boldsymbol{X}_t^b$

- For all choices of X_t^1, identify the set of *possible lagged direct causes* of X_t^1, where a lagged variable $X_{t-i,j}$ is a possible lagged direct cause of X_t^1 if for all $\boldsymbol{X}_t^d \subseteq \mathrm{PCDC}(X_t^1)$, $X_{t-i,j}$ and X_t^1 are dependent given $\boldsymbol{X}_t^d$ and $\boldsymbol{X}^e$. Denote by $\mathrm{PLDC}(X_t^1)$ the set of possible lagged direct causes of X_t^1
- For all choices of X_t^1, identify the set of *permanent lagged predictors* of X_t^1, where $X_{t-i,j}$ is a permanent lagged predictor of X_t^1 if for all $\boldsymbol{X}_t^b \subseteq (\boldsymbol{X}_t \setminus \{X_t^1\})$, $X_{t-i,j}$ and X_t^1 are dependent given $\boldsymbol{X}_t^b$ and $\boldsymbol{X}^e$. Denote by $\mathrm{PLP}(X_t^1)$ the set of permanent lagged predictors of X_t^1

(c) Add edges representing the lagged causes of each variable in $\boldsymbol{X}_t$ to π_f:

i. For all choices of X_t^1, add an edge $X_{t-i,j} \rightarrow X_t^1$ to π_f if $X_{t-i,j} \in \mathrm{PLP}(X_t^1)$.

ii. For all choices of X_t^1, add an edge $X_{t-i,j} \rightarrow X_t^1$ to π_f if $X_{t-i,j} \in \mathrm{PLDC}(X_t^1)$, and $X_{t-i,j}$ is not adjacent to any other variable in π_f.

3. Orient the contemporary PAG according to the following rule:

(a) Repeat the following procedure until no more changes can be made to π_f.

i. If $X_{t-i,j} \rightarrow X_t^1 \circ\!\!-\!\!* X_t^2$ is in π_f, and $X_{t-i,j}$ and X_t^2 are not adjacent, then:
If $X_{t-i,j}$ and X_t^2 are independent given $\boldsymbol{X}^e$, but dependent given X_t^1 and $\boldsymbol{X}^e$, then orient the edge between X_t^1 and X_t^2 as $X_t^1 \leftarrow\!\!* X_t^2$

ii. If $X_{t-i,j} \rightarrow X_t^1 \circ\!\!-\!\!* X_t^2$ is in π_f, and $X_{t-i,j}$ and X_t^2 are not adjacent, then:
If $X_{t-i,j}$ and X_t^2 are dependent conditional on $\boldsymbol{X}^e$, but independent conditional on X_t^1 and $\boldsymbol{X}^e$, then orient the edge between X_t^1 and X_t^2 as $X_t^1 \rightarrow X_t^2$

(b) Apply the orientation step of FCI algorithm to further orient the contemporary PAG π_f.

Proposition 2 provides justification for the first step in this procedure, proposition 3 the third step. Proposition 4 is needed for the second step, as we can see that the set of contemporaneous direct causes of a variable X_t^1 is a subset of $\mathrm{PCDC}(X_t^1)$, thus by proposition 4 we have:

$$\text{Lagged direct causes of } X_t^1 \subseteq \text{PLP}(X_t^1) \subseteq \text{PLDC}(X_t^1) \subseteq \text{Lagged causes of } X_t^1$$

Note that step 2(c) is designed to make the procedure more robust.

The complexity of the above procedure is primarily determined by step 1(a), where $k2^{k-1}$ additive model regressions are performed to test the conditional independence relations required by the later steps.

We want to emphasize that the above procedure can be modified in various ways to accommodate changes in the assumptions about the time series data generating models. In the last section (Section 6) of this paper, we discuss in details about different extensions of the above procedure.

4. Simulation Study

In this section, we conduct a simple simulation study to evaluate the performance of the additive non-linear causal inference algorithm presented in Section 3. In particular, we would like to see if the additive non-linear algorithm can provide a viable solution to the problem of nonlinear time series causal inference. For comparison, we also apply a causal inference procedure designed for linear time series to the simulated data. Because there is no currently available efficient *automated* causal inference algorithm for linear time series with contemporaneous causal relations, the linear procedure used for comparison actually is an extension of our additive non-linear causal inference procedure under the assumption that the time series data are generated from linear models. (Bessler et al. 2002, Demiralp and Hoover 2003, Moneta 2003 and Hoover 2005 discussed efficient ways of identifying the contemporaneous causal pattern, that is, the Markov equivalence classes (MEC) of the causal graphs for contemporaneous variables assuming causal sufficiency. However, their procedures are not complete because, when the MEC consists of multiple contemporaneous causal graphs, these procedures all require further background information to uniquely identify the contemporaneous causal graph before proceeding to derive the causal pattern for both contemporaneous and lagged variables. Oxley et al. (2004) provides a less efficient algorithm for linear time series that treats a k-dimensional lag p structural vector autoregressive model (SVAR(p)) as a linear causal model with $k(p+1)$ variables.) The linear procedure differs from the additive non-linear algorithm only in step 1: unlike the original algorithm which uses additive regression to test conditional independence, the linear procedure uses linear regression instead.

We use the Mersenne Twister algorithm implemented in java package RngPack (version 1.1a) for random number generation, and the `gam` function in the R package gam (version 0.97) for additive model regression.

The simulated data are generated from the four causal structures shown in Figure 2. Note that in this simulation study the true PAGs happen to have no circles, and can be represented by the same graphs in Figure 2. The chain-like contemporaneous causal structure is chosen to evaluate the ability of our algorithm to identify the direction of those contemporaneous causal relations that could not be detected using previous algorithms (Bessler et al., 2002; Demiralp and Hoover, 2003; Moneta, 2003; Hoover, 2005). For each causal structure, we consider the following four types of models, characterized by the type of functional relations between an effect variable and its direct causes:

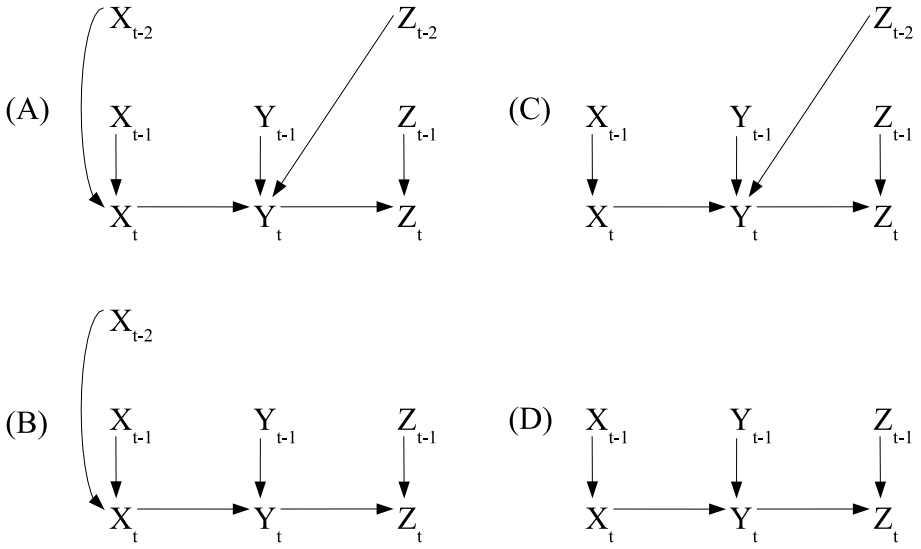


Figure 2: Causal graphs and true PAGs of simulation data

- Trigonometric lag models: Each contemporaneous variable is a linear combination of other contemporaneous variables and univariate trigonometric functions of lagged variables. For example, in one model, we have:

$$Y_t = 0.5X_t + \sin(2Y_{t-1}) - \cos(10Z_{t-2}) + \varepsilon_Y.$$

- Polynomial lag models: Each contemporaneous variable is a linear combination of other contemporaneous variables and univariate polynomial functions of lagged variables. For example, in one model, we have:

$$Y_t = 0.5X_t + 0.3Y_{t-1}^2 - 0.1Z_{t-2}^3 + \varepsilon_Y.$$

- Linear lag models: Each contemporaneous variable is a linear combination of other contemporaneous variables and lagged variables. For example, in one model, we have:

$$Y_t = 0.5X_t + 0.3Y_{t-1} - 0.1Z_{t-2} + \varepsilon_Y.$$

- Trigonometric contemporaneous models: Each contemporaneous variable is a linear combination of univariate trigonometric functions of other contemporaneous variables and lagged variables. For example, in one model, we have:

$$Y_t = \cos(X_t) + \sin(2Y_{t-1}) - \cos(10Z_{t-2}) + \varepsilon_Y.$$

Note that these models do not belong to the family of additive non-linear time series models, for they violate the assumption C1.

In total we have 16 data generating models, with 12 of them being additive non-linear time series models (including 4 linear time series models). For each of the 16 models, we generate 4 random time series data sets of length 200, 500, 1000, and 2000 respectively. For each data set, we run both the additive non-linear procedure and the linear procedure. The upper bound T_{max} of the true lag number T is set to 3 for all simulations, (T is equal to 2 for 12 of the data generating models based on casual structure (A), (B), and (C) in Figure 2, and 1 for the other 4 models based on casual structure (D)). The learned PAGs are compared with the true PAGs, which are also represented by the graphs in Figure 2.

The additive non-linear procedure presented in Section 3 requires, for each contemporaneous variable, say X_t, the following two types of conditional independence information: (1) if X_t is independent of another contemporaneous variable, say Y_t, given all the lagged variables $\boldsymbol{L} = \{X_{t-2}, X_{t-1}, Y_{t-2}, Y_{t-1}, Z_{t-2}, Z_{t-1}\}$ and a subset of the remaining contemporaneous variables, say, $\{Z_t\}$; and (2), if X_t is independent of a lagged variable, say X_{t-1}, given all the other lagged variables and a subset of contemporaneous variables, say, $\{Z_t\}$. These conditional independence relations are tested by checking if $\mathrm{E}[X_t|\boldsymbol{L}, Z_t]$ is constant in Y_t or X_{t-1} respectively. For example, to test if X_{t-1} is present in $\mathrm{E}[X_t|\boldsymbol{L}, Z_t]$, we follow Huang and Yang (2004) by starting from a model A, where X_t is regressed against $\boldsymbol{L}$ and Z_t, and searching for a submodel of A with the lowest BIC score. If X_{t-1} is present in this submodel with lowest BIC score, it is present in $\mathrm{E}[X_t|\boldsymbol{L}, Z_t]$. Otherwise, it is not.

The simulation results are summarized in Figure 3. Each of the four panes in Figure 3 summarizes the results of 16 simulated time series data sets generated from the same type of models. We use the average error rates to evaluate the performance of the two algorithms. The definitions of the various error rates are similar to those in Spirtes and Meek (1995). Consider a p dimensional time series data. An edge omission error occurs when two variables are adjacent in the true PAG but not in the learned PAG. An edge commission error occurs when two variables are adjacent in the learned PAG but absent in the true PAG.

The edge omission error rate is defined as:

$$E_o = \frac{\text{Number of edge omission errors}}{\text{Number of edges in the true PAG}}.$$

The edge commission error rate is defined as:

$$E_c = \frac{\text{Number of edge commission errors}}{\text{Maximum number of possible edge commission errors}}.$$

When inferring causal structure from a p dimensional time series data set, if the upper bound of the true lag number is set to T_{max}, the maximum number of possible edge commission errors is equal to:

$$p^2 T_{max} + \frac{p(p-1)}{2} - \text{Number of edges in the true PAG}$$

where $p^2 T_{max} + p(p-1)/2$ is the maximum number of edges can be found in the unit causal graph for any p-dimension lag T_{max} time series model.

The solid lines in each pane of Figure 3 represent the average omission error rates for different time series lengths; the dotted lines represent the average commission error rates. Blue lines with

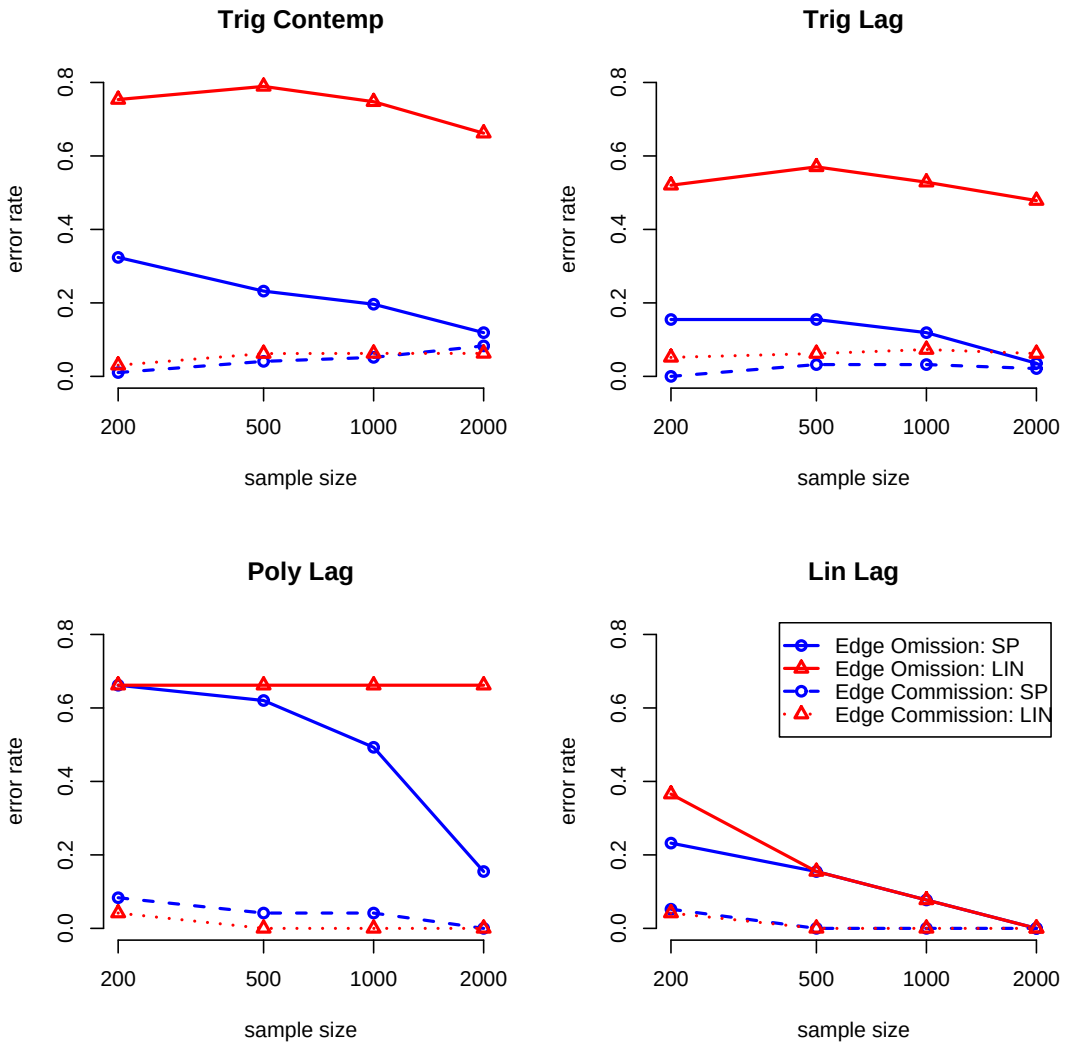


Figure 3: Error rate for edge discovery

circles represent results obtained by the additive non-linear algorithm, red lines with triangles the results by the linear procedure.

The pane with label "Trig Contemp" gives the results for data generated from the *trigonometric contemporaneous models*. We choose these models in the simulation study precisely because they lie outside of the family of additive non-linear time series models, for they violate the functional assumption (C1) in the definition of additive non-linear time series models. The simulation results suggest that, when the assumption C1 is violated, the additive non-linear algorithm can still discover most of the edges. However, as the length of time series increases, the average number of extra edges also increases, apparently because the data generating models are not additive non-linear time series models. The linear procedure is not satisfactory, missing most of the edges in the true models.

The panes labeled with "Trig Lag" and "Poly Lag" show the results for *trigonometric lag models* and *polynomial lag models*, both of which are genuine additive non-linear time series models. The additive non-linear algorithm performs very well for the trigonometric lag models, but less than satisfactory for polynomial lag models. Its performance for polynomial lag models, however, does improve as the length of time series increases. The linear procedure performs poorly in both cases, missing at least half of the edges.

The pane with label "Lin Lag" provides the results for *linear lag models*. Given that a linear lag model is simply a linear time series model, which is a special case of additive non-linear time series model, we expect that both algorithms should perform very well, as they do. This, on the one hand, suggests that the linear procedure is a good choice for linear time series causal inference, on the other hand, implies that the additive non-linear algorithm does not suffer from overfitting.

We also compare the average error rates for orientation of the edges among contemporaneous variables by the additive non-linear algorithm and the linear procedure. Suppose X_t and Y_t are adjacent in both the learned PAG and the true PAG, an arrowhead omission error occurs if the edge is oriented as $X_t *\!\!\rightarrow Y_t$ in the true PAG, but as $X_t *\!\!-\!\!* Y_t$ or $X_t *\!\!-\!\!\circ Y_t$ in the learned PAG. Similarly, an arrowhead commission error occurs if the edge is oriented as $X_t -\!\!* Y_t$ or $X_t \circ\!\!-\!\!* Y_t$ in the true PAG, but as $X_t \leftarrow\!\!* Y_t$ in the learned PAG. Let E be the set of edges among contemporaneous variables in the true PAG such that the pairs of variables connected by these edges are also adjacent in the learned PAG. The arrowhead omission error rate is defined as:

$$A_o = \frac{\text{Number of arrowhead omission errors}}{\sum_{e \in E} \text{Number of arrowheads in } e}.$$

The arrowhead commission error rate is defined as:

$$A_c = \frac{\text{Number of arrowhead commission errors}}{\sum_{e \in E} \text{Number of non-arrowheads in } e}.$$

In Figure 4, the solid lines in each pane represent the average *arrowhead omission error rates*; the dotted lines represent the average *arrowhead commission error rates*. As in Figure 3, blue lines with circles represent results obtained by the additive non-linear algorithm, red lines with triangles the results by the linear procedure. (Note that in the top two panels labeled respectively with "Trig Contemp" and "Trig Lag", the lines representing omission error and commission error for the additive non-linear algorithm overlap. In the bottom two panels labeled respectively with "Poly Lag" and "Lin Lag", the lines representing commission error for the additive non-linear algorithm and the linear algorithm overlap.)

There are two more scores to measure how close a learned PAG is to the true PAG, that is, the tail omission error rate and the tail commission error rate. Suppose X_t and Y_t are adjacent in both the learned PAG and the true PAG, a tail omission error occurs if the edge is oriented as $X_t \rightarrow Y_t$ in the true PAG, but as $X_t \circ\!\!-\!\!* Y_t$ in the learned PAG. A tail commission error occurs if the edge is oriented as $X_t \circ\!\!-\!\!* Y_t$ in the true PAG, but as $X_t \rightarrow Y_t$ in the learned PAG. Note that these definitions are stated so that an arrowhead commission/omission error will not be counted again as a tail omission/commission error. Because there is no circle in the true PAGs in this simulation study, we can only compute the tail omission errors for the learn PAGs, shown in Figure 5. The tail omission error rate is defined as:

$$T_o = \frac{\text{Number of tail omission errors}}{\sum_{e \in E} \text{Number of tails in } e}.$$

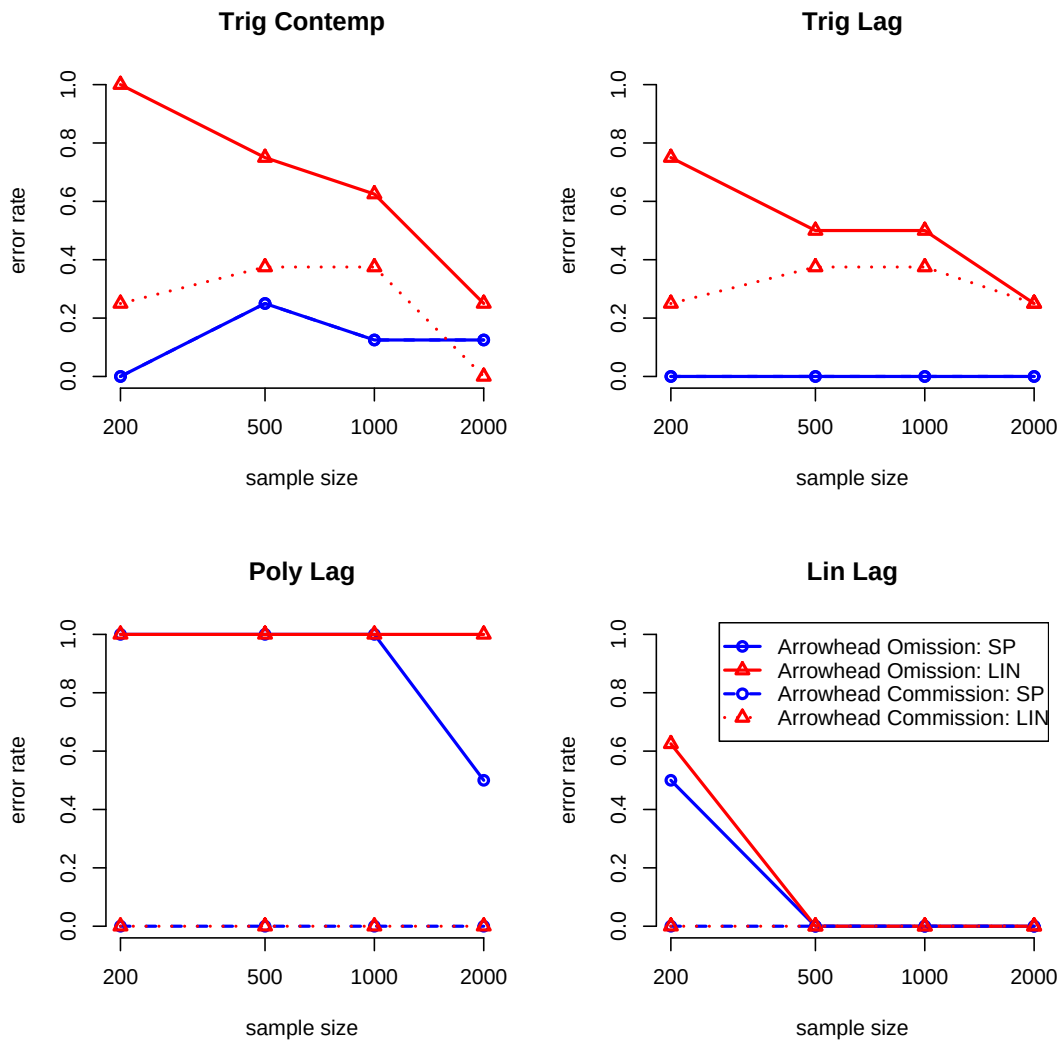


Figure 4: Error rate for edge orientation: Arrowhead

The additive non-linear algorithm gives excellent results. For example, for the data sets generated from additive non-linear models, that is, the trigonometric lag models, the polynomial lag models, and linear lag models, the additive non-linear algorithm makes no arrowhead commission errors. The linear procedure performs quite well for polynomial lag models and linear lag models.

Although the scope of this simulation study is very limited, we can get some general idea about the performance of our additive non-linear casual inference algorithm. If we count the number of variables in a p dimensional lag T additive non-linear time series model as $p(T+1)$, then roughly speaking, for longer time series, (80 or more observations per variable), the additive non-linear algorithm outperforms the linear procedure in all situations, including cases where the true model is more complex than the additive non-linear model and cases where the true model is a linear model.

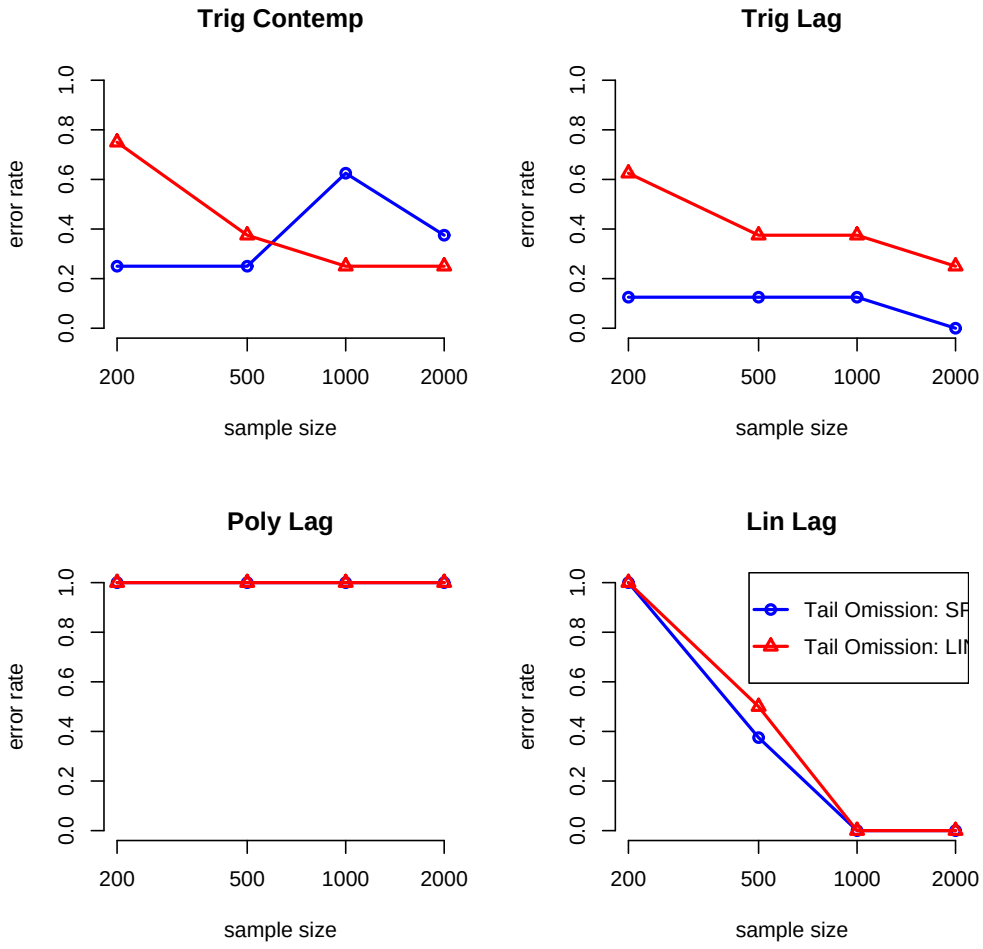


Figure 5: Error rates for edge orientation: Tail

For shorter time series, (less than 40 observations per variable), the additive non-linear model is still better in general, but may be not as good as the linear procedure in some cases. Our suggestion is that, for longer time series always choose the additive non-linear algorithm. For shorter time series, if computational cost is critical, the linear procedure is a reasonable choice; otherwise we still recommend the additive non-linear algorithm, or better yet, try both of them.

5. Case Study: Ocean Climate Indices

To illustrate the application of the additive non-linear causal inference algorithm for nonlinear time series, we use it to study the causal relations among some ocean climate indices.

Climate teleconnections are associations of geospatially remote climate phenomena produced by atmospheric and oceanic processes. The most famous, and first established teleconnection, is the association of the El Nino/Southern Oscillation (ENSO) with the failure of monsoons in India. A variety of associations have been documented among sea surface temperatures (SST), atmospheric pressure at sea level (SLP), land surface temperatures (LST) and precipitation over land areas. Since the 1970s data from a sequence of satellites have provided monthly (and now daily) measurements of such variables, at resolutions as small as 1 square kilometer. Measurements in particular spatial regions have been clustered into time indexed indices for the regions, usually by principal components analysis, but also by other methods. Climate research has established that some of these phenomena are exogenous drivers of others, and has sought physical mechanisms for the teleconnections. We consider here whether constraints on such mechanisms can be obtained by data-driven model selection from time series of ocean indices.

Our data set consists of the following 4 ocean climate indices, recorded monthly from 1958 to 1999, each forming a time series of 504 time steps:

SOI Southern Oscillation Index: Sea Level Pressure (SLP) anomalies between Darwin and Tahiti

WP Western Pacific: Low frequency temporal function of the 'zonal dipole' SLP spatial pattern over the North Pacific.

AO Arctic Oscillation: First principal component of SLP poleward of 20° N

NAO North Atlantic Oscillation: Normalized SLP differences between Ponta Delgada, Azores and Stykkisholmur, Iceland

To check stationarity, we conduct the augmented Dickey-Fuller (ADF) test. ADF tests for all 4 time series reject the null hypothesis that the tested series has a unit root against the alternative that the series is stationary, with p values of the tests smaller than 0.01. As a complementary to ADF tests, we also conduct the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test. For all 4 time series, KPSS tests with lag truncation parameter set to 12 fail to reject the null hypothesis that the tested series is (trend) stationary against the unit root alternative, with p values of the tests higher than 0.1. We also plot the autocorrelations for the 4 time series to check if the data satisfies the strong mixing condition (Figure 6). The idea is that, if a time series satisfies the strong mixing condition, its autocorrelation should decrease rapidly as the lag increases. From the plot, the auto correlations of SOI do not decrease as quickly as for other indices, but they become insignificant when the lag is above 12 months.

We assume that the 4 indices are generated from a lag 12 additive non-linear model. The choice of 12 is partly based on the fact that the ocean indices are monthly data. Another concern is that with a length of 504, the data would be too sparse for a model with a much longer lag. As in the simulation study, the R package gam (version 0.97) is used in this analysis. We first remove any linear trend from the data, then, following the causal inference procedure presented in Section 3, derive a causal structure represented by a PAG for the 4 ocean climate indices. Figure 7 gives the learned causal structure.

Because of the relative shorter length of the ocean indices data (10 observations per variable for a lag 12 model), it is worth conducting another inference on the 4 ocean indices using the linear procedure. The resulting causal structure is given in Figure 8.

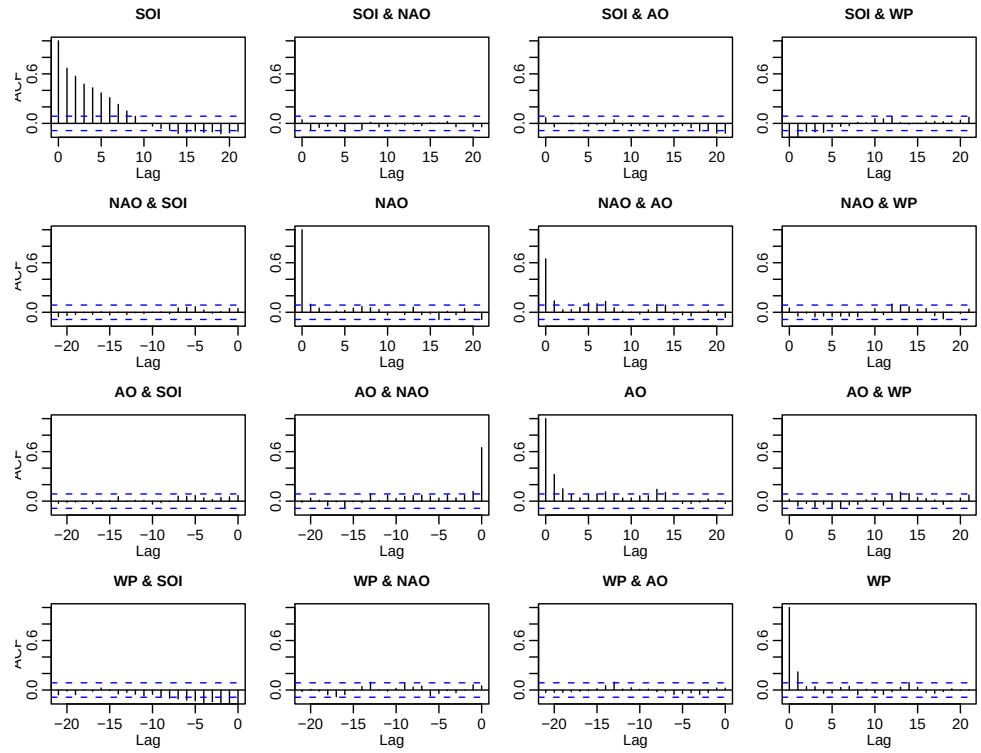


Figure 6: Autocorrelation plot for 4 times series

Without a gold standard, it is hard to say which method gives the more accurate information in this case. But the graph obtained using the linear procedure is likely to miss some nonlinear dependencies. For example, an arrow from SOI_{t-1} to AO_t is present in Figure 7, but absent from Figure 8. It turns out, when regressing AO_t against SOI_{t-1}, AO_{t-1} and NAO_{t-1} using additive model regression, the estimated influence of SOI_{t-1} on AO_t is clearly nonlinear (see Figure 9, where the contribution of SOI_{t-1} to AO_t is plotted as a smooth univariate function of SOI_{t-1}). This is not surprising given the complexity of the processes represented by the ocean climate indices, and illustrates the need of causal inference procedures that can be applied to data generated from nonlinear models.

6. Discussion

Methods of causal inference, first developed in the machine learning literature, have been successfully applied to many diverse fields, including biology, medicine, and sociology (Pearl, 2000; Spirtes et al., 2000). An essential and distinct feature of these methods is that they require comparatively less domain knowledge about the system to be studied.

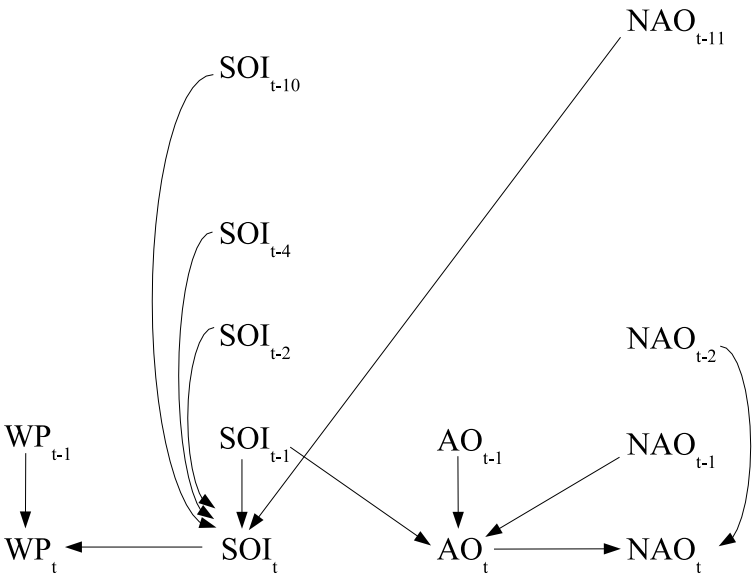


Figure 7: Causal connections among 4 ocean climate indices, using the additive non-linear algorithm

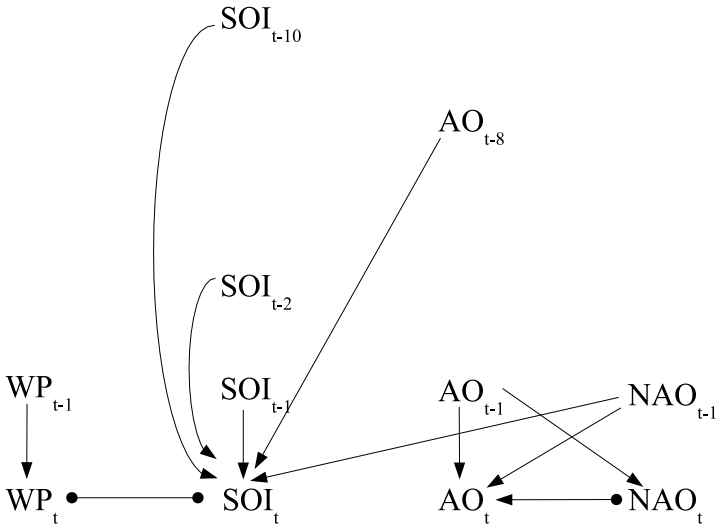


Figure 8: Causal connections among 4 ocean climate indices, using the linear procedure

This study extends the application of causal inference to nonlinear time series data. We present a new procedure that combines semi-automated model search for causal structure with additive

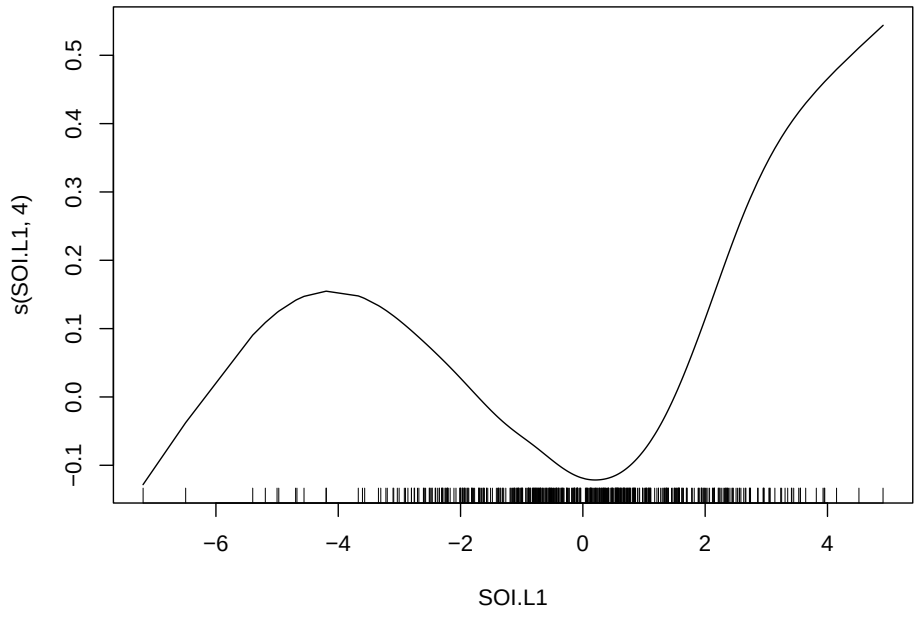


Figure 9: Nonlinear relation between SOI_{t-1} and AO_t

model regression methods. The particular example is to ocean climate indices, but the component procedures have been individually applied to econometric data with some success, suggesting that the criteria for successful application of the joint procedures are statistical and causal rather than domain specific.

Our approach is modular, and its two main components, that is, conditional independence testing and causal model search, could be replaced by other comparable methods. Thus, with appropriate data generated from appropriate mechanisms, related analyses could be conducted under weaker or alternative assumptions. Below we briefly discuss several possible extensions of our method:

6.1 Nonstationary Nonlinear Time Series

In most of this paper we assume that the nonlinear time series are stationary, only because it has been shown that for stationary nonlinear time series data satisfying certain conditions, nonparametric regression is asymptotically consistent. The algorithm and propositions proposed in this paper do not require stationarity. However, to apply our algorithm to nonstationary nonlinear time series data, we must find an efficient regression method to estimate the conditional expectations and conduct conditional independence tests. Cointegration analysis is not suitable for this purpose, because it is mainly designed for and applicable to cointegrated linear time series (Engle and Granger, 1987; Johansen, 1991). However, recent studies on applying nonparametric regression methods to nonstationary time series data (Phillips and Park, 1998; Karlsen and Tjøstheim, 2001; Bandi and Phillips, 2003; Karlsen et al., 2005) seem promising. Not surprisingly, the convergence rate of nonparametric regression for nonstationary time series data may be slower than that of stationary data.

6.2 Feedback Models

The original definition of additive non-linear models in Section 2 does not allow any feedback among contemporaneous variables (see condition C4). To represent mutual influences among contemporary variables, we can remove condition C4 from the original definition. We also have to drop the U terms in Equation 2 because the currently available algorithm capable of handling feedbacks (Richardson and Spirtes, 1999) does not work in the presence of latent common causes. The resulting definition defines a *additive non-linear feedback model*, which, compared to the additive non-linear model, allows feedback, but not latent common causes. Propositions 1, 2, and 3 still hold for the new model, proposition 4 needs some modification:

Proposition 4': Let $\boldsymbol{X}_t^b$ be the set of all contemporary direct causes of X_t^1. Assuming there is no latent common cause, $X_{t-i,j}$ and X_t^1 are dependent conditional on $\boldsymbol{X}_t^b$ and $\boldsymbol{X}^e$ if and only if either $X_{t-i,j}$ is either a direct cause of X_t^1, or a direct cause of a contemporaneous cause of X_t^1.

The only change needed in the causal inference procedure to handle data generated from additive non-linear feedback models is, in step 1(b), that the FCI algorithm should be replaced by a consistent causal inference algorithm capable of outputting cyclic graphs, such as the one proposed in Richardson and Spirtes (1999).

6.3 Score Based Search Procedure

The causal inference procedure presented in Section 3 is constraint based. That is, the procedure requires explicit conditional independence information as input, (although each conditional independence constraint is obtained using a BIC score based model selection procedure). As we mentioned in Section 3, the main advantage of this procedure and its modified version is that they can handle the presence of latent common causes or feedbacks in the contemporaneous causal structure. (Drton et al. 2006 provides a maximum likelihood estimation algorithm that allows the computation of BIC scores for certain types of linear models with correlated error terms, though not for the contemporaneous causal structure of a additive non-linear model.) If we are willing to exclude feedbacks and latent common causes, a simple two-step score based procedure can be used to infer causal information from data generated by additive non-linear models. In the first step, a score based algorithm, such as the GES algorithm (Meek, 1996; Chickering, 2002a,b), is applied to the residuals of additive model regression of contemporaneous variables against all lags to obtain a causal pattern representing a Markov equivalence class π_t of directed acyclic graphs for the contemporaneous variables. In the second step, for each directed acyclic graph G belonging to the Markov equivalence class π_t, we generate a time series causal model M and compute its BIC score in the following way:

- Each contemporaneous variable X_t^i is regressed against its parents in G and all the lagged variables $\boldsymbol{X}^l$. The BIC score method proposed in Huang and Yang (2004) is used to identify the best submodel (with the lowest BIC score s_i). The significant predictors of X_t^i in that best submodel are direct causes of X_t^i in causal model M.

- The BIC score of causal model M is $\sum_i s_i$.

The causal model with the best (lowest) BIC score then is returned as the result of the score based casual inference algorithm.

Acknowledgments

The authors thank the anonymous referees for their helpful comments to improve this paper. This research was completed when the first author was research scientist at Florida Institute for Human and Machine Cognition. The research was supported by NASA contract NC2-1399 to the University of West Florida.

References

H. An and F. Huang. The geometrical ergodicity of nonlinear autoregression models. *Statistica Sinica*, 6:943–956, 1996.

E. Baek and W. Brock. A general test for nonlinear Granger causality: A bivariate model. URL `http://www.ssc.wisc.edu/~wbrock/Baek Brock Granger.pdf`. January 1992.

F. Bandi and P. Phillips. Fully nonparametric estimation of scalar diffusion models. *Econometrica*, 71:241–283, 2003.

D. Bell, J. Kay, and J. Malley. A non-parametric approach to non-linear causality testing. *Economics Letters*, 51:7–18, 1996.

D. Bessler, J. Yang, and M. Wongcharupan. Price dynamics in the international wheat market: Modeling with error correction and directed acyclic graphs. *Journal of Regional Science*, 42: 793–825, 2002.

Z. Cai and E. Masry. Nonparametric estimation of additive nonlinear arx time series: Local linear fitting and projections. *Journal of Econometric Theory*, 16:465–501, 2000.

K. Chan and H. Tong. A note on noisy chaos. *Journal of the Royal Statistical Society B*, 56:301–311, 1994.

R. Chen and R. Tsay. Functional-coefficient autoregressive models. *Journal of the American Statistical Association*, 88:298–308, 1993.

D. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002a.

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

D. Cline and H. Pu. Geometric ergodicity of nonlinear time series. *Statistica Sinica*, 9:1103–1118, 1999.

Y. Davydov. Mixing conditions for Markov chains. *Theory of Probability and its Applications*, 18: 312–328, 1973.

S. Demiralp and K. Hoover. Searching for the causal structure of a vector autoregression. *Oxford Bulletin of Economics and Statistics*, 65:745–767, 2003.

C. Diks and V. Panchenko. A new statistic and practical guidelines for nonparametric Granger causality testing. *Journal of Economic Dynamics and Control*, 30:1647–1669, 2006.

M. Drton, M. Eichler, and T. S. Richardson. Identification and Likelihood Inference for Recursive Linear Models with Correlated Errors. *ArXiv Mathematics e-prints*, August 2006. URL http://arxiv.org/abs/math/0601631v3.

R. Engle and C. Granger. Cointegration and error correction: Representation, estimation, and testing. *Econometrica*, 55:251–276, 1987.

W. Härdle, H. Lütkepohl, and R. Chen. A review of nonparametric time series analysis. *International Statistical Review*, 65:49–72, 1997.

T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall, New York, NY, 1990.

C. Hiemstra and J. Jones. Testing for linear and nonlinear Granger causality in the stock price-volume relation. *Journal of Finance*, 49:1639–1664, 1994.

K. Hoover. Automatic inference of the contemporaneous causal order of a system of equations. *Econometric Theory*, 21:69–77, 2005.

P. O. Hoyer, S. Shimizu, and A. J. Kerminen. Estimation of linear, non-Gaussian causal models in the presence of confounding latent variables. pages 155–162, Prague, Czech Republic, 2006.

J. Huang and L. Yang. Identification of non-linear additive autoregressive models. *Journal of the Royal Statistical Society: Series B*, 66:463–477, 2004.

S. Johansen. Estimation and hypothesis testing of cointegration vectors in Gaussian vector autoregressive models. *Econometrica*, 59:1551–1580, 1991.

H. Karlsen and D. Tjøstheim. Nonparametric estimation in null recurrent time series. *The Annals of Statistics*, 29:372–416, 2001.

H. Karlsen, T. Myklebust, and D. Tjøstheim. Nonparametric estimation in a nonlinear cointegration type model. Working paper, 2005. URL http://www.mi.uib.no/∼karlsen/working_paper/NonlinCoint05.pdf.

C. Meek. *Graphical Models: Selecting Causal and Statistical Models*. PhD thesis, Carnegie Mellon University, Philosophy Department, 1996.

A. Moneta. Graphic models for structural vector autoregressions. Working paper, Laboratory of Economics and Management, Sant'Anna School of Advanced Studies, Pisa, Italy, 2003.

L. Oxley, M. Reale, and G. Tunnicliffe-Wilson. Finding directed acyclic graphs for vector autoregressions. In *Proceedings in Computational Statistics 2004*, pages 1621–1628, 2004.

J. Pearl. *Causality*. Cambridge University Press, Cambridge, UK, 2000.

P. Phillips and J. Park. Nonstationary density estimation and kernel autoregression. Discussion Paper 1181, Cowles Foundation, Yale University, 1998.

T. Richardson and P. Spirtes. Automated discovery of linear feedback models. In C. Glymour and G. Cooper, editors, *Computation, Causation and Discovery*, chapter 7, pages 253–302. MIT Press, Cambridge, MA, 1999.

P. Robinson. Nonparametric estimation for time series models. *Journal of Time Series Analysis*, 4: 185–208, 1983.

R. Silva, R. Scheines, C. Glymour, and P. Spirtes. Learning the structure of linear latent variable models. *Journal of Machine Learning Research*, 7:191–246, 2006.

P. Spirtes and C. Meek. Learning Bayesian networks with discrete variables from data. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 294–299, 1995.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, New York, NY, second edition, 2000.

C. Stone. Additive regression and other nonparametric models. *The Annals of Statistics*, 13:689–705, 1985.

L. Su and H. White. A nonparametric Hellinger metric test for conditional independence. *Econometric Theory*, (forthcoming), 2007.

N.R. Swanson and C.W.J. Granger. Impulse response functions based on a causal approach to residual orthogonalization in vector autoregressions. *Journal of the American Statistical Association*, 92:357–367, 1997.

D. Tjøstheim and B. Auestad. Nonparametric identification of nonlinear time series: Projections. *Journal of the American Statistical Association*, 89:1398–1409, 1994.

Y. Truong and C. Stone. Nonparametric function estimation involving time series. *The Annals of Statistics*, 20:77–97, 1992.

Y. Xia and H. An. Projection pursuit autoregression in time series. *Journal of Time Series Analysis*, 20:693–714, 1999.

Journal of Machine Learning Research 9 (2008) 993-996 Submitted 10/07; Revised 3/08; Published 6/08

Shark

Christian Igel CHRISTIAN.IGEL@NEUROINFORMATIK.RUB.DE
Verena Heidrich-Meisner VERENA.HEIDRICH-MEISNER@NEUROINFORMATIK.RUB.DE
Tobias Glasmachers TOBIAS.GLASMACHERS@NEUROINFORMATIK.RUB.DE
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany

Editor: Soeren Sonnenburg

Abstract

SHARK is an object-oriented library for the design of adaptive systems. It comprises methods for single- and multi-objective optimization (e.g., evolutionary and gradient-based algorithms) as well as kernel-based methods, neural networks, and other machine learning techniques.

Keywords: machine learning software, neural networks, kernel-methods, evolutionary algorithms, optimization, multi-objective-optimization

1. Overview

SHARK is a modular C++ library for the design and optimization of adaptive systems. It serves as a toolbox for real world applications and basic research in computational intelligence and machine learning. The library provides methods for single- and multi-objective optimization, in particular evolutionary and gradient-based algorithms, kernel-based learning methods, neural networks, and many other machine learning techniques. Its main design criteria are flexibility and speed. Here we restrict the description of SHARK to its core components, albeit the library contains plenty of additional functionality. Further information can be obtained from the HTML documentation and tutorials. More than 60 illustrative example programs serve as starting points for using SHARK.

2. Basic Tools—Rng, Array, and LinAlg

The library provides general auxiliary functions and data structures for the development of machine learning algorithms. The Rng module generates reproducible and platform independent sequences of pseudo random numbers, which can be drawn from 14 predefined discrete and continuous parametric distributions. The Array class provides dynamical array templates of arbitrary type and dimension as well as basic operations acting on these templates. LinAlg implements linear algebra algorithms such as matrix inversion and singular value decomposition.

3. ReClaM—Regression and Classification Methods

The goal of the ReClaM module is to provide machine learning algorithms for supervised classification and regression in a unified, modular framework. It is built like a construction kit, where the main building blocks are adaptive data processing models, error functions, and optimization

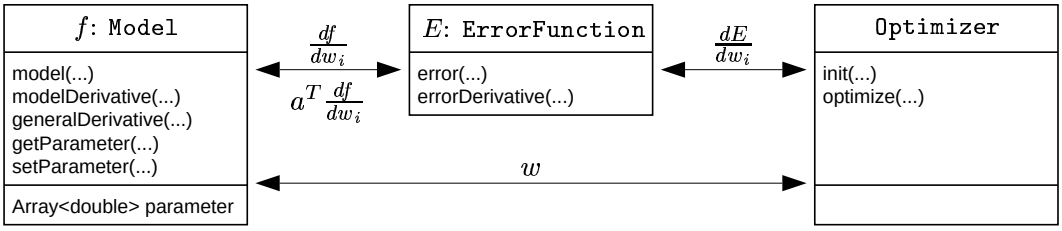


Figure 1: Almost all ReClaM objects are inherited from one of the three base classes `Model`, `ErrorFunction`, and `Optimizer`. The optimizer has access to the parameter vector w of the model $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^m$, $(x, w) \mapsto f_w(x)$, to minimize a scalar error function E. For gradient-based optimization, the error function provides the derivative dE/dw based on df/dw. In many cases we can speed up the computation of dE/dw by a factor of m by using $a^T df/dw$, where a is a vector of coefficients dependent on the error function.

algorithms (see Figure 1). The superclasses representing these components communicate through fixed interfaces. Problem definition and solution are clearly separated. A problem is defined by a model defining a parametric family of candidate hypotheses, and a possibly regularized error function to minimize (and, of course, sample data). It is usually solved with an (iterative) optimization algorithm, which adapts the model parameters in order to minimize the error function evaluated on the given data set. Additional error functions and data sets can then be used to test the resulting performance. This clear structure makes ReClaM easy to use and extend.

ReClaM focuses on kernel methods and neural networks. It offers a variety of predefined network models including feed-forward and recurrent multi-layer perceptron networks, radial basis function networks, and CMACs. Several gradient-based optimization algorithms are available for network training and general purpose optimization including the conjugate gradient method, the BFGS algorithm, and improved Rprop (Igel and Hüsken, 2003).

In the remainder of this section we present the realization of kernel-based learning in more detail. The library offers kernelized versions of several learning machines from nearest neighbor classifiers and simple Gaussian processes to different flavors of support vector machines. These algorithms operate on general kernel objects and users can supply new kernel functions easily. At the time of writing, ReClaM provides the fastest support vector machine (SVM) implementation for dense large-scale learning problems. The SVM training automatically switches between the most efficient SMO-like algorithms available depending on the current problem size (Fan et al., 2005; Glasmachers and Igel, 2006).

On top of these models, ReClaM defines meta-models for model selection of kernel and regularization parameters. It offers more objective functions and optimization methods for model selection than any other library. Objective functions include leave-one-out and cross validation errors, radius-margin quotient, kernel-target alignment, and the span bound (Chapelle et al., 2002; Glasmachers and Igel, 2005; Igel et al., 2007a). For optimization, nested grid-search and evolutionary kernel learning are supported, and efficient gradient-based optimization is available whenever possible. For both model training and model selection, we make use of ReClaM's superclass architecture to describe and solve the optimization problems. For example, a gradient-based optimization algorithm

may decrease a radius-margin quotient in order to adapt the hyperparameters of an SVM, where in each iteration an SVM model is trained by a special quadratic program optimizer to determine the margin.

To reduce the complexity of SVMs and Gaussian processes after training, algorithms for approximating the solutions in feature space are implemented (Romdhani et al., 2004; Suttorp and Igel, 2007).

4. EALib and MOO-EALib—Evolutionary Single- and Multi-objective Optimization

The evolutionary algorithms module (EALib) implements classes for stochastic direct optimization using evolutionary computing, in particular genetic algorithms and evolution strategies (ESs). Evolutionary algorithms (EAs) maintain populations (i.e., multi-sets) of candidate solutions. In the EALib structure, instances of the class `Population` contain instances of `Individual` consisting of one or more `Chromosome`s, which can have different types. Numerous variation (i.e., mutation and recombination) operators for different types of chromosomes, for example real-valued or binary vectors, are available. The user has the choice between many different deterministic and stochastic selection mechanisms operating on population level.

The MOO-EALib extends the EALib to evolutionary multi-objective (i.e., vector valued) optimization (EMO). The goal of EMO is usually to approximate the set of Pareto-optimal solutions, where a solution is Pareto-optimal if it cannot be improved in one objective without getting worse in another one. To our knowledge, the MOO-EALib module makes SHARK one of the most comprehensive libraries for EMO. The efficient implementation of measures for quantifying the quality of sets of candidate solutions is a strong argument for the MOO-EALib.

In SHARK we put an emphasis on variable-metric ESs for real-valued optimization. Thus, the most recent implementation of the covariance matrix adaptation ES (CMA-ES; Hansen et al., 2003) and its EMO counterpart (Igel et al., 2007b) are included. We do not know any C++ toolbox for EAs that comes close to the EALib in terms of flexibility and quality of algorithms for continuous optimization.

5. Availability and Requirements

The C++ source code is available from `http://shark-project.sourceforge.net` under GNU Public License and compiles under MS Windows, Linux, Solaris, and MacOS X. No third-party libraries are required, except Qt and Qwt for graphical examples.

Acknowledgments

The authors of this paper comprise the team responsible for a major revision and the maintenance of the SHARK library at the time of writing the article. The SHARK project was started by M. Kreutz, who wrote the basic components such as LinAlg, Array, and Rng as well as the EALib. Then B. Sendhoff joined the project, which was fused with C. Igel's ReClaM library. Afterwards, many people contributed to the package, in particular (in alphabetic order) R. Alberts, T. Bücher, A. W. Dietrich, who invented the name Shark, T. Glasmachers, who extended the ReClaM library, M. Hüsken, T. Okabe, who wrote the MOO-EALib, S. Roth, P. Stagge, T. Suttorp, M. Toussaint, and T. Voß. The SHARK project is supported by the Honda Research Institute Europe.

References

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using the second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

T. Glasmachers and C. Igel. Gradient-based adaptation of general Gaussian kernels. *Neural Computation*, 17(10):2099–2105, 2005.

T. Glasmachers and C. Igel. Maximum-gain working set selection for support vector machines. *Journal of Machine Learning Research*, 7:1437–1466, 2006.

N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11 (1):1–18, 2003.

C. Igel and M. Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50(C):105–123, 2003.

C. Igel, T. Glasmachers, B. Mersch, N. Pfeifer, and P. Meinicke. Gradient-based optimization of kernel-target alignment for sequence kernels applied to bacterial gene start detection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):216–226, 2007a.

C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007b.

S. Romdhani, P. Torr, B. Schölkopf, and A. Blake. Efficient face detection by a cascaded support-vector machine expansion. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 460(2051):3283–3297, 2004.

T. Suttorp and C. Igel. Resilient simplification of kernel classifiers. In J. Marques de Sá et al., editors, *International Conference on Artificial Neural Networks (ICANN 2007)*, volume 4668 of *LNCS*, pages 139–148. Springer-Verlag, 2007.

Journal of Machine Learning Research 9 (2008) 997-1017 Submitted 8/07; Revised 1/08; Published 6/08

Hit Miss Networks with Applications to Instance Selection

Elena Marchiori ELENAM@CS.RU.NL
Department of Computer Science
Radboud University
Nijmegen, The Netherlands

Editor: Leon Bottou

Abstract

In supervised learning, a training set consisting of labeled instances is used by a learning algorithm for generating a model (classifier) that is subsequently employed for deciding the class label of new instances (for generalization). Characteristics of the training set, such as presence of noisy instances and size, influence the learning algorithm and affect generalization performance. This paper introduces a new network-based representation of a training set, called hit miss network (HMN), which provides a compact description of the nearest neighbor relation over pairs of instances from each pair of classes. We show that structural properties of HMN's correspond to properties of training points related to the one nearest neighbor (1-NN) decision rule, such as being border or central point. This motivates us to use HMN's for improving the performance of a 1-NN classifier by removing instances from the training set (instance selection). We introduce three new HMN-based algorithms for instance selection. HMN-C, which removes instances without affecting accuracy of 1-NN on the original training set, HMN-E, based on a more aggressive storage reduction, and HMN-EI, which applies iteratively HMN-E. Their performance is assessed on 22 data sets with different characteristics, such as input dimension, cardinality, class balance, number of classes, noise content, and presence of redundant variables. Results of experiments on these data sets show that accuracy of 1-NN classifier increases significantly when HMN-EI is applied. Comparison with state-of-the-art editing algorithms for instance selection on these data sets indicates best generalization performance of HMN-EI and no significant difference in storage requirements. In general, these results indicate that HMN's provide a powerful graph-based representation of a training set, which can be successfully applied for performing noise and redundance reduction in instance-based learning.

Keywords: graph-based training set representation, nearest neighbor, instance selection for instance-based learning

1. Introduction

In supervised learning, a machine learning algorithm is given a training set, consisting of training examples called labeled *instances* (here called also *points*). Each instance consists of an input vector of values, one for each *variable* of the learning task, and has assigned a class label. A machine learning algorithm uses the training set to generate a so-called *model* that is subsequently used for deciding the class label of (*classify*) new instances. In particular, the 1-NN rule classifies an unknown point into the class of the nearest of the training set points. This rule does not rely on knowledge of the underlying data distribution (non-parametric classification). Moreover, for all distributions, its probability of error is bounded above by twice the Bayes' probability of error (Cover and Hart, 1967).

A central issue in 1-NN classification, and more generally in instance-based learning, concerns storage requirements. The basic 1-NN rule stores all training instances, hence can be slow when classifying new instances. Moreover, when the training set contains noisy instances, generalization accuracy can be negatively affected if these instances are stored as well (see Wilson and Martinez, 2000). Instance selection algorithms tackle these issues by selecting a subset of the training set in order to reduce storage and possibly also enhance accuracy of the 1-NN rule on new instances (generalization performance).

In this paper we introduce a new graph-based representation of a training set, called Hit Miss Network. In an HMN, nodes are instances of the considered training set. Edges are defined as follows: for each node x and for each class, there is a directed edge from x to its nearest neighbor among training set instances belonging to that class. Thus HMN represents a 'more specific' nearest neighbor relation, namely between points from *each* pair of classes. Exact computation of HMN has quadratic time complexity. This bound can be reduced by using metric trees or other spatial data structures (Grother et al., 1997).

We show that structural properties of HMN's correspond to properties of training instances related to the decision boundary of the 1-NN rule, such as being border or central point. These observations motivate the use of HMN for performing instance selection for the 1-NN rule. We introduce three new instance selection algorithms. The first algorithm, called HMN-C, discards instances corresponding to nodes of the HMN with no incoming edges (zero in-degree nodes). We prove that instance selection by means of this algorithm does not change the 1-NN classification of instances in the original training set. The second algorithm, called HMN-E, employs a more aggressive deletion strategy, removing a larger number of training instances, including those with zero in-degree. The last algorithm, called HMN-EI, applies iteratively HMN-E. These algorithms have the desirable properties of being order-independent and of having quadratic time complexity, which can be reduced using metric trees or other spatial data structures.

We assess effectiveness of the proposed algorithms with respect to generalization performance of the 1-NN rule and storage requirements, using 22 data sets with different characteristics, such as input dimension, cardinality, class balance, number of classes, noise content, and presence of redundant variables. Results of experiments show that HMN-EI improves significantly average accuracy of the 1-NN rule, and achieves significantly better performance than HMN-C and HMN-E. Experiments on the same data sets are conducted with the following three algorithms, which have been analyzed in Brighton and Mellish's paper on advances in instance selection (Brighton and Mellish, 2002). Edited Nearest Neighbor (E-NN), designed for noise reduction (Wilson, 1972), and two state-of-the-art editing algorithms: Iterative Case Filtering (ICF) (Brighton and Mellish, 1999) and the best of the Decremental Reduction Optimization algorithms introduced in Wilson and Martinez (1997) (DROP3). Comparison of the results shows that HMN-EI achieves best accuracy, with storage requirements similar to those of ICF and DROP3.

These results indicate that HMN's provide a powerful graph-based representation of training sets, with local structural graph properties useful for analyzing and enhancing 1-NN-based classification.

1.1 Related Work

Graphs have been successfully used for representing relations between points of a given data set, such as functional interaction between proteins (protein-protein interaction networks) or proximity (nearest neighbor graphs) (Dorogovtsev and Mendes, 2003). Graph representations in the context of

1-NN instance-based learning mainly use proximity graphs. Proximity graphs are defined as graphs in which points close to each other by some definition of closeness are connected (Barnett, 1976). The nearest neighbor graph (NNG) is a typical example of proximity graph, where each vertex is a data point that is joined by an edge to its nearest neighbor. The minimum spanning tree (MST) is also a proximity graph. Graph-based applications to instance-based learning algorithms mainly use the Gabriel graph (GG). Exact computation of the Gabriel graph is cubic in the number of nodes. Both the NNG and MST are subgraphs of the GG. The GG is a subgraph of the Delaunay Triangulation (DT), the dual of the Voronoi diagram. The Voronoi diagram and correspondingly the DT of a point set capture all the proximity information about the point set because they represent the original 1-NN rule decision boundary.

There are two main differences between the above proximity graphs and HMN's. First, HMN's explicitly use the class label of points in the definition of edges. As a consequence, while the above proximity graphs can be applied to any data set, HMN's are specifically defined for labeled data. Second, HMN's are directed graphs, while the above proximity graphs are not.

A class of directed proximity graphs, called class cover catch digraphs (CCCD's) has been introduced in Marchette et al. (2003), which provide a graph-based representation of one (target) class versus a different (non-target) class. In a CCCD of two such classes, nodes are the target instances and the maximal covering balls centered on each target instance, where a maximal covering ball of a target point is the ball centered in that point with maximum radius, which does not contain any non-target point. Each maximal covering ball is connected to its center by a directed edge.

CCCD's have been used for translating the so-called 'constrained class cover problem' (CCCP) to a problem on directed graphs. The CCCP amounts to find a minimum cardinality set of open covering balls with centers in target class points whose union covers the target class and does not contain any point of the non-target class.

The problem of finding an optimal solution to an instance of the CCCP has been shown to be equivalent to the one of finding a minimum cardinality dominating set in a general digraph. For CCCD's with points on Euclidean L_2 metric space, the problem can be solved in $O(n^m)$ time, with n and m equal to the number of target and non-target points, respectively. Further information about analysis of CCCD's and their application to classification can be found in DeVinney and Priebe (2005), DeVinney and Priebe (2006) and D.J. Marchette and Priebe (2005).

While both HMN's and CCCD are directed graphs, they describe different relations: HMN's describe the nearest neighbor relation between points of each pair of classes, while CCCD's describe the relation between maximal covering balls and target instances of one class.

Representations of a data set based on proximity graphs have been used to define algorithms for reducing the size of the training set (for instance, Bhattacharya, 1982), for removing noisy instances (for instance, Sánchez et al., 1997), and for detecting critical instances close to the decision boundary (for instance, Bhattacharya and Kaller, 1998), in order to improve storage and accuracy of 1-NN.

In particular, in Toussaint et al. (1984) a so-called Voronoi condensed data set is introduced, obtained by discarding all those points whose Voronoi cell shares a face with those cells that contain points of the same class. The 1-NN decision boundary is then characterized by the union of the common faces of the Voronoi diagram between Voronoi cell neighbors of different classes. The resulting instance selection algorithm produces a decision-boundary consistent set. Voronoi condensing does not reduce the number of points to a great extent and its computational complexity in higher dimensions is exponential in the number of dimensions (Toussaint et al., 1984).

Faster algorithms for instance selection based on the GG and the Reduced Neighborhood graph (Jaromczyk and Toussaint, 1992) have been proposed, for instance in Bhattacharya and Kaller (1998), Bhattacharya et al. (2005), Bhattacharya (1982), Mukherjee (2004) and Sánchez et al. (1997). In particular, in Bhattacharya et al. (2005) a specific data-structure for efficient computation of approximate Gabriel neighbor is proposed. Moreover, three instance selection algorithms are considered: Gabriel-Graph algorithm, ICF, and a so-called Hybrid. Hybrid incorporates E-NN, ICF, and the Gabriel graph rule. Specifically, it consists of the sequential application of a modified version of E-NN based on approximate Gabriel neighbor, a condensing step using Gabriel graph rule, and a filtering step of ICF. The authors provide a rather short discussion of results, and do not test the difference in quality of the average results of the algorithms.

For a thorough survey of graph-based methods for nearest neighbor classification, the reader is referred to Toussaint (2002).

The rest of the paper is organized as follows. After introducing the terminology used throughout the paper, the next section defines HMN's and discusses their properties. Section 3 presents a brief review of instance selection methods. Section 4 introduces HMN-C, HMN-E and HMN-EI. Section 5 describes experiments. Finally, in Section 6, we conclude and point to future work.

1.2 Terminology

The following notions and terms will be used in the sequel.
- X: a training set,
- $L = 1,\ldots,c$: class labels of X
- x: an element of X,
- $|X|$: the number of elements (cardinality) of X,
- X_i: the set of points of X with label i,
- $label(x)$: the class label of x,
- 1-NN(x,l): the nearest neighbor of x among those points (different from x) with label l,
- G: a directed graph with nodes representing elements of X,
- $e = (x,y)$: an edge of G, with x the vertex from which e is directed and y the vertex to which e is directed,
- $d(x)$: the number of edges where x occurs (the degree of x),
- $d(G)$: the total number of edges of G (the degree of G),
- *in-degree* of x: the number of edges pointing to x,

2. Hit Miss Networks

Suppose X consists of points from c different classes. In an HMN of X, a directed edge from point x to y is defined if y is the nearest neighbor of x in the class of y. Thus each point x has c outgoing edges, one for each class. When the classes of x and y are the same, we call x a *hit of* y, otherwise a *miss of* y . The name hit miss network is derived from these terms.

Definition 2.1 (Hit Miss Network) *The Hit Miss Network of* X, HMN(X), *is a directed graph* $G = (V,E)$ *with*

- $V = X$ *and*

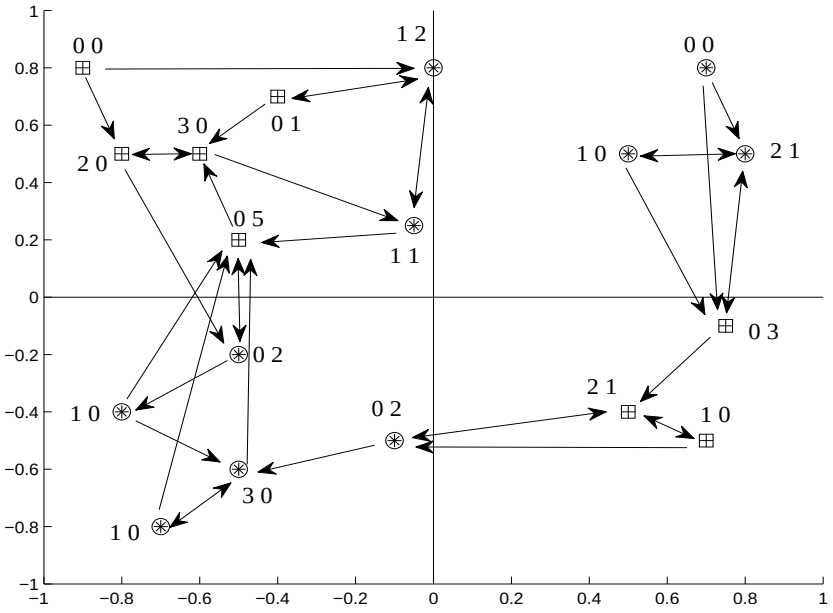


Figure 1: HMN graph of the training set for an artificial classification problem. Hit- and miss-degree of each node is written on the left and right side of the node, respectively.

- $E = \{(x, \texttt{1-NN}(x,l)) \textit{ for each } x \in X \textit{ and } l \in L\}$.

Definition 2.2 (Hit, Miss Points) *Let $G = \texttt{HMN}(X)$. A hit of x (respectively, miss of x) is any point y such that $e = (y,x)$ is an edge of G and $label(y) = label(x)$ (respectively, $label(y) \neq label(x)$).*

We call *hit-degree* (respectively *miss-degree*) of x the number of hit (respectively miss) nodes of x. $Hit(x)$ (respectively $Miss(x)$) denotes the set of hit (respectively miss) nodes of x.

Figure 1 shows the HMN of the training set for a toy binary classification task. Observe that the two points with zero in-degree are relatively isolated and far from points of the opposite class, while points with high miss-degree are closer to points of the opposite class and to the 1-NN decision boundary.

Computing HMN requires quadratic time complexity in the number of points. Nevertheless, by using metric trees or other spatial data structures this bound can be reduced. For instance, using *kd* trees, whose construction takes time proportional to $n\log(n)$, nearest neighbor search exhibits approximately $O(n^{1/2})$ behavior (Grother et al., 1997). A recent fast all nearest neighbor algorithm for applications involving large point-clouds is introduced in Sankaranarayanan et al. (2007).

By construction, the degree of G, and the degree $d(x)$ of a node $x \in V$ satisfy the following properties:

$$d(G) = c \cdot |X|$$

and

$$c \leq d(x) \leq |X| + c - 1.$$

HMN's describe the nearest neighbor relation over pairs of points from *each pair of classes* of the training set. Formally, it is easy to check that

$$\mathtt{HMN}(X) = \cup_{i,j,i\neq j,i,j\in[1,c]} \mathtt{HMN}(X_i \cup X_j).$$

Therefore, the HMN's of pairs of classes can be constructed independently, supporting parallel execution. Moreover, if a new class is added, one does not need to reconstruct the entire HMN, but the HMN, between the new class and each of the other ones.

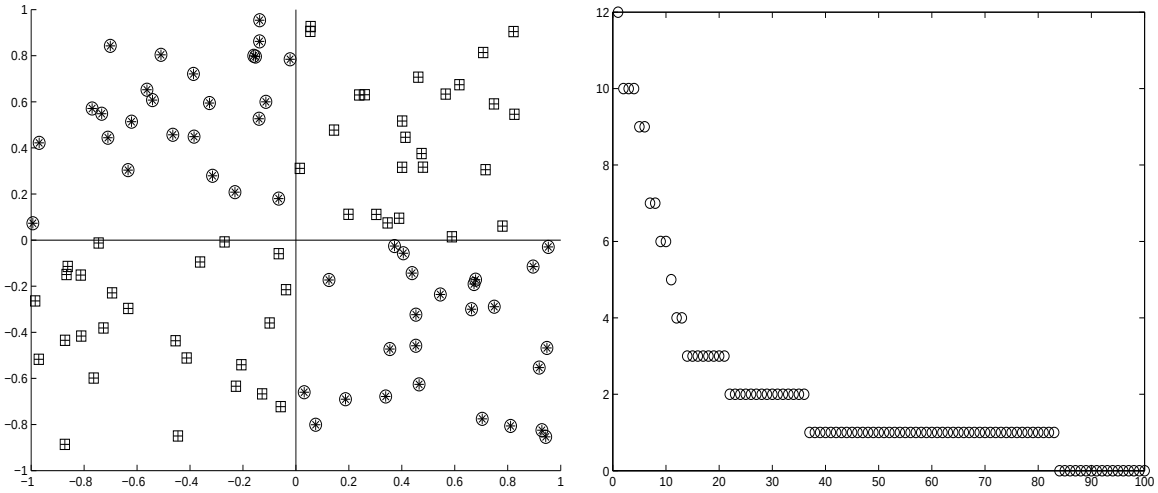


Figure 2: A XOR problem data set (left) and plot of sorted in-degrees (y-axis) of nodes (x-axis), in decreasing order, of the corresponding HMN graph (right).

Figure 2 shows a training set for a XOR classification task, and the sorted in-degrees of its HMN graph. The in-degree distribution seems to follow a Power law, where very few nodes have high in-degree. If we randomly permute the class labels of the training set then the degree distribution changes, with lower in-degree values and more nodes having small in-degree (cf., Figure 3).

These observations indicate that the local structure of HMN provides information about properties of the training points, and motivate us to use HMN's for defining a new instance selection technique. Before that, in the next section we review briefly instance selection algorithms.

3. Instance Selection Algorithms

In instance-based learning, the training set is stored, and the machine learning algorithm computes a distance between the new instance and the stored ones in order to classify new instances. In particular, in the one nearest neighbor algorithm (1-NN) the class label of a new instance is the one of the stored instance with minimum distance.

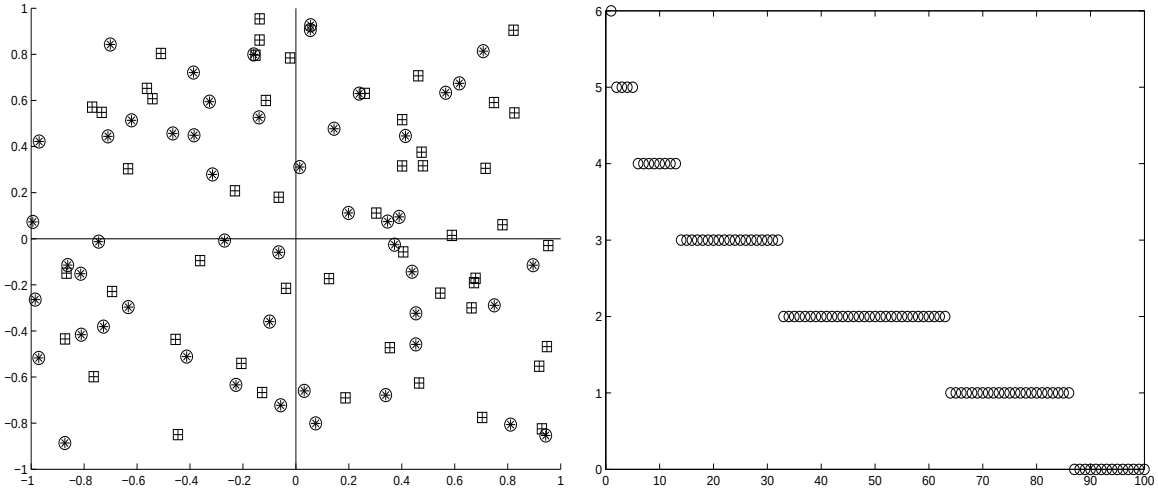


Figure 3: Training points of a XOR problem data set with labels randomly permuted (left figure) and plot of in-degrees, sorted in decreasing order, obtained by applying `HMN` (right figure).

Instance selection techniques, here also called *editing* techniques, select a subset of the training set in order to improve the storage and possibly the generalization performance of an instance-based learning algorithm. In this paper we focus on the `1-NN` classifier.

Research on instance selection started with the seminal work of Hart (1968). Subsequent research focussed mainly on three types of training set condensation techniques (Brighton and Mellish, 2002): competence preservation, competence enhancement, and hybrid approaches.

- *Competence preservation* algorithms compute a *training set consistent subset* by removing irrelevant points that do not affect the classification accuracy of the training set (see for instance Angiulli, 2007; Dasarathy, 1994).
- *Competence enhancement* methods remove noisy points in order to increase classifier accuracy. Noise reduction techniques can remove exception instances or border instances which cannot be distinguished from true noise by the technique, hence can possibly affect negatively the generalization performance of the classifier that uses only the selected instances (see for instance Vezhnevets and Barinova, 2007; Wilson, 1972).
- *Hybrid* methods aim at finding a subset of the training set that is both noise free and does not contain irrelevant points (for instance, Brighton and Mellish, 2002; Wilson and Martinez, 1997). Alternative methods use prototypes instead of instances of the training set (see for instance Pekalska et al., 2006).

In Wilson and Martinez (2000), Wilson and Martinez present a comprehensive survey of concepts and issues related to reduction techniques for instance-based learning algorithms, including a thorough experimental comparison of algorithms. Other, more recent surveys of instance selection techniques are Brighton and Mellish (2002), Jankowski and Grochowski (2004a) and Jankowski and Grochowski (2004b).

In particular, in Brighton and Mellish (2002) the authors compare experimentally Edited Nearest Neighbor (E-NN) and the state-of-the-art editing algorithms Iterative Case Filtering (ICF) and Decremental Reduction Optimization Procedure 3 (DROP3). E-NN is an algorithm generally considered in comparative experimental analysis of editing methods mainly because it provides useful information on the amount of 'noisy' instances contained in the considered data sets, and on the improvement of accuracy obtained by their removal. Iterative Case Filtering uses E-NN as pre-processing noise reduction step, followed by an iterative procedure for deleting 'superfluous points'. Also DROP3 begins with the application of a simple noise reduction step, followed by another simple type of heuristic for discarding 'superfluous points'.

Results of an extensive comparative experimental analysis performed in Wilson and Martinez (2000) and in Brighton and Mellish (2002) indicate that ICF and DROP3 are cutting-edge instance selection algorithms, achieving best K-NN accuracy and storage reduction on a large number of learning tasks over many other editing methods. These algorithms, together with E-NN, are described in more detail below and used to assess comparatively the performance of the HMN-based editing algorithms we propose.

3.1 Edited Nearest Neighbor

Wilson (1972) introduced the *Edited Nearest Neighbor* (E-NN), where each point x is removed from X if it does not agree with the majority of its K nearest neighbors. This editing rule removes noisy points as well as points close to the decision boundary, yielding to smoother decision boundaries.

3.2 Iterative Case Filtering

In Brighton and Mellish (1999) the *Iterative Case Filtering* algorithm (ICF) was proposed, which first applies E-NN iteratively until it cannot remove any point, and next iteratively removes other points as follows. At each iteration, all points for which the so-called *reachability* set is smaller than the *coverage* one are deleted. The reachability of a point x consists of the points inside the largest hyper-sphere containing only points of the same class as x. The *coverage* of x is defined as the set of points that contain x in their reachability set.

3.3 Decremental Reduction Optimization

The family of Decremental Reduction Optimization (DROP) algorithms was first introduced by Wilson and Martinez (1997), and further extended and analyzed in Wilson and Martinez (2000). It consists of five algorithms DROP1-5. DROP1 is the basic removal rule, which removes a point x from X if the accuracy of the K-NN rule on the set of its associates does not decrease. Each point has a list of K nearest neighbors and a list of associates, which are updated each time a point is removed from X. A point y is an *associate* of x if x belongs to the set of K nearest neighbors of y. If x is removed then the list of K nearest neighbors of each of its associates y is updated by adding a new neighbor point z, and y is added to the list of associates of z. Moreover, for each of the K nearest neighbors y of x, x is removed from the list of associates of y.

DROP2 is obtained from DROP1 by discarding the last update step, hence it considers all associates in the entire training set when testing accuracy performance in the removal rule. Moreover, the removal rule is applied to the points sorted in decreasing order of distance from their nearest

neighbor from the other classes (nearest enemy). In this way, points furthest from their nearest enemy are selected first.

DROP3 applies a pre-processing step which discards points of X misclassified by their K nearest neighbors, and then applies DROP2.

DROP4 uses a stronger pre-processing step which discards points of X misclassified by their K nearest neighbors if their removal does not hurt the classification of other instances.

Finally DROP5 modifies DROP2 by considering the reverse order of selection of points, in such a way that instances are considered for removal beginning with instances that are nearest to their nearest enemy.

DROP3 achieves the best mix of storage reduction and generalization accuracy of the DROP methods (see Wilson and Martinez, 2000). Moreover, results of experiments conducted in Wilson and Martinez (1997, 2000) show that DROP3 achieves higher accuracy and smaller storage requirements than several other methods, such as CNN (Hart, 1968), SNN (Ritter et al., 1975), E-NN (Wilson, 1972), the All k-NN method (Tomek, 1976), IB2, IB3 (Aha et al., 1991), and the Explore method (Cameron-Jones, 1995). Therefore we use DROP3 and ICF as representatives of the state-of-the-art, in order to assess the performance of the HMN-based editing algorithms introduced in the following section.

4. Instance Selection with Hit Miss Networks

Zero in-degree nodes of HMN(X) include relatively isolated points, and points not too close to the decision boundary. This is illustrated in the HMN-C sub-plot of Figure 5, where zero in-degree nodes of the HMN for a XOR data set are highlighted in bold.

Zero in-degree nodes can be safely removed from X without affecting 1-NN classification of the original training. Formally, we have the following result.

Proposition 4.1 *Let S be obtained by removing from X all points with zero in-degree. Then S is a decision-boundary consistent subset.*

Proof

Suppose there exists $x \in X$ s.t. 1-NN$(x,X) = y$, 1-NN$(x,S) = y_1$ and $l(y) \neq l(y_1)$. Then $y \neq y_1$ and y has been removed. So y has in-degree equal to 0.

From 1-NN$(x,X) = y$ it follows that x is in $Hit(y)$ or in $Miss(y)$, hence the in-degree of y is at least 1, which yields a contradiction.

Then $l(y) \neq l(y_1)$ was false. Hence S is a training set consistent subset.

We call HMN-C (HMN for training set Consistent instance selection) the algorithm that removes from the training set all instances with zero in-degree.

HMN-C does not remove noisy instances, which are in general close to the class decision boundary. Therefore, a more aggressive removal strategy is adopted in the following instance selection heuristic algorithm, called HMN-E (HMN for Editing), which compares the hit- and miss-degree of each node for deciding whether to remove it.

Pseudo-code of this algorithm is given in Figure 1. HMN-E is based on four if-then rules, described below.

(1) **compute** HMN(X)
(2) **for** x in X
(3) **if** $w_{l(x)} * |Miss(x)| + \varepsilon > (1 - w_{l(x)}) * |Hit(x)|$
(4) **flag** x for removal (rule R1)
(5) **end if**
(6) **end for**
(7) $X_{R1,remove} = \{x \in X$ with **flag** for removal$\}$
(8) **for** l in $1 \ldots c$
(9) $Left_l = \{x \notin X_{R1,remove} \mid l(x) = l\}$
(10) **if** $|Left_l| < 4$
(11) **unflag** $\{z \in X_{R1,remove} \mid l(z) = l,$ *in-degree*$(z) > 0\}$ (rule R2)
(12) **end if**
(13) **end for**
(14) **for** x in $X_{R1,remove}$
(15) **if** $c > 3$ and $|Miss(x)| < c/2$ and *in-degree*$(x) > 0$
(16) **unflag** x (rule R3)
(17) **end if**
(18) **if** $|Hit(x)| \geq |X_{l(x)}|/4$
(19) **unflag** x (rule R4)
(20) **end if**
(21) **end for**
(22) **remove** from X all x with **flag** for removal

Figure 4: Pseudo-code of HMN-E algorithm. Input: training set X. Output: subset of X.

1. The first rule removes x if its miss-degree is greater or equal than its hit-degree, that is $|Miss(x)| \geq |Hit(x)|$. This amounts to discard a point when it is isolated (that is, has zero in-degree), as well as when it has more 'miss' than 'hit' points.

   In order to deal with unbalanced data sets, the terms of the inequality are weighted by the fraction of points of the same and other classes, respectively, resulting in rule R1 (lines 3-5 in Figure 1) which removes a point x from X if

   $$w_{l(x)} * |Miss(x)| + \varepsilon > (1 - w_{l(x)}) * |Hit(x)|, \tag{1}$$

   where $w_{l(x)} = |\{z \mid l(z) = l(x)\}|/|X|$ and $\varepsilon < 1$ ($\varepsilon = 0.1$ is used in our experiments).

2. On small data sets, application of R1 could remove too many points of one class. Rule R2 (lines 10-12 in Figure 1) handles this case. It checks if the size of a class becomes too small after application of R1. In such a case all points of that class having positive in-degree are added. The threshold used in the rule is set in such a way that the minimum size of a condensed class becomes equal to 4. We consider this to be a reasonable class storage lower bound for the condensed 1-NN rule.

3. Suppose for simplicity each class has equal size ($|X|/c$). Then $|Miss(x)| \leq \frac{(c-1)^2|X|}{c}$, and $|Hit(x)| \leq |X|/c - 1$. This $|Miss(x)|$'s upper bound grows linearly with the number c of

classes, while the $|Hit(x)|$'s upper bound depends on c in an inversely linear way. Therefore $|Miss(x)|$ is more likely to grow faster than $|Hit(x)|$ in the presence of many classes. This justifies the introduction of the heuristic Rule R3 (lines 15-17 in Figure 1), which deals with data sets having more than three classes. For more than three classes, a point x with in-degree greater than 0 is added if it has a low number of 'miss' points, low with respect to c. Here we use as threshold half of the total number of classes.

4. Points with many 'hits' are closer to the 'centroid' of the class, hence are considered to be relevant for discriminating the classes, even when they are close to points of other classes. This case is implemented in rule R4 (lines 18-20 in Figure 1) which adds x if it is the 'hit' of at least 25% of the points of its class.

Rules R2 - R4 are 'rules of thumb'. The threshold in each of these rules has been fixed to a value considered reasonable, and has not been tuned on each specific data set. These rules could be improved by means of parameter tuning or domain knowledge on the specific data distribution of the learning task.

In order to remove more "redundant" points, HMN-E can be applied iteratively as follows: repeat the application of HMN-E until the generalization accuracy of 1-NN on the original training set with the reduced set decreases. We call this algorithm HMN-E *Iterated* (HMN-EI).

Observe that the three HMN-based editing algorithms are order independent, that is, their output does not depend on the order in which training points are processed. Moreover, by construction, points removed by HMN-C are also removed by HMN-E, and points removed by HMN-E are also removed by HMN-EI.

4.1 Comparison of the Methods on the XOR Problem

Figure 5 shows application of the considered editing algorithms to the training set of a XOR classification task. Points removed by an algorithm are shown in bold. As expected, points removed by E-NN are close to the decision boundary. ICF and DROP3 delete also 'safe' points far from the decision boundary (in order to enhance storage requirements). HMN-C removed points 'locally' isolated, while HMN-E removes also 'safe' points as well as points close to the decision boundary. Its iterated version HMN-EI selects very few points far from the decision boundary. The figures do not show any other apparent set-theoretic relationship between the subsets of points removed by the methods.

Figure 6 plots the sorted in-degrees of the considered XOR training set, where in-degree of points removed by a method are marked with triangles. As expected, points removed by ICF and not already deleted by E-NN have low in-degree. The majority of points removed by E-NN have high in-degree, showing the tendency of 'noisy' points to have high in-degree. HMN-E removes more points with high in-degree than E-NN, and it selects points with low, but not zero, degree. While HMN-EI selects only points with in-degree 1 and 2, ICF and DROP3 select also points of higher degree. On this example, HMN-EI achieves best storage reduction.

5. Experiments

The following seven algorithms are considered: 1-NN (no instance selection), HMN-C, HMN-E, HMN-EI, E-NN, ICF, and DROP3. In order to assess their comparative performance, we implemented the above

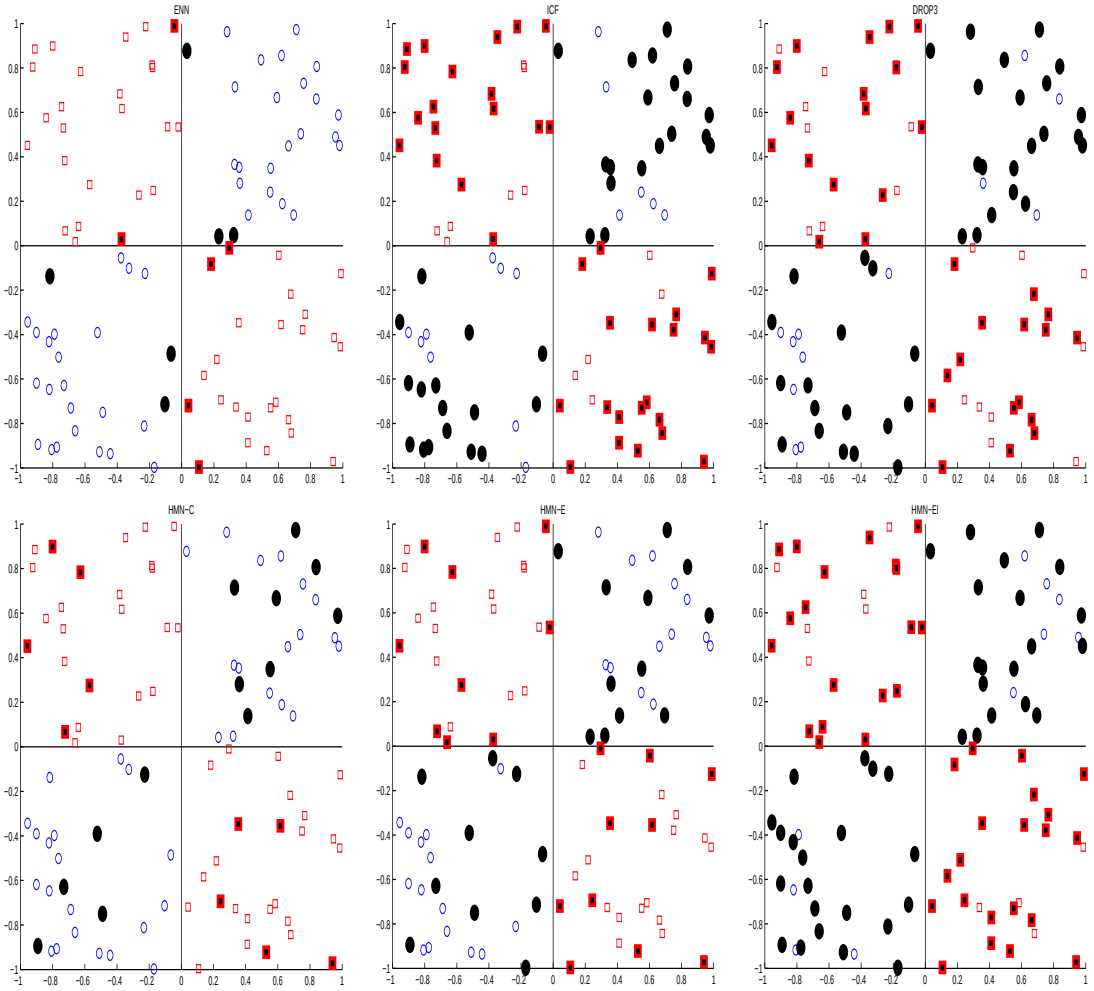


Figure 5: Effect of the algorithms on a XOR problem training set: removed points are shown with filled markers. Top row, from left to right: `E-NN`, `ICF`, `DROP3`. Bottom row, from left to right: `HMN-C`, `HMN-E` and `HMN-EI`.

algorithms and conducted extensive experiments on 22 Machine Learning benchmark data sets. All algorithms are tested using one neighbor.

The performance measures here used are (average) test accuracy of the classifier and (average) percentage of the training set removed by the method.

5.1 Data Sets

The following 22 publicly available benchmark data sets used in previous studies on model selection for (semi)supervised learning, are considered.

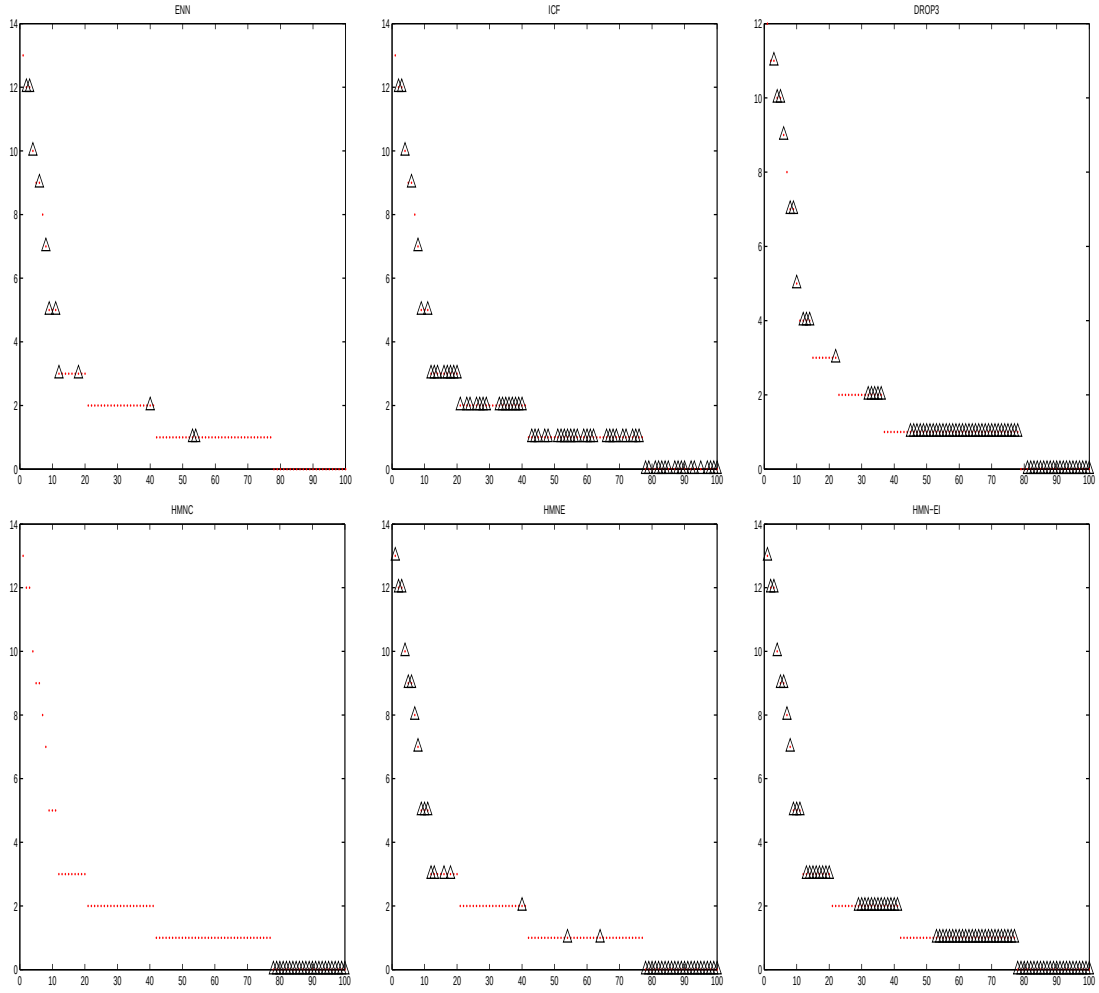


Figure 6: In-degree of nodes of the HMN built on the considered XOR training set, sorted in decreasing order. The in-degree of points removed by an algorithm are marked with triangles. Top row, from left to right: E-NN, ICF, DROP3. Bottom row, from left to right: HMN-C, HMN-E and HMN-EI.

1. Raetsch's binary classification benchmark data sets have been used in Rätsch et al. (2001): they consists of 1 artificial and 12 real-life data sets from the UCI, DELVE and STATLOG benchmark repositories.

   For each experiment, the 100 (20 for Splice and Image) partitions of each data set into training and test set available in the repository are here used.

2. Chapelle's benchmark data sets used in Chapelle and Zien (2005) are from two artificial binary classification and three real-life multi-class classification problems. Specifically, g50c and g10n are generated from two standard normal multi-variate Gaussians. In g50c, the labels correspond to the Gaussians, and the means are located in 50-dimensional space such that the

Bayes' error is 5%. In contrast, g10n is a deterministic problem in 10 dimensions, where the decision function traverses the centers of the Gaussians, and depends on only two of the input dimensions.

The three real world data sets are Coil20, consisting of gray-scale images of 20 different objects taken from different angles, in steps of 5 degrees, Uspst, the test data part of the USPS data on handwritten digit recognition, and Text consisting of the classes 'mac' and 'mswindows' of the Newsgroup20 data set.

For each experiment, the 10 partitions of each data set into training and test set available in the repository are used.

3. Finally, we consider four standard benchmark data sets from the UCI Machine Learning repository: Iris, Bupa, Pima, and Breast-W.

   For each experiment, 100 partitions of each data set into training and test set are used. Each partition randomly divides the data set into training and test set, equal to 80% and 20% of the data, respectively.

Thus the benchmark data consists of 3 artificial data sets (Banana, g50c, g10n) and 19 real-life ones, with different characteristics as shown in Table 1. In particular, Chapelle's data sets are balanced, that is, all classes are represented by similar number of points, while some of Raetsch's data sets are rather unbalanced.

5.2 Results

Cross validation is applied to each data set. For each partition of the data set, each editing algorithm is applied to the training set X from which a subset S is returned. The one nearest neighbor classifier that uses only points of S is applied to the test set. The average accuracy on the test set over the given partitions is reported for each algorithm (cf., Table 2, Table 3). The average percentage of instances that are excluded from S is also reported under the column with label R. Average and median accuracy and training set reduction percentage for each algorithm over all the 22 data sets is reported near the bottom of the Table.

We compare statistically HMN-EI with each of the other algorithms as follows.

- First a paired t-test on the cross validation results on each data set is applied, to assess whether the average accuracy for HMN-EI is significantly different than each of the other algorithms. In Tables 2, 3 a '+' indicates that HMN-EI's average accuracy is significantly higher than the other algorithm at a 0.05 significance level. Similarly, a '-' indicates that HMN-EI's average accuracy is significantly lower than the other algorithm at a 0.05 significance level. The row labeled 'Sig.acc.+/-' reports the number of times HMN-EI's average accuracy is significantly better and worse than each of the other algorithms at a 0.05 significance level. A paired t-test is also applied to assess significance of differences in storage reduction percentages for each experiment.

- Second, in order to assess whether differences in accuracy and storage reduction on all runs of the entire group of data sets are significant, a non-parametric paired test, the Wilcoxon Signed Ranks test[1] is applied to compare HMN-EI with each of the other algorithms. A '+'

1. We used 'wilcoxon' Matlab routine by G. Cardillo.

Data Set	CL	VA	TR	Cl.Inst.	TE	Cl.Inst.
Banana	2	2	400	212-188	4900	2712-2188
B.Cancer	2	9	200	140-60	77	56-21
Diabetis	2	8	468	300-168	300	200-100
German	2	20	700	478-222	300	222-78
Heart	2	13	170	93-77	100	57-43
Image	2	18	1300	560-740	1010	430-580
Ringnorm	2	20	400	196-204	7000	3540-3460
F.Solar	2	9	666	293-373	400	184-216
Splice	2	60	1000	525-475	2175	1123-1052
Thyroid	2	5	140	97-43	75	53-22
Titanic	2	3	150	104-46	2051	1386-66
Twonorm	2	20	400	186-214	7000	3511-3489
Waveform	2	21	400	279-121	4600	3074-1526
g50	2	50	550	252-248	50	23-27
g10n	2	10	550	245-255	50	29-21
Coil20	20	1024	1440	70	40	2
Text	2	7511	1946	959-937	50	26-24
Uspst	10	256	2007	267-201-169-192-137 -171-169-155-175	50	6-5-9-4-3-3-4-5-5
Iris	3	4	120	40-40-40	30	10-10-10
Bupa	2	6	276	119-157	69	26-43
Pima	2	8	615	398-217	153	102-51
Breast-W	2	9	546	353-193	137	91-46

Table 1: Data Sets used in the experiments. Raetsch's benchmark repository available at `http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm`. Chapelle's one at `http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/lds/`. Four popular benchmark data sets from UCI Machine Learning repository available at `http://mlearn.ics.uci.edu/MLRepository.html`. CL = number of classes, TR = training set, TE = test set, VA = number of variables, Cl.Inst. = number of instances in each class.

(respectively '-') in the row labeled 'Wilcoxon' indicates that `HMN-EI` is significantly better (respectively worse) than the other algorithm.

Results of Table 2 show that `HMN-EI` achieves best generalization accuracy, significantly better than the one of `1-NN` and of `HMN-C`. Moreover, `HMN-EI` outperforms significantly `HMN-E` with respect to storage requirements and achieves similar generalization performance. For these reasons, `HMN-EI` is chosen for further comparison with state-of-the-art editing algorithms.

5.3 Comparison with Other Algorithms

From the results of the experiments reported in Table 3 we derive the following observations.

Data Set	1-NN	HMN-C	R	HMN-E	R	HMN-EI	R
Banana	86.4 +	85.6 +	19.7 +	88.2 +	38.5 +	**88.6**	57.9
B.Cancer	67.3 +	65.9 +	20.1 +	66.1 +	50.0 +	**69.2**	72.8
Diabetis	69.9 +	68.6 +	22.4 +	72.5 +	53.1 +	**73.5**	73.1
German	70.5 +	69.4 +	26.0 +	72.5	56.4 +	**72.9**	75.5
Heart	76.8 +	76.1 +	23.7 +	**81.7**	52.9 +	81.6	79.3
Image	**96.6** -	96.1 -	23.7 +	94.8 -	41.1 +	92.7	57.3
Ringnorm	65.0 +	63.4 +	33.5 +	**66.6** -	63.9 +	65.6	82.9
F.Solar	60.8 +	60.5 +	80.1 +	63.5 +	86.9 +	**64.7**	92.1
Splice	71.1 -	70.1 +	46.0 +	**72.3** -	71.7 +	70.7	86.6
Thyroid	**95.6** -	94.9 -	24.2 +	93.4	38.9 +	93.2	59.1
Titanic	67.0 +	66.9 +	79.6 +	70.9 +	84.9 +	**76.0**	94.7
Twonorm	93.3 +	92.8 +	39.4 +	95.7	60.4 +	**95.9**	83.5
Waveform	84.2 +	83.6 +	36.2 +	**86.0** -	58.0 +	85.4	79.9
g50c	79.6 +	80.2 +	42.7 +	**87.4** -	71.0 +	86.8	88.3
g10n	75.0 +	74.6 +	26.0 +	75.8 +	63.5 +	**79.2**	82.5
Coil20	**100** -	**100** -	6.7 +	**100** -	10.4 +	99.5	15.0
Text	**92.8** -	90.8 -	16.7 +	89.4 -	54.1 +	86.4	78.9
Uspst	**94.6** -	**94.6** -	12.5 +	94.4 -	20.3 +	93.6	29.8
Iris	**95.5**	95.0	24.7 +	95.1	38.7 +	95.4	75.2
Breast-W	95.7 +	95.5 +	50.7 +	**97.1**	54.9 +	96.9	71.8
Bupa	61.6 +	59.5 +	18.5 +	63.4 +	54.7 +	**64.5**	76.0
Pima	67.8 +	66.5 +	21.4 +	70.8 +	50.8 +	**71.7**	68.1
Average	80.3	79.6	31.6	81.7	53.4	**82.2**	**71.8**
Median	78.2	78.1	24.5	**83.9**	54.4	83.5	**75.8**
Sig.+/-	15/6	16/5	22/0	8/8	22/0	n/a	n/a
Wilcoxon	+	+	+	$\sim$	+	n/a	n/a

Table 2: Results of experiments on ML benchmark data sets. Each column labeled with the name of an algorithm reports its average test set accuracy on each data set. R = percentage of training points removed. Best results are shown in bold. Average (Median) = average (median) results over data sets. Sig.+/- = number of times HMN-EI average accuracy (storage reduction) is significantly better (+) or significantly worse (-) than the other algorithm, according to a paired t-test at 0.05 significance level. Wilcoxon = a '+' indicates HMN-EI significantly better than the other algorithm at a 0.01 significance level according to a Wilcoxon test for paired samples, $\sim$ indicates no significant difference.

- On the g50c data set, HMN-EI achieves highest average accuracy, significantly better than that of all other methods. With an average error of about 13%, close to twice the Bayes probability of error, HMN-EI performs almost optimally, and discards about 88% of the training data. This shows effectiveness and robustness of this method with respect to the presence of noise (on this type of classification task).

Data Set	HMN-EI	R	ICF	R	E-NN	R	DROP3	R
Banana	**88.6**	57.9	86.1 +	**79.2** -	87.8 +	13.1 +	87.6 +	68.2 -
B.Cancer	69.2	72.8	67.0 +	**79.0** -	69.4	33.3 +	**69.7** -	72.9
Diabetis	**73.5**	73.1	69.8 +	**83.1** -	72.6 +	30.3 +	72.3 +	73.4
German	72.9	75.5	68.6 +	**82.2** -	**73.0**	30.1 +	72.0 +	74.3 +
Heart	**81.6**	79.3	76.7 +	**80.9** -	80.6 +	23.1 +	80.2 +	72.1 +
Image	92.7	57.3	93.8	**80.3** -	**95.8** -	3.4 +	95.1 -	64.9 -
Ringnorm	**65.6**	82.9	61.2 +	**85.5** -	54.8 +	35.3 +	54.7 +	80.6 +
F.Solar	**64.7**	92.1	61.0 +	52.0 +	61.3 +	39.8 +	61.4 +	**93.8** -
Splice	**70.7**	**86.6**	66.3 +	85.5 +	68.4 +	28.3 +	67.6 +	79.01 +
Thyroid	**93.2**	59.1	91.9 +	**85.6** -	94.0 -	4.0 +	92.7 +	65.7 -
Titanic	**76.0**	**94.7**	67.5	54.3 +	67.3 +	33.0 +	67.7 +	94.3
Twonorm	**95.9**	83.5	89.2 +	**90.7** -	94.1 +	6.4 +	94.3 +	72.7 +
Waveform	**85.4**	79.9	82.1	**86.8** -	**85.4**	15.7 +	84.9 +	73.6 +
g50c	**86.8**	**88.3**	82.2 +	56.3 +	82.2 +	19.7 +	82.8 +	77.7 +
g10n	**79.2**	**82.5**	73.0 +	53.9 +	74.0 +	22.8+	75.0 +	71.4 +
Coil20	99.5	15.0	98.5 +	**42.6** -	**100** -	0.0 +	95.5 +	64.4 -
Text	86.4	**78.9**	88.2 -	68.8 +	**91.6** -	7.7 +	88.0 -	66.7 +
Uspst	93.6	29.8	86.2	**87.8** -	**94.0**	4.7 +	91.4 +	67.3 -
Iris	95.4	**75.2**	95.3	69.7 +	**95.9** -	4.2 +	95.8 -	66.4 +
Breast-W	**96.9**	71.8	95.4 +	9**3.8** -	96.6	4.1 +	96.8	74.2 -
Bupa	**64.5**	**76.0**	60.9 +	74.3 +	63.2 +	38.1+	63.1 +	73.8 +
Pima	**71.7**	68.1	67.9 +	**78.7** -	69.7 +	32.4 +	69.4 +	73.3 -
Average	**82.0**	71.8	78.6	**75.0**	80.5	19.5	79.9	**73.7**
Median	**83.5**	75.8	79.4	**79.8**	81.4	21.25	81.5	73.1
Sig.+/-	n/a	n/a	16/2	7/15	12/5	22/0	17/4	11/8
Wilcoxon	n/a	n/a	+	$\sim$	+	+	+	$\sim$

Table 3: Results of experiments on ML benchmark data sets of HMN-EI, ICF, Wilson's editing, and DROP3.

- On the g10n data set, HMN-EI achieves significantly better performance than that of the other methods, indicating robustness to the presence of irrelevant variables (on this type of classification task).
- On data sets with more than three classes, HMN-EI has worse storage requirements than the other algorithms, but also generally higher accuracy, due to the more conservative editing strategy (Rule 3) HMN-EI uses on data sets with many classes.
- Results of a paired t-test at a 0.05 significance level shows better accuracy performance of HMN-EI over ICF, E-NN and DROP3 (cf., row Sig.+/-) on 15, 12, and 17 of the data sets, and worse accuracy on 2, 5, and 4 data sets, respectively. Storage reduction of HMN-EI is 7, 22, and 11 times better, and 15, 0, and 8 worse, indicating better storage performance of

ICF, according to this test. As shown, for instance, in Demsar (2006), comparison of the performance of two algorithms based on the t-test is only indicative because the assumptions of the test are not satisfied, and the Wilcoxon test is shown to provide more reliable estimates.

- Results of the non parametric Wilcoxon test for paired samples at a 0.01 significance level indicate that the performance of HMN-EI on the entire set of classification tasks is significantly better than each one of the other algorithms with respect to accuracy, and that there is no significant difference in storage reduction between HMN-EI and state-of-the-art editing algorithms (cf., last row of the table).

- The three best performing instance selection algorithms, DROP3, ICF and HMN-EI have quadratic computational complexity in the number of instances (which can be reduced by using ad-hoc data structures such as kd-trees). ICF and HMN-EI are in principle slower than the other algorithms, due to their multiple passes over (selected) instances. Nevertheless, in our experiments these algorithms require a small number of iterations (about 7 for ICF and 3 for HMN-EI). Thus their computational complexity is not significantly worse than that of DROP3.

In summary, results of these experiments indicate effectiveness of HMN-based instance selection and robustness of HMN-EI with respect to the presence of high number of variables, training examples, multiple classes, noise and irrelevant variables. Comparison with results obtained by E-NN, ICF and DROP3 shows improved average accuracy and similar storage requirement of HMN-EI, ICF and DROP3 on these data sets.

6. Conclusions and Future Work

This paper proposed a new graph-based representation of a training set and showed how local structural properties of nodes provide information about the closeness of the corresponding points to the decision boundary of the 1-NN rule. We formalized these properties by means of the notions of *Hit* and *Miss* set, and used such notions for defining three algorithms for 1-NN's instance selection. We proved that HMN-C removes instances without affecting the accuracy of the 1-NN rule on the original training set (it computes a decision-boundary consistent subset). We showed that HMN-E and HMN-EI remove more points than HMN-C, including those close to the decision boundary. Results of extensive experiments indicated that HMN-EI significantly improves the generalization accuracy of 1-NN and reduces significantly its storage requirements.

We compared experimentally HMN-EI with a popular noise reduction algorithm (E-NN), and two state-of-the-art editing algorithms (ICF and DROP3). Results of extensive experiments on 19 real-life data sets and 3 artificial ones showed that HMN-EI achieved best average accuracy, and storage reduction similar to that of ICF and DROP3. This indicates that simple local topological properties of the proposed graph-based data set representation provide an effective tool for 1-NN's instance selection.

The design of condensing algorithms could also be based on an extension of HMN for describing the K-nearest neighbor relation between each pair of classes. We conducted preliminary experiments to investigate whether using more than one neighbor to classify new points affects the accuracy performance of the condensing algorithms here considered. Results of experiments on seven UCI ML data setdata sets, using 3 and 5 neighbors for classifying new points, showed that HMN-EI still achieves best average accuracy. In general, the generalization performance increased (of about 1%, 2%) when 3 and 5 neighbors were used.

In this paper we use only the degree of nodes as mean for analyzing a training set in order to improving 1-NN's performance. It would be interesting to investigate whether other graph-theoretical properties of HMN's, such as information on path distance, clustering coefficient and diameter, provide useful information for studying and improving the 1-NN's performance.

Other future work includes the use of HMN's to tackle the following interesting problems: measuring the difficulty of a learning task with respect to a given training set (see for instance Zighed et al., 2002); enhancing classification techniques based on a notion of margin, such as Support Vector Machines (see for instance Shin and Cho, 2007); improving Boosting algorithms by means of editing techniques (see for instance Vezhnevets and Barinova, 2007), and, more generally, tackling over-fitting in supervised learning.

Acknowledgments

Many thanks to the Reviewers and to the Editor Leon Bottou for their constructive comments.

References

D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6: 37–66, 1991.

F. Angiulli. Fast nearest neighbor condensation for large data sets classification. *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1450–1464, 2007. ISSN 1041-4347.

V. Barnett. The ordering of multivariate data. *J. Roy. Statist. Soc.*, Ser. A 139(3):318–355, 1976.

B. Bhattacharya and D. Kaller. Reference set thinning for the k-nearest neighbor decision rule. *Proceedings of the 14th International Conference on Pattern Recognition*, (1):238–243, 1998.

B. Bhattacharya, K. Mukherjee, and G. Toussaint. Geometric decision rules for instance-based learning problems. In *Proceedings of the 1st International Conference on Pattern Recognition and Machine Intelligence (PReMI'05)*, number LNCS 3776, pages 60–69. Springer, 2005.

B.K. Bhattacharya. Application of computational geometry to pattern recognition problems. *PhD Thesis. Simon Fraser University, School of Computing Science, Technical Report*, TR 82-3, 1982.

H. Brighton and C. Mellish. On the consistency of information filters for lazy learning algorithms. In *PKDD '99: Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, pages 283–288. Springer-Verlag, 1999.

H. Brighton and C. Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, (6):153–172, 2002.

R.M. Cameron-Jones. Instance selection by encoding length heuristic with random mutation hill climbing. In *Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence*, pages 99–106, 1995.

O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 57–64, 2005.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

B. V. Dasarathy. Minimal consistent set (mcs) identification for optimal nearest neighbor decision systems design. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3):511–517, 1994.

J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

J. DeVinney and C.E. Priebe. The use of domination number of a random proximity catch digraph for testing spatial patterns of segregation and association. *Statistics and Probability Letters*, 73 (1):37–50, 2005.

J. DeVinney and C.E. Priebe. A new family of proximity graphs: Class cover catch digraphs. *Discrete Applied Mathematics*, 154(14):1975–1982, 2006.

E.J. Wegman D.J. Marchette and C.E. Priebe. Fast algorithms for classification using class cover catch digraphs. *Handbook of Statistics*, 24:331–358, 2005.

S.N. Dorogovtsev and J.F.F. Mendes. *Evolution of Networks: From Biological Nets to the Internet and WWW*. Oxford University Press, 2003.

P.J. Grother, G.T. Candela, and J.L. Blue. Fast implementation of nearest neighbor classifiers. *Pattern Recognition*, 30:459–465, 1997.

P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14: 515–516, 1968.

N. Jankowski and M. Grochowski. Comparison of instances selection algorithms i. algorithms survey. In *Artificial Intelligence and Soft Computing*, pages 598–603. Springer, 2004a.

N. Jankowski and M. Grochowski. Comparison of instances selection algorithms ii. results and comments. In *Artificial Intelligence and Soft Computing*, pages 580–585. Springer, 2004b.

J.W. Jaromczyk and G.T. Toussaint. Relative neighborhood graphs and their relatives. *P-IEEE*, 80: 1502–1517, 1992.

C.D.J. Marchette, C.E. Priebe, D.A. Socolinsky, and J.G. DeVinney. Classification using class cover catch digraphs. *Journal of classification*, 20(1):3–23, 2003.

K. Mukherjee. Application of the gabriel graph to instance-based learning. In *M.sc. project, School of Computing Science, Simon Fraser University*, 2004.

E. Pekalska, R.P. W. Duin, and P. Paclík. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189–208, 2006.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3): 287–320, 2001.

G.L. Ritter, H.B. Woodruff, S.R. Lowry, and T.L. Isenhour. An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21(6):665–669, 1975.

J.S. Sánchez, F. Pla, and F.J. Ferri. Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognition Letters*, 18:507–513, 1997.

J. Sankaranarayanan, H. Samet, and A. Varshney. A fast all nearest neighbor algorithm for applications involving large point-clouds. *Comput. Graph.*, 31(2):157–174, 2007.

H. Shin and S. Cho. Neighborhood property based pattern selection for support vector machines. *Neural Computation*, (19):816–855, 2007.

I. Tomek. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(6):448–452, 1976.

G.T. Toussaint. Proximity graphs for nearest neighbor decision rules: recent progress. In *Interface-2002, 34th Symposium on Computing and Statistics*, pages 83–106, 2002.

G.T. Toussaint, B.K. Bhattacharya, and R.S. Poulsen. The application of voronoi diagrams to nonparametric decision rules. In *Proceedings of the 16th Symposium on Computer Science and Statistics*, pages 97 –108, 1984.

A. Vezhnevets and O. Barinova. Avoiding boosting overfitting by removing "confusing" samples. In *Proceedings of the 18th European Conference on Machine Learning (ECML)*, volume 4701, pages 430–441. LNCS, 2007.

D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, (2):408–420, 1972.

D.R. Wilson and T.R. Martinez. Instance pruning techniques. In *Proc. 14th International Conference on Machine Learning*, pages 403–411. Morgan Kaufmann, 1997.

D.R. Wilson and T.R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.

D.A. Zighed, S. Lallich, and F. Muhlenbach. Separability index in supervised learning. In *PKDD '02: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 475–487. Springer-Verlag, 2002.

Journal of Machine Learning Research 8 (2008) 1019-1048 Submitted 10/07; Revised 3/08; Published 6/08

Consistency of Trace Norm Minimization

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

Editor: Xiaotong Shen

Abstract

Regularization by the sum of singular values, also referred to as the *trace norm*, is a popular technique for estimating low rank rectangular matrices. In this paper, we extend some of the consistency results of the Lasso to provide necessary and sufficient conditions for rank consistency of trace norm minimization with the square loss. We also provide an adaptive version that is rank consistent even when the necessary condition for the non adaptive version is not fulfilled.

Keywords: convex optimization, singular value decomposition, trace norm, consistency

1. Introduction

In recent years, regularization by various non Euclidean norms has seen considerable interest. In particular, in the context of linear supervised learning, norms such as the ℓ_1-norm may induce sparse loading vectors, that is, loading vectors with low cardinality or ℓ_0-norm. Such regularization schemes, also known as the Lasso (Tibshirani, 1994) for least-square regression, come with efficient path following algorithms (Efron et al., 2004). Moreover, recent work has studied conditions under which such procedures consistently estimate the sparsity pattern of the loading vector (Yuan and Lin, 2007; Zhao and Yu, 2006; Zou, 2006).

When learning on rectangular matrices, the rank is a natural extension of the cardinality, and the sum of singular values, also known as the trace norm or the nuclear norm, is the natural extension of the ℓ_1-norm; indeed, as the ℓ_1-norm is the convex envelope of the ℓ_0-norm on the unit ball (i.e., the largest lower bounding convex function) (Boyd and Vandenberghe, 2003), the trace norm is the convex envelope of the rank over the unit ball of the spectral norm (Fazel et al., 2001). In practice, it leads to low rank solutions (Fazel et al., 2001; Srebro et al., 2005) and has seen recent increased interest in the context of collaborative filtering (Srebro et al., 2005), multi-task learning (Abernethy et al., 2006; Argyriou et al., 2007; Abernethy et al., 2008) or classification with multiple classes (Amit et al., 2007).

In this paper, we consider the rank consistency of trace norm regularization with the square loss, that is, if the data were actually generated by a low-rank matrix, will the matrix and its rank be consistently estimated? In Section 4, we provide necessary and sufficient conditions for the rank consistency that are extensions of corresponding results for the Lasso (Yuan and Lin, 2007; Zhao and Yu, 2006; Zou, 2006) and the group Lasso (Bach, 2008). We do so under two sets of sampling

assumptions detailed in Section 3.2: a full i.i.d assumption and a non i.i.d assumption which is natural in the context of collaborative filtering.

As for the Lasso and the group Lasso, the necessary condition implies that such procedures do not always estimate the rank correctly; similar to the adaptive version of the Lasso and group Lasso (Zou, 2006), we design an adaptive version to achieve $n^{-1/2}$-consistency and rank consistency, with no consistency conditions. Following Zou (2006), the adaptive version is based on a unregularized least-squares estimates which is used to design appropriate reweighted matrices. Finally, in Section 6, we present a smoothing approach to convex optimization with the trace norm, while in Section 6.3, we show simulations on toy examples to illustrate the consistency results.

2. Notation

In this paper we consider various norms on vectors and matrices. On vectors x in $\mathbb{R}^d$, we always consider the Euclidean norm, that is, $\|x\| = (x^\top x)^{1/2}$. On rectangular matrices in $\mathbb{R}^{p\times q}$, however, we consider several norms, based on singular values (Stewart and Sun, 1990): the *spectral norm* $\|M\|_2$ is the largest singular value (defined as $\|M\|_2 = \sup_{x\in\mathbb{R}^q} \frac{\|Mx\|}{\|x\|}$), the *trace norm* (or *nuclear norm*) $\|M\|_*$ is the sum of singular values, and the *Frobenius norm* $\|M\|_F$ is the ℓ_2-norm of singular values—also defined as $\|M\|_F = (\mathrm{tr} M^\top M)^{1/2}$. In Appendix A and B, we review and derive relevant tools and results regarding perturbation of singular values as well as the trace norm.

Given a matrix $M \in \mathbb{R}^{p\times q}$, $\mathrm{vec}(M)$ denotes the vector in $\mathbb{R}^{pq}$ obtained by stacking its columns into a single vector; and $A \otimes B$ denotes the Kronecker product between matrices $A \in \mathbb{R}^{p_1\times q_1}$ and $B \in \mathbb{R}^{p_2\times q_2}$, defined as the matrix in $\mathbb{R}^{p_1 p_2 \times q_1 q_2}$, defined by blocks of sizes $p_2 \times q_2$ equal to $a_{ij}B$. We make constant use of the following identities: $(B^\top \otimes A)\,\mathrm{vec}(X) = \mathrm{vec}(AXB)$ and $\mathrm{vec}(uv^\top) = v \otimes u$. For more details and properties, see Golub and Loan (1996) and Magnus and Neudecker (1998). We also use the notation ΣW for $\Sigma \in \mathbb{R}^{pq\times pq}$ and $W \in \mathbb{R}^{p\times q}$ to design the matrix in $\mathbb{R}^{p\times q}$ such that $\mathrm{vec}(\Sigma W) = \Sigma\,\mathrm{vec}(W)$ (note the potential confusion with ΣW when Σ is a matrix with p columns).

We use the following standard asymptotic notations: a random variable Z_n is said to be of order $O_p(a_n)$ if for any $\eta > 0$, there exists $M > 0$ such that $\sup_n P(|Z_n| > Ma_n) < \eta$. Moreover, Z_n is said to be of order $o_p(a_n)$ if Z_n/a_n converges to zero in probability, that is, if for any $\eta > 0$, $P(|Z_n| \geqslant \eta a_n)$ converges to zero. See Van der Vaart (1998) and Shao (2003) for further definitions and properties of asymptotics in probability.

Finally, we use the following two conventions: lowercase for vectors and uppercase for matrices, while bold fonts are reserved for population quantities.

3. Trace Norm Minimization

We consider the problem of predicting a real random variable z as a linear function of a matrix $M \in \mathbb{R}^{p\times q}$, where p and q are two fixed strictly positive integers. Throughout this paper, we assume that we are given n observations (M_i, z_i), $i = 1,\dots,n$, and we consider the following optimization problem with the square loss:

$$\min_{W\in\mathbb{R}^{p\times q}} \frac{1}{2n}\sum_{i=1}^n (z_i - \mathrm{tr} W^\top M_i)^2 + \lambda_n \|W\|_*, \tag{1}$$

where $\|W\|_*$ denotes the *trace norm* of W.

3.1 Special Cases

Regularization by the trace norm has numerous applications (see, e.g., Recht et al., 2007, for a review); in this paper, we are particularly interested in the following two situations:

Lasso and group Lasso When $x_i \in \mathbb{R}^m$, we can define $M_i = \mathrm{Diag}(x_i) \in \mathbb{R}^{m \times m}$ as the diagonal matrix with x_i on the diagonal. In this situation, the minimization of problem in Eq. (1) must lead to diagonal solutions (indeed the minimum trace norm matrix with fixed diagonal is the corresponding diagonal matrix, which is a consequence of Lemma 20 and Proposition 21) and for a diagonal matrix the trace norm is simply the ℓ_1 norm of the diagonal. Once we have derived our consistency conditions, we check in Section 4.5 that they actually lead to the known ones for the Lasso (Yuan and Lin, 2007; Zhao and Yu, 2006; Zou, 2006).

We can also see the group Lasso as a special case; indeed, if $x_{ij} \in \mathbb{R}^{d_j}$ for $j = 1, \ldots, m$, $i = 1, \ldots, n$, then we define $M_i \in \mathbb{R}^{(\sum_{j=1}^m d_j) \times m}$ as the block diagonal matrix (with non square blocks) with diagonal blocks x_{ji}, $j = 1, \ldots, m$. Similarly, the optimal $\hat{W}$ must share the same block-diagonal form, and its singular values are exactly the norms of each block, that is, the trace norm is indeed the sum of the norms of each group. We also get back results from Bach (2008) in Section 4.5.

Note that the Lasso and group Lasso can be seen as special cases where the singular vectors are fixed. However, the main difficulty in analyzing trace norm regularization, as well as the main reason for it use, is that singular vectors are not fixed and those can often be seen as implicit features learned by the estimation procedure (Srebro et al., 2005). In this paper we derive consistency results about the value and numbers of such features.

Collaborative filtering and low-rank completion Another natural application is collaborative filtering where two types of attributes x and y are observed and we consider bilinear forms in x and y, which can be written as a linear form in $M = xy^\top$ (thus it corresponds to situations where all matrices M_i have rank one). In this setting, the matrices M_i are not usually i.i.d. but exhibit a statistical dependence structure outlined in Section 3.2. A special case here is when then no attributes are observed and we simply wish to complete a partially observed matrix (Srebro et al., 2005; Abernethy et al., 2006). The results presented in this paper do not immediately apply because the dimension of the estimated matrix may grow with the number of observed entries and this situation is out of the scope of this paper.

Multivariate linear supervised learning When predicting multiple variables, in the context of multivariate linear regression (Yuan et al., 2007) or in the multiple category classification (Amit et al., 2007), the trace norm allows to perform feature selection.

3.2 Assumptions

We make the following assumptions on the sampling distributions of $M \in \mathbb{R}^{p \times q}$ for the problem in Eq. (1). We let denote: $\hat{\Sigma}_{mm} = \frac{1}{n} \sum_{i=1}^n \mathrm{vec}(M_i)\,\mathrm{vec}(M_i)^\top \in \mathbb{R}^{pq \times pq}$, and we consider the following assumptions:

(**A1**) Given M_i, $i = 1, \ldots, n$, the n values z_i are i.i.d. and there exists $\mathbf{W} \in \mathbb{R}^{p \times q}$ such that for all i, $\mathbb{E}(z_i | M_1, \ldots, M_n) = \mathrm{tr}\mathbf{W}^\top M_i$ and $\mathrm{var}(z_i | M_1, \ldots, M_n)$ is a strictly positive constant σ^2. $\mathbf{W}$ is not equal to zero and does not have full rank.

(**A2**) There exists an *invertible* matrix $\Sigma_{mm} \in \mathbb{R}^{pq \times pq}$ such that $\mathbb{E}\|\hat{\Sigma}_{mm} - \Sigma_{mm}\|_F^2 = O(\zeta_n^2)$ for a certain sequence ζ_n that tends to zero.

(**A3**) The random variable $n^{-1/2}\sum_{i=1}^n \varepsilon_i \operatorname{vec}(M_i)$ is converging in distribution to a normal distribution with mean zero and covariance matrix $\sigma^2\Sigma_{mm}$.

Assumption (**A1**) states that given the input matrices M_i, $i = 1,\dots,n$ we have a linear prediction model, where the loading matrix $\mathbf{W}$ is non trivial and rank-deficient, the goal being to estimate this rank (as well as the matrix itself). We let denote $\mathbf{W} = \mathbf{U}\operatorname{Diag}(\mathbf{s})\mathbf{V}^\top$ its singular value decomposition, with $\mathbf{U} \in \mathbb{R}^{p\times \mathbf{r}}$, $\mathbf{V} \in \mathbb{R}^{q\times \mathbf{r}}$, and $\mathbf{r} \in (0, \min\{p,q\})$ denotes the rank of $\mathbf{W}$. We also let denote $\mathbf{U}_\perp \in \mathbb{R}^{p\times(p-\mathbf{r})}$ and $\mathbf{V}_\perp \in \mathbb{R}^{q\times(q-\mathbf{r})}$ any orthogonal complements of $\mathbf{U}$ and $\mathbf{V}$.

We let denote $\varepsilon_i = z_i - \operatorname{tr}\mathbf{W}^\top M_i$ and $\hat{\Sigma}_{Mz} = \frac{1}{n}\sum_{i=1}^n z_i M_i \in \mathbb{R}^{p\times q}$, $\hat{\Sigma}_{M\varepsilon} = \frac{1}{n}\sum_{i=1}^n \varepsilon_i M_i = \hat{\Sigma}_{Mz} - \hat{\Sigma}_{mm}\mathbf{W} \in \mathbb{R}^{p\times q}$. We may then rewrite Eq. (1) as

$$\min_{W\in\mathbb{R}^{p\times q}} \frac{1}{2}\operatorname{vec}(W)^\top \hat{\Sigma}_{mm}\operatorname{vec}(W) - \operatorname{tr}W^\top \hat{\Sigma}_{Mz} + \lambda_n \|W\|_*,$$

or, equivalently,

$$\min_{W\in\mathbb{R}^{p\times q}} \frac{1}{2}\operatorname{vec}(W-\mathbf{W})^\top \hat{\Sigma}_{mm}\operatorname{vec}(W-\mathbf{W}) - \operatorname{tr}W^\top \hat{\Sigma}_{M\varepsilon} + \lambda_n \|W\|_*.$$

The sampling assumptions (**A2**) and (**A3**) may seem restrictive, but they are satisfied in the following two natural situations. The first situation corresponds to a classical full i.i.d problem, where the pairs (z_i, M_i) are sampled i.i.d:

Lemma 1 *Assume (**A1**). If the matrices M_i are sampled i.i.d., z and M have finite fourth order moments, and $\mathbb{E}\left\{\operatorname{vec}(M)\operatorname{vec}(M)^\top\right\}$ is invertible, then (**A2**) and (**A3**) are satisfied with $\zeta_n = n^{-1/2}$.*

Note the further refinement when for each i, $M_i = x_i y_i^\top$ and x_i and y_i are independent, which implies that Σ_{mm} is factorized as a Kronecker product, of the form $\Sigma_{yy} \otimes \Sigma_{xx}$ where Σ_{xx} and Σ_{yy} are the (invertible) second order moment matrices of x and y.

The second situation corresponds to a collaborative filtering situation where two types of attributes are observed, for example, x and y, and for every pair (x,y) we wish to predict z as a bilinear form in x and y: we first sample n_x values for x, and n_y values for y, and we select uniformly at random a subset of $n \leqslant n_x n_y$ observations from the $n_x n_y$ possible pairs. The following lemma, proved in Appendix C.1, shows that this set-up satisfies our assumptions:

Lemma 2 *Assume (**A1**). Assume moreover that n_x values $\tilde{x}_1,\dots,\tilde{x}_{n_x}$ are sampled i.i.d and n_y values $\tilde{y}_1,\dots,\tilde{y}_{n_y}$ are also sampled i.i.d. from distributions with finite fourth order moments and invertible second order moment matrices Σ_{xx} and Σ_{yy}. Assume also that a random subset of size n of pairs (i_k, j_k) in $\{1,\dots,n_x\}\times\{1,\dots,n_y\}$ is sampled uniformly, then if n_x, n_y and n tend to infinity, then (**A2**) and (**A3**) are satisfied with $\Sigma_{mm} = \Sigma_{yy}\otimes\Sigma_{xx}$ and $\zeta_n = n^{-1/2} + n_x^{-1/2} + n_y^{-1/2}$.*

3.3 Optimality Conditions

From the expression of the subdifferential of the trace norm in Proposition 21 (Appendix B), we can identify the optimality condition for problem in Eq. (1), that we will constantly use in the paper:

Proposition 3 *The matrix W with singular value decomposition $W = U\,\mathrm{Diag}(s)V^\top$ (with strictly positive singular values s) is optimal for the problem in Eq. (1) if and only if*

$$\hat{\Sigma}_{mm}W - \hat{\Sigma}_{Mz} + \lambda_n UV^\top + N = 0,$$

with $U^\top N = 0$, $NV = 0$ and $\|N\|_2 \leqslant \lambda_n$.

This implies notably that W and $\hat{\Sigma}_{mm}W - \hat{\Sigma}_{Mz}$ have *simultaneous* singular value decompositions, and the largest singular values are less than λ_n, and exactly equal to λ_n for the corresponding strictly positive singular values of W. Note that when all matrices are diagonal (the Lasso case), we obtain the usual optimality conditions (see also Recht et al., 2007, for further discussions).

4. Consistency Results

We consider two types of consistency; first, the *regular consistency*, that is, we want the probability $\mathbb{P}(\|\hat{W} - \mathbf{W}\| \geqslant \varepsilon)$ to tend to zero as n tends to infinity, for all $\varepsilon > 0$. We also consider the *rank consistency*, that is, we want that $\mathbb{P}(\mathrm{rank}(\hat{W}) \neq \mathrm{rank}(\mathbf{W}))$ tends to zero as n tends to infinity. Following the similar properties for the Lasso, the consistency depends on the decay of the regularization parameter. Essentially, we obtain the following results:

a) if λ_n does not tend to zero, then the trace norm estimate $\hat{W}$ is not consistent;

b) if λ_n tends to zero faster than $n^{-1/2}$, then the estimate is consistent and its error is $O_p(n^{-1/2})$ while it is not rank-consistent with probability tending to one (see Section 4.1);

c) if λ_n tends to zero exactly at rate $n^{-1/2}$, then the estimator is consistent with error $O_p(n^{-1/2})$ but the probability of estimating the correct rank is converging to a limit in $(0,1)$ (see Section 4.2);

d) if λ_n tends to zero more slowly than $n^{-1/2}$, then the estimate is consistent with error $O_p(\lambda_n)$ and its rank consistency depends on specific *consistency conditions* detailed in Section 4.3.

The following sections will look at each of these cases, and state precise theorems. We then consider some special cases, that is, factored second-order moments and implications for the special cases of the Lasso and group Lasso.

The first proposition (proved in Appendix C.2) considers the case where the regularization parameter λ_n is converging to a certain limit λ_0. When this limit is zero, we obtain regular consistency (Corollary 5 below), while if $\lambda_0 > 0$, then $\hat{W}$ tends in probability to a limit which is always different from $\mathbf{W}$:

Proposition 4 *Assume (**A1-3**). Let $\hat{W}$ be a global minimizer of Eq. (1). If λ_n tends to a limit $\lambda_0 \geqslant 0$, then $\hat{W}$ converges in probability to the unique global minimizer of*

$$\min_{W \in \mathbb{R}^{p\times q}} \frac{1}{2}\,\mathrm{vec}(W - \mathbf{W})^\top \Sigma_{mm}\,\mathrm{vec}(W - \mathbf{W}) + \lambda_0 \|W\|_*.$$

Corollary 5 *Assume (**A1-3**). Let $\hat{W}$ be a global minimizer of Eq. (1). If λ_n tends to zero, then $\hat{W}$ converges in probability to $\mathbf{W}$.*

We now consider finer results when λ_n tends to zero at certain rates, slower or faster than $n^{-1/2}$, or exactly at rate $n^{-1/2}$.

4.1 Fast Decay of Regularization Parameter

The following proposition—which is a consequence of standard results in M-estimation (Shao, 2003; Van der Vaart, 1998)—considers the case where $n^{1/2}\lambda_n$ is tending to zero, where we obtain that $\hat{W}$ is asymptotically normal with mean $\mathbf{W}$ and covariance matrix $n^{-1}\sigma^2\Sigma_{mm}^{-1}$, that is, for fast decays, the first order expansion is the same as the one with no regularization parameter:

Proposition 6 *Assume (**A1-3**). Let $\hat{W}$ be a global minimizer of Eq. (1). If $n^{1/2}\lambda_n$ tends to zero, $n^{1/2}(\hat{W}-\mathbf{W})$ is asymptotically normal with mean $\mathbf{W}$ and covariance matrix $\sigma^2\Sigma_{mm}^{-1}$.*

We now consider the corresponding rank consistency results, when λ_n goes to zero faster than $n^{-1/2}$. The following proposition (proved in Appendix C.3) states that for such regularization parameter, the solution has rank strictly greater than $\mathbf{r}$ with probability tending to one and can thus not be rank consistent:

Proposition 7 *Assume (**A1-3**). If $n^{1/2}\lambda_n$ tends to zero, then $\mathbb{P}(\mathrm{rank}(\hat{W}) > \mathrm{rank}(\mathbf{W}))$ tends to one.*

4.2 $n^{-1/2}$-decay of the Regularization Parameter

We first consider regular consistency through the following proposition (proved in Appendix C.4), then rank consistency (proposition proved in Appendix C.5):

Proposition 8 *Assume (**A1-3**). Let $\hat{W}$ be a global minimizer of Eq. (1). If $n^{1/2}\lambda_n$ tends to a limit $\lambda_0 > 0$, then $n^{1/2}(\hat{W}-\mathbf{W})$ converges in distribution to the unique global minimizer of*

$$\min_{\Delta\in\mathbb{R}^{p\times q}} \frac{1}{2}\mathrm{vec}(\Delta)^\top\Sigma_{mm}\mathrm{vec}(\Delta) - \mathrm{tr}\Delta^\top A + \lambda_0\left[\mathrm{tr}\mathbf{U}^\top\Delta\mathbf{V} + \|\mathbf{U}_\perp^\top\Delta\mathbf{V}_\perp\|_*\right],$$

where $\mathrm{vec}(A)\in\mathbb{R}^{pq}$ is normally distributed with mean zero and covariance matrix $\sigma^2\Sigma_{mm}$.

Proposition 9 *Assume (**A1-3**). If $n^{1/2}\lambda_n$ tends to a limit $\lambda_0 > 0$, then the probability that the rank of $\hat{W}$ is different from the rank of $\mathbf{W}$ is converging to $\mathbb{P}(\|\Lambda - \lambda_0^{-1}\Theta\|_2 \leqslant 1) \in (0,1)$ where $\Lambda \in \mathbb{R}^{(p-\mathbf{r})\times(q-\mathbf{r})}$ is defined in Eq. (3) (Section 4.3) and $\Theta\in\mathbb{R}^{(p-\mathbf{r})\times(q-\mathbf{r})}$ has a normal distribution with mean zero and covariance matrix*

$$\sigma^2\left((\mathbf{V}_\perp\otimes\mathbf{U}_\perp)^\top\Sigma_{mm}^{-1}(\mathbf{V}_\perp\otimes\mathbf{U}_\perp)\right)^{-1}.$$

The previous proposition ensures that the estimate $\hat{W}$ cannot be rank consistent with this decay of the regularization parameter. Note that when we take λ_0 small (i.e., we get closer to fast decays), the probability $\mathbb{P}(\|\Lambda-\lambda_0^{-1}\Theta\|_2\leqslant 1)$ tends to zero, while when we take λ_0 large (i.e., we get closer to slow decays), the same probability tends to zero or one depending on the sign of $\|\Lambda\|_2 - 1$. This heuristic argument is made more precise in the following section.

4.3 Slow Decay of Regularization Parameter

When λ_n tends to zero more slowly than $n^{-1/2}$, the first order expansion is deterministic, as the following proposition shows (proof in Appendix C.6):

Proposition 10 *Assume (**A1-3**). Let $\hat{W}$ be a global minimizer of Eq. (1). If $n^{1/2}\lambda_n$ tends to $+\infty$ and λ_n tends to zero, then $\lambda_n^{-1}(\hat{W}-\mathbf{W})$ converges in probability to the unique global minimizer Δ of*

$$\min_{\Delta\in\mathbb{R}^{p\times q}} \frac{1}{2}\operatorname{vec}(\Delta)^\top \Sigma_{mm}\operatorname{vec}(\Delta) + \operatorname{tr}\mathbf{U}^\top\Delta\mathbf{V} + \|\mathbf{U}_\perp^\top\Delta\mathbf{V}_\perp\|_*. \tag{2}$$

Moreover, we have $\hat{W} = \mathbf{W} + \lambda_n\Delta + O_p(\lambda_n + \zeta_n + \lambda_n^{-1}n^{-1/2})$.

The last proposition gives a first order expansion of $\hat{W}$ around $\mathbf{W}$. From Proposition 18 (Appendix B), we obtain immediately that if $\mathbf{U}_\perp^\top\Delta\mathbf{V}_\perp$ is different from zero, then the rank of $\hat{W}$ is ultimately strictly larger than $\mathbf{r}$. The condition $\mathbf{U}_\perp^\top\Delta\mathbf{V}_\perp = 0$ is thus necessary for rank consistency when $\lambda_n n^{1/2}$ tends to infinity while λ_n tends to zero. The next lemma (proved in Appendix 11), gives a necessary and sufficient condition for $\mathbf{U}_\perp^\top\Delta\mathbf{V}_\perp = 0$.

Lemma 11 *Assume Σ_{mm} is invertible, and $\mathbf{W} = \mathbf{U}\operatorname{Diag}(\mathbf{s})\mathbf{V}^\top$ is the singular value decomposition of $\mathbf{W}$. Then the unique global minimizer of*

$$\operatorname{vec}(\Delta)^\top \Sigma_{mm}\operatorname{vec}(\Delta) + \operatorname{tr}\mathbf{U}^\top\Delta\mathbf{V} + \|\mathbf{U}_\perp^\top\Delta\mathbf{V}_\perp\|_*$$

satisfies $\mathbf{U}_\perp^\top\Delta\mathbf{V}_\perp = 0$ if and only if

$$\left\| \left((\mathbf{V}_\perp\otimes\mathbf{U}_\perp)^\top\Sigma_{mm}^{-1}(\mathbf{V}_\perp\otimes\mathbf{U}_\perp)\right)^{-1}\left((\mathbf{V}_\perp\otimes\mathbf{U}_\perp)^\top\Sigma_{mm}^{-1}(\mathbf{V}\otimes\mathbf{U})\operatorname{vec}(\mathbf{I})\right)\right\|_2 \leqslant 1.$$

This leads to consider the matrix $\Lambda\in\mathbb{R}^{(p-\mathbf{r})\times(q-\mathbf{r})}$ defined as

$$\operatorname{vec}(\Lambda) = \left((\mathbf{V}_\perp\otimes\mathbf{U}_\perp)^\top\Sigma_{mm}^{-1}(\mathbf{V}_\perp\otimes\mathbf{U}_\perp)\right)^{-1}\left((\mathbf{V}_\perp\otimes\mathbf{U}_\perp)^\top\Sigma_{mm}^{-1}(\mathbf{V}\otimes\mathbf{U})\operatorname{vec}(\mathbf{I})\right), \tag{3}$$

and the two weak and strict *consistency conditions*:

$$\|\Lambda\|_2 \leqslant 1, \tag{4}$$

$$\|\Lambda\|_2 < 1. \tag{5}$$

Note that if Σ_{mm} is proportional to identity, they are always satisfied because then $\Lambda = 0$. We can now prove that the condition in Eq. (5) is sufficient for rank consistency when $n^{1/2}\lambda_n$ tends to infinity, while the condition Eq. (4) is necessary for the existence of a sequence λ_n such that the estimate is both consistent and rank consistent (which is a stronger result than restricting λ_n to be tending to zero slower than $n^{-1/2}$). The following two theorems are proved in Appendix C.8 and C.9:

Theorem 12 *Assume (**A1-3**). Let $\hat{W}$ be a global minimizer of Eq. (1). If the condition in Eq. (5) is satisfied, and if $n^{1/2}\lambda_n$ tends to $+\infty$ and λ_n tends to zero, then the estimate $\hat{W}$ is consistent and rank-consistent.*

Theorem 13 *Assume (**A1-3**). Let $\hat{W}$ be a global minimizer of Eq. (1). If the estimate $\hat{W}$ is consistent and rank-consistent, then the condition in Eq. (4) is satisfied.*

As opposed to the Lasso, where Eq. (4) is a necessary and sufficient condition for rank consistency (Yuan and Lin, 2007), this is not even true in general for the group Lasso (Bach, 2008). Looking at the limiting case $\|\Lambda\|_2 = 1$ would similarly lead to additional but more complex sufficient and necessary conditions, and is left out for future research.

Moreover, it may seem surprising that even when the sufficient condition Eq. (5) is fulfilled, that the first order expansion of $\hat{W}$, that is, $\hat{W} = \mathbf{W} + \lambda_n \Delta + o_p(\lambda_n)$ is such that $\mathbf{U}_\perp^\top \Delta \mathbf{V}_\perp = 0$, but nothing is said about $\mathbf{U}_\perp^\top \Delta \mathbf{V}$ and $\mathbf{U}^\top \Delta \mathbf{V}_\perp$, which are not equal to zero in general. This is due to the fact that the first $\mathbf{r}$ singular vectors U and V of $\mathbf{W} + \lambda_n \Delta$ are not fixed; indeed, the $\mathbf{r}$ first singular vectors (i.e., the implicit features) do rotate but with no contribution on $\mathbf{U}_\perp \mathbf{V}_\perp^\top$. This is to be contrasted with the adaptive version where asymptotically the first order expansion has constant singular vectors (see Section 5).

Finally, in this paper, we have only proved whether the probability of correct rank selection tends to zero or one. Proposition 9 suggests that when $\lambda_n n^{1/2}$ tends to infinity slowly, then this probability is close to $\mathbb{P}(\|\Lambda - \lambda_n^{-1} n^{1/2} \Theta\|_2 \leqslant 1)$, where Θ has a normal distribution with known covariance matrix, which converges to one exponentially fast when $\|\Lambda\|_2 < 1$. We are currently investigating additional assumptions under which such results are true and thus estimate the convergence rates of the probability of good rank selection as done by Zhao and Yu (2006) for the Lasso.

4.4 Factored Second Order Moment

Note that in the situation where n_x points in $\mathbb{R}^p$ and n_y points in $\mathbb{R}^q$ are sampled i.i.d and a random subset of n points in selected, then, we can refine the condition as follows (because $\Sigma_{mm} = \Sigma_{yy} \otimes \Sigma_{xx}$):

$$\Lambda = (\mathbf{U}_\perp^\top \Sigma_{xx}^{-1} \mathbf{U}_\perp)^{-1} \mathbf{U}_\perp^\top \Sigma_{xx}^{-1} \mathbf{U} \mathbf{V}^\top \Sigma_{yy}^{-1} \mathbf{V}_\perp (\mathbf{V}_\perp^\top \Sigma_{yy}^{-1} \mathbf{V}_\perp)^{-1},$$

which is equal to (by the expression of inverses of partitioned matrices):

$$\Lambda = (\mathbf{U}_\perp^\top \Sigma_{xx} \mathbf{U})(\mathbf{U}^\top \Sigma_{xx} \mathbf{U})^{-1} (\mathbf{V}^\top \Sigma_{yy} \mathbf{V})^{-1} (\mathbf{V}^\top \Sigma_{yy} \mathbf{V}_\perp).$$

This also happens when $M_i = x_i y_i^\top$ and x_i and y_i independent for all i.

4.5 Corollaries for the Lasso and Group Lasso

For the Lasso or the group Lasso, all proposed results in Section 4.3 should hold with the additional conditions that W and Δ are diagonal (block-diagonal for the group Lasso). In this situation, the singular values of the diagonal matrix $W = \mathrm{Diag}(w)$ are the norms of the diagonal blocks, while the left singular vectors are equal to the normalized versions of the block (the signs for the Lasso). However, the results developed in Section 4.3 do not immediately apply since the assumptions regarding the invertibility of the second order moment matrix is not satisfied. For those problems, all matrices M that are ever considered belong to a strict subspace of $\mathbb{R}^{p \times q}$ and we need to satisfy invertibility on that subspace.

More precisely, we assume that all matrices M are such that $\mathrm{vec}(M) = Hx$ where H is a given *design matrix* in $\mathbb{R}^{pq \times s}$ where s is the number of implicit parameter and $x \in \mathbb{R}^s$. If we replace the invertibility of Σ_{mm} by the invertibility of $H^\top \Sigma_{mm} H$, then all results presented in Section 4.3 are

valid, in particular, the matrix Λ may be written as

$$\mathrm{vec}(\Lambda) = \left((\mathbf{V}_\perp \otimes \mathbf{U}_\perp)^\top H (H^\top \Sigma_{mm} H)^{-1} H^\top (\mathbf{V}_\perp \otimes \mathbf{U}_\perp)\right)^\dagger \times \left((\mathbf{V}_\perp \otimes \mathbf{U}_\perp)^\top H (H^\top \Sigma_{mm} H)^{-1} H^\top (\mathbf{V} \otimes \mathbf{U}) \mathrm{vec}(\mathbf{I})\right), \quad (6)$$

where $A^\dagger$ denotes the pseudo-inverse of A (Golub and Loan, 1996).

We now apply Eq. (6) to the case of the group Lasso (which includes the Lasso as a special case). In this situation, we have $M = \mathrm{Diag}(x_1, \ldots, x_m)$ and each $x_j \in \mathbb{R}^{d_j}$, $j = 1, \ldots, m$; we consider w as being defined by blocks $w_1, \ldots, w_m$, where each $w_j \in \mathbb{R}^{d_j}$. The design matrix H is such that $Hw = \mathrm{vec}(\mathrm{Diag}(w))$ and the matrix $H^\top \Sigma_{mm} H$ is exactly equal to the joint covariance matrix Σ_{xx} of $x = (x_1, \ldots, x_m)$. Without loss of generality, we assume that the generating sparsity pattern corresponds to the first $\mathbf{r}$ blocks. We can then compute the singular value decomposition in closed form as $\mathbf{U} = \binom{(\mathrm{Diag}(\mathbf{w}_i / \|\mathbf{w}_i\|)_{i \leqslant \mathbf{r}})}{0}$, $\mathbf{V} = \binom{\mathbf{I}}{0}$ and $\mathbf{s} = (\|\mathbf{w}_j\|)_{j \leqslant \mathbf{r}}$. If we let denote, for each j, $\mathbf{O}_j$ a basis of the subspace orthogonal to $\mathbf{w}_j$, we have: $\mathbf{U}_\perp = \begin{pmatrix} \mathrm{Diag}(\mathbf{O}_i)_{i \leqslant \mathbf{r}} & 0 \\ 0 & \mathbf{I} \end{pmatrix}$ and $\mathbf{V}_\perp = \binom{0}{\mathbf{I}}$. We can put these singular vectors into Eq. (6) and get $(H^\top \Sigma_{mm} H)^{-1} H^\top (\mathbf{V} \otimes \mathbf{U}) \mathrm{vec}(\mathbf{I}) = (\Sigma_{xx}^{-1})_{\mathbf{J} \cup \mathbf{J}^c, \mathbf{J}} \eta_{\mathbf{J}}$, where $\mathbf{J} = \{1, \ldots, \mathbf{r}\}$ and $\eta_{\mathbf{J}}$ is the vector of normalized $\mathbf{w}_j$, $j \in \mathbf{J}$. Thus, for the group Lasso, we finally obtain:

$$\begin{aligned} \|\Lambda\|_2 &= \left\|\mathrm{Diag}\left[((\Sigma_{xx}^{-1})_{\mathbf{J}^c \mathbf{J}^c})^{-1} (\Sigma_{xx}^{-1})_{\mathbf{J}^c, \mathbf{J}} \eta_{\mathbf{J}}\right]\right\|_2 \\ &= \left\|\mathrm{Diag}\left[(\Sigma_{xx})_{\mathbf{J}^c \mathbf{J}} (\Sigma_{xx})_{\mathbf{J},\mathbf{J}}^{-1} \eta_{\mathbf{J}}\right]\right\|_2 \text{ by the partitioned matrices inversion lemma,} \\ &= \max_{i \in \mathbf{J}^c} \left\|\Sigma_{x_i x_{\mathbf{J}}} \Sigma_{x_{\mathbf{J}} x_{\mathbf{J}}}^{-1} \eta_{\mathbf{J}}\right\|. \end{aligned}$$

The condition on the invertibility of $H^\top \Sigma_{mm} H$ is exactly the invertibility of the full joint covariance matrix of $x = (x_1, \ldots, x_m)$ and is a standard assumption for the Lasso or the group Lasso (Yuan and Lin, 2007; Zhao and Yu, 2006; Zou, 2006; Bach, 2008). Moreover the condition $\|\Lambda\|_2 \leqslant 1$ is exactly the one for the group Lasso (Bach, 2008), where the pattern consistency is replaced by the consistency for the number of non zero groups.

Note that we only obtain a result in terms of numbers of selected groups of variables and not in terms of the identities of the groups themselves. However, because of regular consistency, we know that at least the $\mathbf{r}$ true groups will be selected, and then correct model size is asymptotically equivalent to the correct groups being selected.

5. Adaptive Version

We can follow the adaptive version of the Lasso to provide a consistent algorithm with no consistency conditions such as Eq. (4) or Eq. (5). More precisely, we consider the least-square estimate $\mathrm{vec}(\hat{W}_{LS}) = \hat{\Sigma}_{mm}^{-1} \mathrm{vec}(\hat{\Sigma}_{Mz})$. We have the following well known result for least-square regression:

Lemma 14 *Assume (**A1-3**). Then $n^{1/2}(\hat{\Sigma}_{mm}^{-1} \mathrm{vec}(\hat{\Sigma}_{Mz}) - \mathrm{vec}(\mathbf{W}))$ is converging in distribution to a normal distribution with zero mean and covariance matrix $\sigma^2 \Sigma_{mm}^{-1}$.*

We consider the singular value decomposition of $\hat{W}_{LS} = U_{LS} \mathrm{Diag}(s_{LS}) V_{LS}^\top$, where $s_{LS} \geqslant 0$. With probability tending to one, $\min\{p, q\}$ singular values are strictly positive (i.e., the rank of $\hat{W}_{LS}$ is

full). We consider the *full* decomposition where U_{LS} and V_{LS} are orthogonal *square* matrices and the matrix $\mathrm{Diag}(s_{LS})$ is rectangular. We complete the singular values $s_{LS} \in \mathbb{R}^{\min\{p,q\}}$ by $n^{-1/2}$ to reach dimensions p and q (we keep the same notation for both dimensions for simplicity).

For $\gamma \in (0,1]$, we let denote

$$A = U_{LS}\,\mathrm{Diag}(s_{LS})^{-\gamma} U_{LS}^\top \in \mathbb{R}^{p\times p} \text{ and } B = V_{LS}\,\mathrm{Diag}(s_{LS})^{-\gamma} V_{LS}^\top \in \mathbb{R}^{q\times q},$$

two positive definite symmetric matrices, and, following the adaptive Lasso of Zou (2006), we consider replacing $\|W\|_*$ by $\|AWB\|_*$—note that in the Lasso special case, this exactly corresponds to the adaptive Lasso of Zou (2006). We obtain the following consistency theorem (proved in Appendix C.10):

Theorem 15 *Assume (**A1-3**). If $\gamma \in (0,1]$, $n^{1/2}\lambda_n$ tends to 0 and $\lambda_n n^{1/2+\gamma/2}$ tends to infinity, then any global minimizer $\hat{W}_A$ of*

$$\frac{1}{2n}\sum_{i=1}^n (z_i - \mathrm{tr} W^\top M_i)^2 + \lambda_n \|AWB\|_*$$

is consistent and rank consistent. Moreover, $n^{1/2}\,\mathrm{vec}(\hat{W}_A - \mathbf{W})$ is converging in distribution to a normal distribution with mean zero and covariance matrix

$$\sigma^2 (\mathbf{V}\otimes\mathbf{U}) \Big[(\mathbf{V}\otimes\mathbf{U})^\top \Sigma_{mm} (\mathbf{V}\otimes\mathbf{U}) \Big]^{-1} (\mathbf{V}\otimes\mathbf{U})^\top.$$

Note the restriction $\gamma \leqslant 1$ which is due to the fact that the least-square estimate $\hat{W}_{LS}$ only estimates the singular subspaces at rate $O_p(n^{-1/2})$. In Section 6.3, we illustrate the previous theorem on synthetic examples. In particular, we exhibit some singular behavior for the limiting case $\gamma = 1$.

6. Algorithms and Simulations

In this section we provide a simple algorithm to solve problems of the form

$$\min_{W\in\mathbb{R}^{p\times q}} \frac{1}{2}\mathrm{vec}(W)^\top \Sigma\, \mathrm{vec}(W) - \mathrm{tr} W^\top Q + \lambda \|W\|_*, \tag{7}$$

where $\Sigma \in \mathbb{R}^{pq\times pq}$ is a positive definite matrix (note that we do not restrict Σ to be of the form $\Sigma = A\otimes B$ where A and B are positive semidefinite matrices of size $p\times p$ and $q\times q$). We assume that $\mathrm{vec}(Q)$ is in the column space of Σ, so that the optimization problem is bounded from below (and thus the dual is feasible). In our setting, we have $\Sigma = \hat{\Sigma}_{mm}$ and $Q = \hat{\Sigma}_{Mz}$.

We focus on problems where p and q are not too large so that we can apply Newton's method to obtain convergence up to machine precision, which is required for the fine analysis of rank consistency in Section 6.3. For more efficient algorithms with larger p and q, see Srebro et al. (2005); Rennie and Srebro (2005) and Abernethy et al. (2006); Lu et al. (2008).

Because the dual norm of the trace norm is the spectral norm (see Appendix B), the dual is easily obtained as

$$\max_{V\in\mathbb{R}^{p\times q}, \|V\|_2\leqslant 1} -\frac{1}{2}\mathrm{vec}(Q-\lambda V)^\top \Sigma^{-1}\,\mathrm{vec}(Q-\lambda V). \tag{8}$$

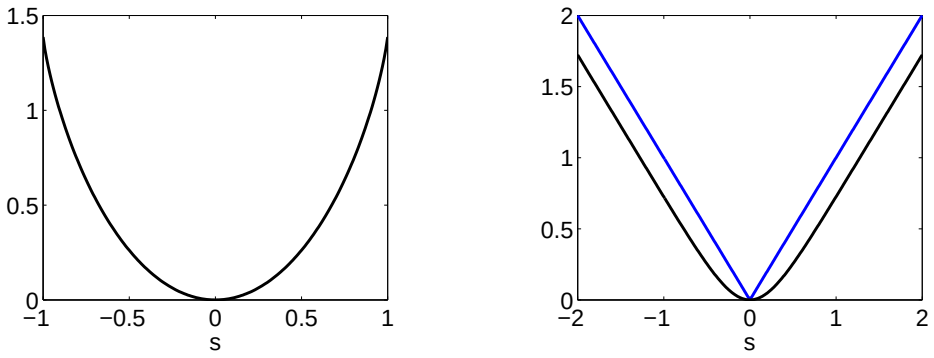


Figure 1: Spectral barrier functions: (left) primal function $b(s)$ and (right) dual functions $b^*(s)$.

Indeed, we have:

$$\begin{aligned}
& \min_{W\in\mathbb{R}^{p\times q}} \frac{1}{2}\mathrm{vec}(W)^\top \Sigma \mathrm{vec}(W) - \mathrm{tr}W^\top Q + \lambda\|W\|_* \\
= & \min_{W\in\mathbb{R}^{p\times q}} \max_{V\in\mathbb{R}^{p\times q},\|V\|_2\leqslant 1} \frac{1}{2}\mathrm{vec}(W)^\top \Sigma \mathrm{vec}(W) - \mathrm{tr}W^\top Q + \lambda \mathrm{tr}V^\top W \\
= & \max_{V\in\mathbb{R}^{p\times q},\|V\|_2\leqslant 1} \min_{W\in\mathbb{R}^{p\times q}} \frac{1}{2}\mathrm{vec}(W)^\top \Sigma \mathrm{vec}(W) - \mathrm{tr}W^\top Q + \lambda \mathrm{tr}V^\top W \\
= & \max_{V\in\mathbb{R}^{p\times q},\|V\|_2\leqslant 1} -\frac{1}{2}\mathrm{vec}(Q-\lambda V)^\top \Sigma^{-1}\mathrm{vec}(Q-\lambda V),
\end{aligned}$$

where strong duality holds because both the primal and dual problems are convex and strictly feasible (Boyd and Vandenberghe, 2003).

6.1 Smoothing

The problem in Eq. (7) is convex but non differentiable; in this paper we consider adding a strictly convex function to its dual in Eq. (8) in order to make it differentiable, while controlling the increase of duality gap yielded by the added function (Bonnans et al., 2003).

We thus consider the following smoothing of the trace norm, namely we define

$$F_\varepsilon(W) = \max_{V\in\mathbb{R}^{p\times q},\|V\|_2\leqslant 1} \mathrm{tr}V^\top W - \varepsilon B(V),$$

where $B(V)$ is a spectral function (i.e., that depends only on singular values of V, equal to $B(V) = \sum_{i=1}^{\min\{p,q\}} b(s_i(V))$ where $b(s) = (1+s)\log(1+s) + (1-s)\log(1-s)$ if $|s| \leqslant 1$ and $+\infty$ otherwise ($s_i(V)$ denotes the i-th largest singular values of V). This function F_ε may be computed in closed form as:

$$F_\varepsilon(W) = \sum_{i=1}^{\min\{p,q\}} b^*(s_i(W)),$$

where $b^*(s) = \varepsilon\log(1+e^{s/\varepsilon}) + \varepsilon\log(1+e^{-s/\varepsilon}) - 2\varepsilon\log 2$. These functions are plotted in Figure 1; note that $|b^*(s) - |s||$ is uniformly bounded by $2\log 2$.

We finally get the following pairs of primal/dual optimization problems:

$$\min_{W\in\mathbb{R}^{p\times q}} \frac{1}{2}\operatorname{vec}(W)^\top \Sigma \operatorname{vec}(W) - \operatorname{tr} W^\top Q + \lambda F_{\varepsilon/\lambda}(W),$$

$$\max_{V\in\mathbb{R}^{p\times q}, \|V\|_2\leqslant 1} -\frac{1}{2}\operatorname{vec}(Q-\lambda V)^\top \Sigma^{-1}\operatorname{vec}(Q-\lambda V) - \varepsilon B(V).$$

We can now optimize directly in the primal formulation which is infinitely differentiable, using Newton's method. Note that the stopping criterion should be an $\varepsilon \times \min\{p,q\}$ duality gap, as the controlled smoothing also leads to a small additional gap on the solution of the original non smoothed problem. More precisely, a duality gap of $\varepsilon \times \min\{p,q\}$ on the smoothed problem, leads to a gap of at most $(1+2\log 2)\varepsilon \times \min\{p,q\}$ for the original problem.

6.2 Implementation Details

In this section, we provide details about the implementation of the estimation algorithm presented earlier.

Derivatives of spectral functions Note that derivatives of spectral functions of the form $B(W) = \sum_{i=1}^{\min\{p,q\}} b(s_i(W))$, where b is an even twice differentiable function such that $b(0) = b'(0) = 0$, are easily calculated as follows; Let $U\operatorname{Diag}(s)V^\top$ be the singular value decomposition of W. We then have the following Taylor expansion (Lewis and Sendov, 2002):

$$B(W+\Delta) = B(W) + \operatorname{tr}\Delta^\top U \operatorname{Diag}(b'(s_i))V^\top + \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{q}\frac{b'(s_i)-b'(s_j)}{s_i-s_j}(u_i^\top \Delta v_j)^2,$$

where the vector of singular values is completed by zeros, and $\frac{b'(s_i)-b'(s_j)}{s_i-s_j}$ is defined as $b''(s_i)$ when $s_i = s_j$.

Choice of ε and computational complexity Following the common practice in barrier methods we decrease the parameter geometrically after each iteration of Newton's method (Boyd and Vandenberghe, 2003). Each of these Newton iterations has complexity $O(p^3q^3)$. Empirically, the number of iterations does not exceed a few hundreds for solving one problem up to machine precision. We are currently investigating theoretical bounds on the number of iterations through self concordance theory (Boyd and Vandenberghe, 2003).

Start and end of the path In order to avoid to consider useless values of the regularization parameter and thus use a well adapted grid for trying several λ's, we can consider a specific interval for λ. When λ is large, the solution is exactly zero, while when λ is small, the solution tends to $\operatorname{vec}(W) = \Sigma^{-1}\operatorname{vec}(Q)$.

More precisely, if λ is larger than $\|Q\|_2$, then the solution is exactly zero (because in this situation 0 is in the subdifferential). On the other side, we consider for which λ, $\Sigma^{-1}\operatorname{vec}(Q)$ leads to a duality gap which is less than $\varepsilon\operatorname{vec}(Q)^\top\Sigma^{-1}\operatorname{vec}(Q)$, where ε is small. A looser condition is to take $V=0$, and the condition becomes $\lambda\|\Sigma^{-1}\operatorname{vec}(Q)\|_* \leqslant \varepsilon\operatorname{vec}(Q)^\top\Sigma^{-1}\operatorname{vec}(Q)$. Note that this is in the correct order (i.e., lower bound smaller than upper bound), because

$$\operatorname{vec}(Q)^\top\Sigma^{-1}\operatorname{vec}(Q) = \langle \operatorname{vec}(Q), \Sigma^{-1}\operatorname{vec}(Q)\rangle \leqslant \|\Sigma^{-1}\operatorname{vec}(Q)\|_*\|\operatorname{vec}(Q)\|_2.$$

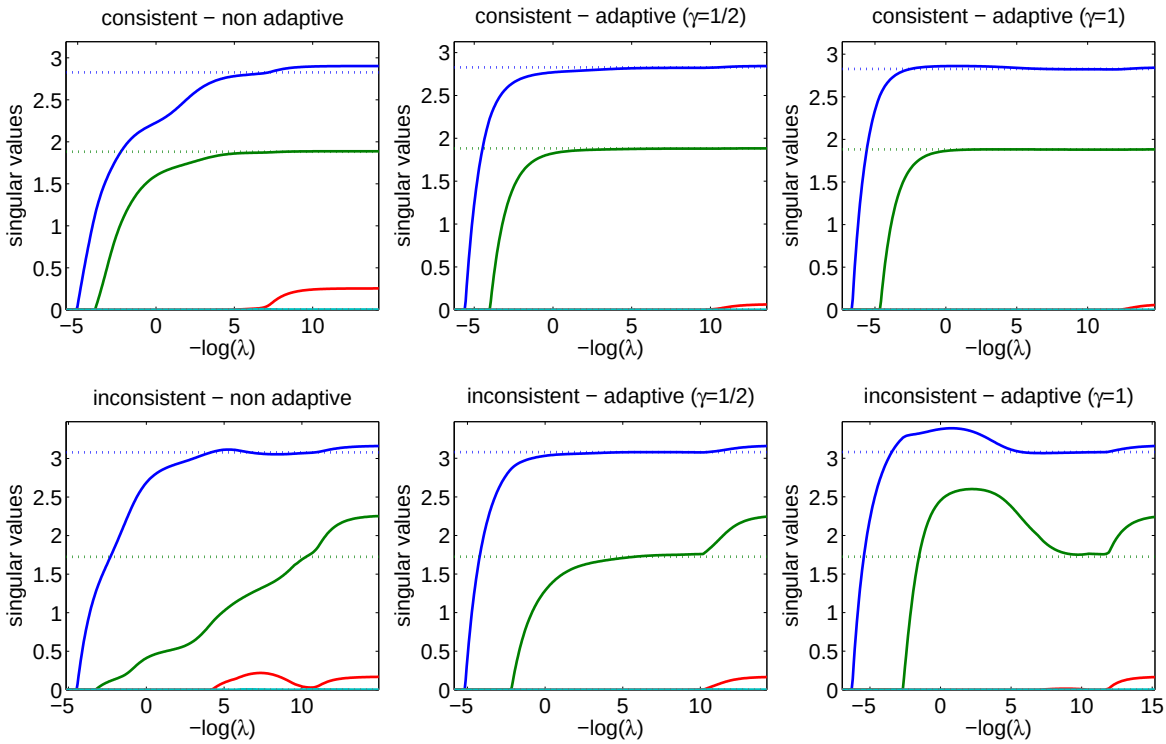


Figure 2: Examples of paths of singular values for $\|\Lambda\|_2 = 0.49 < 1$ (consistent, top) and $\|\Lambda\|_2 = 4.78 > 1$ (inconsistent, bottom) rank selection: regular trace norm penalization (left) and adaptive penalization with $\gamma = 1/2$ (center) and $\gamma = 1$ (right). Estimated singular values are plotted in plain, while population singular values are dotted.

This allows to design a good interval for searching for a good value of λ or for computing the regularization path by uniform grid sampling (in log scale), or for numerical path following with predictor-corrector methods such as used by Bach et al. (2004).

6.3 Simulations

In this section, we perform simulations on toy examples to illustrate our consistency results. We generate random i.i.d. data $\tilde{X}$ and $\tilde{Y}$ with Gaussian distributions and we select a low rank matrix $\mathbf{W}$ at random and generate $Z = \mathrm{diag}(\tilde{X}^\top \mathbf{W} \tilde{Y}) + \varepsilon$ where ε has i.i.d components with normal distributions with zero mean and known variance. In this section, we always use $\mathbf{r} = 2$, $p = q = 4$, while we consider several numbers of samples n, and several distributions for which the consistency conditions Eq. (4) and Eq. (5) may or may not be satisfied.[1]

In Figure 2, we plot regularization paths for $n = 10^3$, by showing the singular values of $\hat{W}$ compared to the singular values of $\mathbf{W}$, in two particular situations (Eq. (4) and Eq. (5) satisfied and not satisfied), for the regular trace norm regularization and the adaptive versions, with $\gamma = 1/2$ and $\gamma = 1$. Note that in the consistent case (top), the singular values and their cardinalities are well jointly estimated, both for the non adaptive version (as predicted by Theorem 12) and the adaptive

1. Simulations may be reproduced with MATLAB code available from `http://www.di.ens.fr/~fbach/tracenorm/`.

versions (Theorem 15), while the range of correct rank selection increases compared to the adaptive versions. However in the inconsistent case, the non adaptive regularizations scheme (bottom left) cannot achieve regular consistency together with rank consistency (Theorem 13), while the adaptive schemes can. Note the particular behavior of the limiting case $\gamma = 1$, which still achieves both consistencies but with a singular behavior for large λ.

In Figure 3, we select the distribution used for the rank-consistent case of Figure 2, and compute the paths from 200 replications for $n = 10^2$, 10^3, 10^3 and 10^5. For each λ, we plot the proportion of estimates with correct rank on the left plots (i.e., we get an estimation of $\mathbb{P}(\operatorname{rank}(\hat{W}) = \operatorname{rank}(\mathbf{W}))$, while we plot the logarithm of the average root mean squared estimation error $\|\hat{W} - \mathbf{W}\|$ on the right plot. For the three regularization schemes, the range of values with high probability of correct rank selection increases as n increases, and, most importantly achieves good mean squared error (right plot); in particular, for the non adaptive schemes (top plots), this corroborates the results from Proposition 9, which states that for $\lambda_n = \lambda_0 n^{-1/2}$ the probability tends to a limit in $(0,1)$: indeed, when n increases, the value λ_n which achieves a particular limit grows as $n^{-1/2}$, and considering the log-scale for λ_n in Figure 3 and the uniform sampling for n in log-scale as well, the regular spacing between the decaying parts observed in Figure 3 is coherent with our results.

In Figure 4, we perform the same operations but with the inconsistent case of Figure 2. For the non adaptive case (top plot), the range of values of λ that achieve high probability of correct rank selection does not increase when n increases and stays bounded, in places where the estimation error is not tending to zero: in the inconsistent case, the trace norm regularization does not manage to solve the trade-off between rank consistency and regular consistency. However, for the adaptive versions, it does, still with a somewhat singular behavior of the limiting case $\gamma = 1$.

Finally, in Figure 5, we consider 400 different distributions with various values of $\|\Lambda\|_2$ smaller or greater than one, and computed the regularization paths with $n = 10^3$ samples. From the paths, we consider the estimate $\hat{W}$ with correct rank and best distance to $\mathbf{W}$ and plot the best error versus $\log_{10}(\|\Lambda\|_2)$. For positive values of $\log_{10}(\|\Lambda\|_2)$, the best error is far from zero, and the error grows with the distance to zero; while for negative values, we get low errors with lower errors for small $\log_{10}(\|\Lambda\|_2)$, corroborating the influence of $\|\Lambda\|_2$ described in Proposition 9.

7. Conclusion

We have presented an analysis of the rank consistency for the penalization by the trace norm, and derived general necessary and sufficient conditions. This work can be extended in several interesting ways: first, by going from the square loss to more general losses, in particular for other types of supervised learning problems such as classification; or by looking at the collaborative filtering setting where only some of the attributes are observed (Abernethy et al., 2006) and dimensions p and q are allowed to grow. Moreover, we are currently pursuing non asymptotic extensions of the current work, making links with the recent work of Recht et al. (2007) and of Meinshausen and Yu (2006).

Appendix A. Tools for Analysis of Singular Value Decomposition

In this appendix, we review and derive precise results regarding singular value decompositions. We consider $W \in \mathbb{R}^{p \times q}$ and we let denote $W = U \operatorname{Diag}(s) V^\top$ its singular value decomposition with $U \in \mathbb{R}^{p \times r}$, $V \in \mathbb{R}^{q \times r}$ with orthonormal columns, and $s \in \mathbb{R}^r$ with strictly positive values (r is the rank

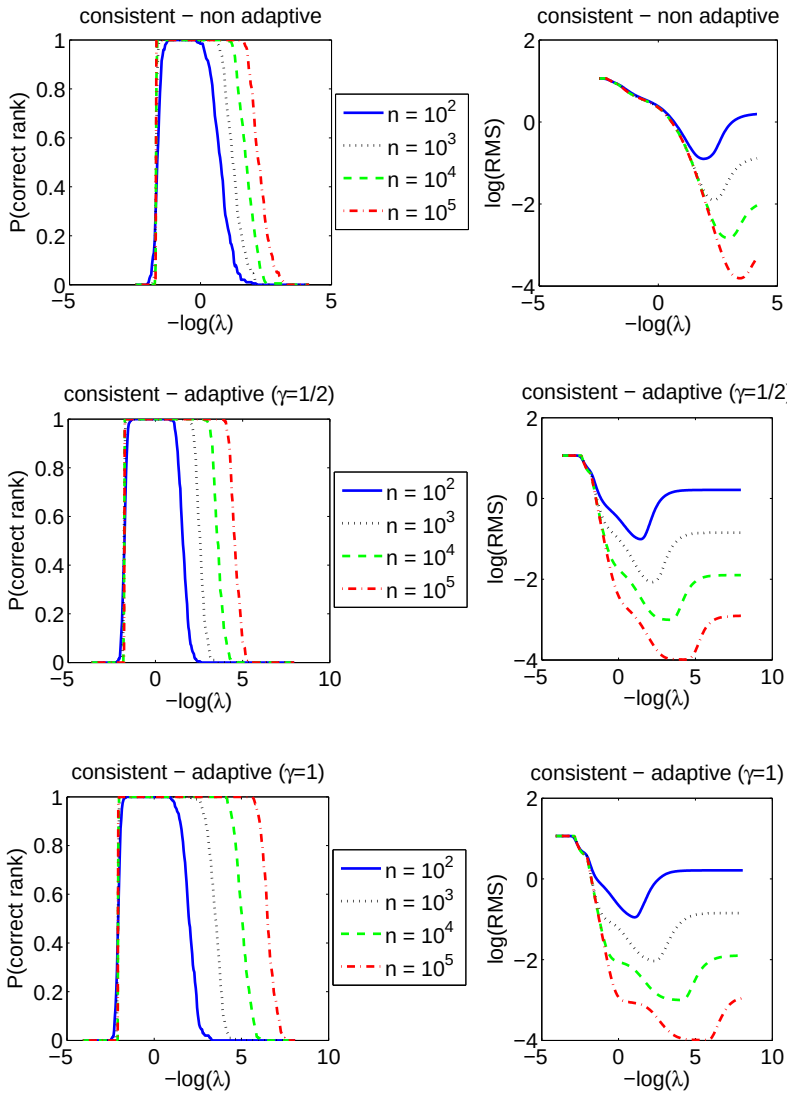


Figure 3: Synthetic example where consistency condition in Eq. (5) is satisfied: probability of correct rank selection (left) and logarithm of the expected mean squared estimation error (right), for several number of samples as a function of the regularization parameter, for regular regularization (top), adaptive regularization with $\gamma = 1/2$ (center) and $\gamma = 1$ (bottom).

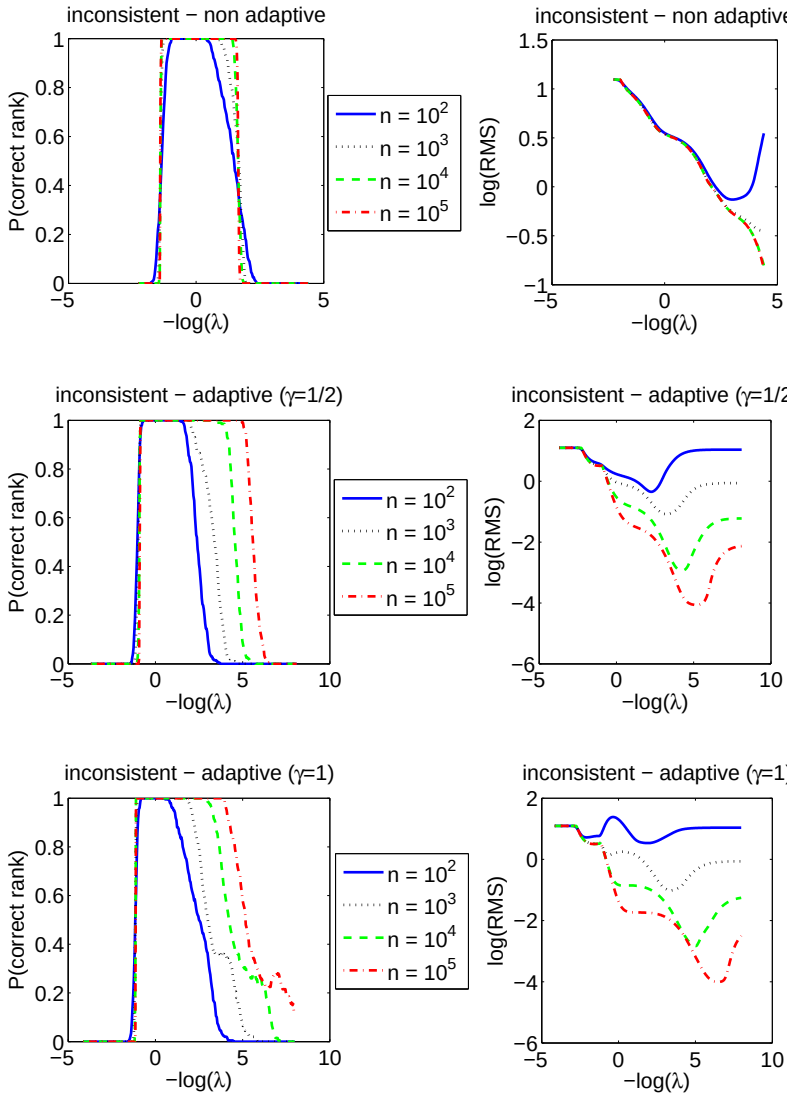


Figure 4: Synthetic example where consistency condition in Eq. (4) is not satisfied: probability of correct rank selection (left) and logarithm of the expected mean squared estimation error (right), for several number of samples as a function of the regularization parameter, for regular regularization (top), adaptive regularization with $\gamma = 1/2$ (center) and $\gamma = 1$ (bottom).

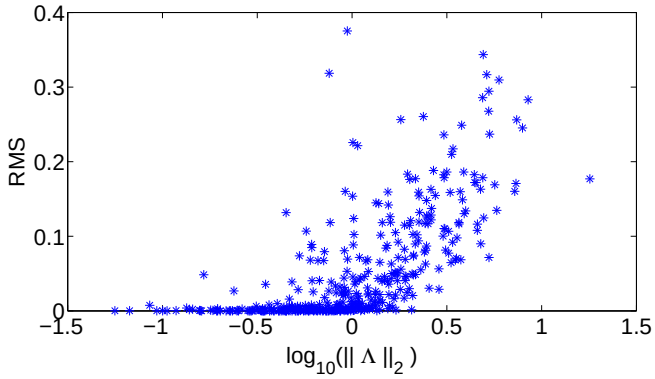


Figure 5: Scatter plots of $\log_{10}(\|\Lambda\|_2)$ versus the squared error of the best estimate with correct rank (i.e., such that $\mathrm{rank}(\hat{W}) = \mathbf{r}$ and $\|\hat{W} - \mathbf{W}\|$ as small as possible). See text for details.

of W). Note that when a singular value s_i is simple, that is, does not coalesce with any other singular values, then the vectors u_i and v_i are uniquely defined up to simultaneous sign flips, that is, only the matrix $u_i v_i^\top$ is unique. However, when some singular values coalesce, then the corresponding singular vectors are defined up to a rotation, and thus in general care must be taken and considering isolated singular vectors should be avoided (Stewart and Sun, 1990). All tools presented in this appendix are robust to the particular choice of the singular vectors.

A.1 Jordan-Wielandt Matrix

We use the fact that singular values of W can be obtained from the eigenvalues of the Jordan-Wielandt matrix $\bar{W} = \begin{pmatrix} 0 & W \\ W^\top & 0 \end{pmatrix} \in \mathbb{R}^{(p+q)\times(p+q)}$ (Stewart and Sun, 1990). Indeed this matrix has eigenvalues s_i and $-s_i$, $i = 1,\ldots,r$, where s_i are the (strictly positive) singular values of W, with eigenvectors $\frac{1}{\sqrt{2}}\begin{pmatrix} u_i \\ v_i \end{pmatrix}$ and $\frac{1}{\sqrt{2}}\begin{pmatrix} u_i \\ -v_i \end{pmatrix}$ where u_i, v_i are the left and right associated singular vectors. Also, the remaining eigenvalues are all equal to zero, with eigensubspace (of dimension $p+q-2r$) composed of all $\begin{pmatrix} u \\ v \end{pmatrix}$ such that for all $i \in \{1,\ldots,r\}$, $u^\top u_i = v^\top v_i = 0$. We let denote $\bar{U}$ the eigenvectors of $\bar{W}$ corresponding to non zero eigenvalues in $\bar{S}$. We have $\bar{U} = \frac{1}{\sqrt{2}}\begin{pmatrix} U & U \\ V & -V \end{pmatrix}$ and $\bar{S} = \frac{1}{\sqrt{2}}\begin{pmatrix} \mathrm{Diag}(s) & 0 \\ 0 & -\mathrm{Diag}(s) \end{pmatrix}$ and $\bar{W} = \bar{U}\bar{S}\bar{U}^\top$, $\bar{U}\bar{U}^\top = \begin{pmatrix} UU^\top & 0 \\ 0 & VV^\top \end{pmatrix}$, and $\bar{U}\mathrm{sign}(\bar{S})\bar{U}^\top = \begin{pmatrix} 0 & UV^\top \\ VU^\top & 0 \end{pmatrix}$.

A.2 Cauchy Residue Formula and Eigenvalues

Given the matrix $\bar{W}$, and a simple closed curve $\mathcal{C}$ in the complex plane that does not go through any of the eigenvalues of $\bar{W}$, then

$$\Pi_{\mathcal{C}}(\bar{W}) = \frac{1}{2i\pi}\oint_{\mathcal{C}} \frac{d\lambda}{\lambda\mathbf{I} - \bar{W}}$$

is equal to the orthogonal projection onto the orthogonal sum of all eigensubspaces of $\bar{W}$ associated with eigenvalues in the interior of $\mathcal{C}$ (Kato, 1966). This is easily seen by writing down the eigenvalue decomposition and the Cauchy residue formula ($\frac{1}{2i\pi}\oint_{\mathcal{C}}\frac{d\lambda}{\lambda-\lambda_i}=1$ if λ_i is in the interior $\text{int}(\mathcal{C})$ of $\mathcal{C}$ and 0 otherwise), and:

$$\frac{1}{2i\pi}\oint_{\mathcal{C}}\frac{d\lambda}{\lambda\mathbf{I}-\bar{W}}=\sum_{i=1}^{2r}\bar{u}_i\bar{u}_i^\top\times\frac{1}{2i\pi}\oint_{\mathcal{C}}\frac{d\lambda}{\lambda-\bar{s}_i}=\sum_{i,\,\bar{s}_i\in\text{int}(\mathcal{C})}u_iu_i^\top.$$

See Rudin (1987) for an introduction to complex analysis and Cauchy residue formula. Moreover, we can obtain the restriction of $\bar{W}$ onto a specific eigensubspace as:

$$\bar{W}\Pi_{\mathcal{C}}(\bar{W})=\frac{1}{2i\pi}\oint_{\mathcal{C}}\frac{\bar{W}d\lambda}{\lambda\mathbf{I}-\bar{W}}=-\frac{1}{2i\pi}\oint_{\mathcal{C}}\frac{\lambda d\lambda}{\lambda\mathbf{I}-\bar{W}}.$$

We let denote s_1 and s_r the largest and smallest strictly positive singular values of W; if $\|\Delta\|_2<s_r/2$, then $W+\Delta$ has r singular values strictly greater than $s_r/2$ and the remaining ones are strictly less than $s_r/2$ (Stewart and Sun, 1990). Thus, if we denote $\mathcal{C}$ the oriented circle of radius $s_r/2$, $\Pi_{\mathcal{C}}(\bar{W})$ is the projector on the $p+q-2r$-dimensional null space of $\bar{W}$, and for any Δ such that $\|\Delta\|_2<s_r/2$, $\Pi_{\mathcal{C}}(\bar{W}+\bar{\Delta})$ is also the projector on the $p+q-2r$-dimensional invariant subspace of $\bar{W}+\bar{\Delta}$, which corresponds to the smallest eigenvalues. We let denote $\Pi_o(\bar{W}+\bar{\Delta})$ that projector and $\Pi_r(\bar{W}+\bar{\Delta})=\mathbf{I}-\Pi_o(\bar{W}+\bar{\Delta})$ the orthogonal projector (which is the projection onto the $2r$-th principal subspace).

We can now find expansions around $\Delta=0$ as follows:

$$\begin{aligned}\Pi_o(\bar{W}+\bar{\Delta})-\Pi_o(\bar{W}) &= \frac{1}{2i\pi}\oint_{\mathcal{C}}(\lambda\mathbf{I}-\bar{W})^{-1}\bar{\Delta}(\lambda\mathbf{I}-\bar{W}-\bar{\Delta})^{-1}d\lambda\\ &= \frac{1}{2i\pi}\oint_{\mathcal{C}}(\lambda\mathbf{I}-\bar{W})^{-1}\bar{\Delta}(\lambda\mathbf{I}-\bar{W})^{-1}d\lambda\\ &\quad+\frac{1}{2i\pi}\oint_{\mathcal{C}}(\lambda\mathbf{I}-\bar{W})^{-1}\bar{\Delta}(\lambda\mathbf{I}-\bar{W})^{-1}\bar{\Delta}(\lambda\mathbf{I}-\bar{W}-\bar{\Delta})^{-1}d\lambda,\end{aligned}$$

and

$$\begin{aligned}(\bar{W}+\bar{\Delta})\Pi_o(\bar{W}+\bar{\Delta})-\bar{W}\Pi_o(\bar{W}) &= -\frac{1}{2i\pi}\oint_{\mathcal{C}}\lambda(\lambda\mathbf{I}-\bar{W})^{-1}\bar{\Delta}(\lambda\mathbf{I}-\bar{W}-\bar{\Delta})^{-1}d\lambda\\ &= -\frac{1}{2i\pi}\oint_{\mathcal{C}}\lambda(\lambda\mathbf{I}-\bar{W})^{-1}\bar{\Delta}(\lambda\mathbf{I}-\bar{W})^{-1}d\lambda\\ &\quad-\frac{1}{2i\pi}\oint_{\mathcal{C}}\lambda(\lambda\mathbf{I}-\bar{W})^{-1}\bar{\Delta}(\lambda\mathbf{I}-\bar{W})^{-1}\bar{\Delta}(\lambda\mathbf{I}-\bar{W}-\bar{\Delta})^{-1}d\lambda,\end{aligned}$$

which lead to the following two propositions:

Proposition 16 *Assume W has rank r and $\|\Delta\|_2<s_r/4$ where s_r is the smallest positive singular value of W. Then the projection $\Pi_r(\bar{W})$ on the first r eigenvectors of $\bar{W}$ is such that*

$$\|\Pi_o(\bar{W}+\bar{\Delta})-\Pi_o(\bar{W})\|_2\leqslant\frac{4}{s_r}\|\Delta\|_2$$

and

$$\|\Pi_o(\bar{W}+\bar{\Delta})-\Pi_o(\bar{W})-(\mathbf{I}-\bar{U}\bar{U}^\top)\bar{\Delta}\bar{U}\bar{S}^{-1}\bar{U}^\top-\bar{U}\bar{S}^{-1}\bar{U}^\top\bar{\Delta}(\mathbf{I}-\bar{U}\bar{U}^\top)\|_2\leqslant\frac{8}{s_r^2}\|\bar{\Delta}\|_2^2.$$

Proof For $\lambda \in \mathcal{C}$ we have: $\|(\lambda \mathbf{I} - \bar{W})^{-1}\|_2 \geqslant 2/s_r$ and $\|(\lambda \mathbf{I} - \bar{W} - \bar{\Delta})^{-1}\|_2 \geqslant 4/s_r$, which implies

$$\begin{aligned}\|\Pi_r(\bar{W}+\bar{\Delta}) - \Pi_r(\bar{W})\|_2 &\leqslant \frac{1}{2\pi}\oint_{\mathcal{C}} \|(\lambda \mathbf{I} - \bar{W})^{-1}\|_2 \|\Delta\|_2 \|(\lambda \mathbf{I} - \bar{W} - \bar{\Delta})^{-1}\|_2 \\ &\leqslant \left(\frac{1}{2\pi} 2\pi \frac{s_r}{2}\right) \|\Delta\|_2 \frac{2}{s_r}\frac{4}{s_r}.\end{aligned}$$

In order to prove the other result, we simply need to compute:

$$\begin{aligned}\frac{1}{2i\pi}\oint_{\mathcal{C}} (\lambda \mathbf{I} - \bar{W})^{-1}\bar{\Delta}(\lambda \mathbf{I} - \bar{W})^{-1} d\lambda &= \sum_{i,j} \bar{u}_i \bar{u}_i^\top \Delta \bar{v}_j \bar{v}_j^\top \frac{1}{2i\pi}\oint_{\mathcal{C}} \frac{1}{(\lambda - \bar{s}_i)(\lambda - \bar{s}_j)} d\lambda \\ &= \sum_{i,j} \bar{u}_i \bar{u}_i^\top \Delta \bar{v}_j \bar{v}_j^\top \left(\frac{1_{i \notin \mathrm{int}(\mathcal{C})} 1_{j \in \mathrm{int}(\mathcal{C})}}{\bar{s}_i} + \frac{1_{j \notin \mathrm{int}(\mathcal{C})} 1_{i \in \mathrm{int}(\mathcal{C})}}{\bar{s}_j}\right) \\ &= (\mathbf{I} - \bar{U}\bar{U}^\top)\bar{\Delta}\bar{U}\bar{S}^{-1}\bar{U}^\top + \bar{U}\bar{S}^{-1}\bar{U}^\top\bar{\Delta}(\mathbf{I} - \bar{U}\bar{U}^\top).\end{aligned}$$

■

Proposition 17 *Assume W has rank r and $\|\Delta\|_2 < s_r/4$ where s_r is the smallest positive singular value of W. Then the projection $\Pi_r(\bar{W})$ on the first r eigenvectors of $\bar{W}$ is such that*

$$\|\Pi_o(\bar{W}+\bar{\Delta})(\bar{W}+\bar{\Delta}) - \Pi_o(\bar{W})\bar{W}\|_2 \leqslant 2\|\Delta\|_2$$

and

$$\|\Pi_o(\bar{W}+\bar{\Delta})(\bar{W}+\bar{\Delta}) - \Pi_o(\bar{W})\bar{W} + (\mathbf{I} - \bar{U}\bar{U}^\top)\bar{\Delta}(\mathbf{I} - \bar{U}\bar{U}^\top)\|_2 \leqslant \frac{4}{s_r}\|\bar{\Delta}\|_2^2.$$

Proof For $\lambda \in \mathcal{C}$ we have: $\|(\lambda \mathbf{I} - \bar{W})^{-1}\|_2 \geqslant 2/s_r$ and $\|(\lambda \mathbf{I} - \bar{W} - \bar{\Delta})^{-1}\|_2 \geqslant 4/s_r$, which implies

$$\begin{aligned}\|\Pi_r(\bar{W}+\bar{\Delta}) - \Pi_r(\bar{W})\|_2 &\leqslant \frac{1}{2\pi}\oint_{\mathcal{C}} |\lambda| \|(\lambda \mathbf{I} - \bar{W})^{-1}\|_2 \|\Delta\|_2 \|(\lambda \mathbf{I} - \bar{W} - \bar{\Delta})^{-1}\|_2 \\ &\leqslant \left(\frac{1}{2\pi} 2\pi \frac{s_r}{2}\right) \frac{s_r}{2}\|\Delta\|_2 \frac{2}{s_r}\frac{4}{s_r}.\end{aligned}$$

In order to prove the other result, we simply need to compute:

$$\begin{aligned}-\frac{1}{2i\pi}\oint_{\mathcal{C}} \lambda(\lambda \mathbf{I} - \bar{W})^{-1}\bar{\Delta}(\lambda \mathbf{I} - \bar{W})^{-1} d\lambda &= -\sum_{i,j} \bar{u}_i \bar{u}_i^\top \Delta \bar{v}_j \bar{v}_j^\top \frac{1}{2i\pi}\oint_{\mathcal{C}} \frac{\lambda}{(\lambda - \bar{s}_i)(\lambda - \bar{s}_j)} d\lambda \\ &= -\sum_{i,j} \bar{u}_i \bar{u}_i^\top \Delta \bar{v}_j \bar{v}_j^\top \left(1_{i \in \mathrm{int}(\mathcal{C})} 1_{j \in \mathrm{int}(\mathcal{C})}\right) \\ &= -(\mathbf{I} - \bar{U}\bar{U}^\top)\bar{\Delta}(\mathbf{I} - \bar{U}\bar{U}^\top).\end{aligned}$$

■

The variations of $\Pi(\bar{W})$ translates immediately into variations of the singular projections $UU^\top$ and $VV^\top$. Indeed we get that the first order variation of $UU^\top$ is $-(\mathbf{I} - UU^\top)\Delta V S^{-1}U^\top$ and the variation of V is equal to $-(\mathbf{I} - VV^\top)\Delta^\top U S^{-1}V^\top$, with errors bounded in spectral norm by $\frac{8}{s_r^2}\|\Delta\|_2^2$. Similarly, when restricted to the small singular values, the first order expansion is $(\mathbf{I} - UU^\top)\Delta(\mathbf{I} - VV^\top)$, with error term bounded in spectral norm by $\frac{4}{s_r}\|\Delta\|_2^2$. Those results lead to the following proposition that gives a local sufficient condition for $\mathrm{rank}(W+\Delta) > \mathrm{rank}(W)$:

Proposition 18 *Assume W has rank $r < \min\{p,q\}$ with ordered singular value decomposition $W = U\,\mathrm{Diag}(s)V^\top$. If $\frac{4}{s_r}\|\Delta\|_2^2 < \|(\mathbf{I}-UU^\top)\Delta(\mathbf{I}-VV^\top)\|_2$, then $\mathrm{rank}(W+\Delta) > r$.*

Appendix B. Some Facts about the Trace Norm

In this appendix, we review known properties of the trace norm that we use in this paper. Most of the results are extensions of similar results for the ℓ_1-norm on vectors. First, we have the following result:

Lemma 19 (Dual norm, Fazel et al., 2001) *The trace norm $\|\cdot\|_*$ is a norm and its dual norm is the operator norm $\|\cdot\|$.*

Note that the dual norm $N(W)$ is defined as Boyd and Vandenberghe (2003):

$$N(W) = \sup_{\|V\|_* \leqslant 1} \mathrm{tr} W^\top V.$$

This immediately implies the following result:

Lemma 20 (Fenchel conjugate) *We have: $\max_{W \in \mathbb{R}^{p\times q}} \mathrm{tr} W^\top V - \|W\|_* = 0$ if $\|V\| \leqslant 1$ and $+\infty$ otherwise.*

In this paper, we need to compute the subdifferential and directional derivatives of the trace norm. We have from Recht et al. (2007) or Borwein and Lewis (2000):

Proposition 21 (Subdifferential) *If $W = U\,\mathrm{Diag}(s)V^\top$ with $U \in \mathbb{R}^{p\times m}$ and $V \in \mathbb{R}^{q\times m}$ having orthonormal columns, and $s \in \mathbb{R}^m$ is strictly positive, is the singular value decomposition of W, then $\|W\|_* = \sum_{i=1}^m s_i$ and the subdifferential of $\|\cdot\|_*$ is equal to*

$$\partial\|\cdot\|_*(W) = \left\{UV^\top + M, \text{ such that } \|M\|_2 \leqslant 1,\ U^\top M = 0 \text{ and } MV = 0\right\}.$$

This result can be extended to compute directional derivatives:

Proposition 22 (Directional derivative) *The directional derivative at $W = USV^\top$ is equal to:*

$$\lim_{\varepsilon \to 0^+} \frac{\|W+\varepsilon\Delta\|_* - \|W\|_*}{\varepsilon} = \mathrm{tr} U^\top \Delta V + \|U_\perp^\top \Delta V_\perp\|_*,$$

where $U_\perp \in \mathbb{R}^{p\times(p-m)}$ and $V_\perp \in \mathbb{R}^{q\times(q-m)}$ are any orthonormal complements of U and V.

Proof From the subdifferential, we get the directional derivative (Borwein and Lewis, 2000) as

$$\lim_{\varepsilon \to 0^+} \frac{\|W+\varepsilon\Delta\|_* - \|W\|_*}{\varepsilon} = \max_{V \in \partial\|\cdot\|_*(W)} \mathrm{tr}\Delta^\top V$$

which exactly leads to the desired result. ■

The final result that we use is a bit finer as it gives an upper bound on the error in the previous limit:

Proposition 23 *Let $W = U\,\mathrm{Diag}(s)V^\top$ the ordered singular value decomposition, where* $\mathrm{rank}(W) = r$, $s > 0$ *and $U_\perp$ and $V_\perp$ be orthogonal complement of U and V; then, if $\|\Delta\|_2 \leqslant s_r/4$:*

$$\left| \|W+\Delta\|_* - \|W\|_* - \mathrm{tr} U^\top \Delta V - \|U_\perp^\top \Delta V_\perp\|_* \right| \leqslant 16 \min\{p,q\} \frac{s_1^2}{s_r^3} \|\Delta\|_2^2.$$

Proof The trace norm of $\|W+\Delta\|_*$ may be divided into the sum of the r largest and the sum of the remaining singular values. The sums of the remaining ones are given through Proposition 17 by $\|U_\perp^\top \Delta V_\perp\|_*$ with an error bounded by $\min\{p,q\}\frac{4}{s_r}\|\Delta\|_2^2$. For the first r singular values, we need to upperbound the second derivative of the sum of the r largest eigenvalues of $\bar{W}+\bar{\Delta}$ with strictly positive eigengap, which leads to the given bound by using the same Cauchy residue technique described in Appendix A. ■

Appendix C. Proofs

In this appendix, we give the proofs of the results presented in the paper.

C.1 Proof of Lemma 2

We let denote $S \in \{0,1\}^{n_x \times n_y}$ the sampling matrix; that is, $S_{ij} = 1$ if the pair (i,j) is observed and zero otherwise. We let denote $\tilde{X}$ and $\tilde{Y}$ the data matrices. We can write $M_k = \tilde{X}^\top \delta_{i_k} \delta_{j_k}^\top \tilde{Y}$ and:

$$\begin{aligned}
\frac{1}{n}\sum_{k=1}^n \mathrm{vec}(M_k)\mathrm{vec}(M_k)^\top &= \frac{1}{n}\sum_{k=1}^n (\tilde{Y}\otimes\tilde{X})^\top \mathrm{vec}(\delta_{i_k}\delta_{j_k}^\top)\mathrm{vec}(\delta_{i_k}\delta_{j_k}^\top)^\top (\tilde{Y}\otimes\tilde{X}) \\
&= \frac{1}{n}(\tilde{Y}\otimes\tilde{X})^\top \mathrm{Diag}(\mathrm{vec}(S))(\tilde{Y}\otimes\tilde{X}),
\end{aligned}$$

which leads to (denoting $\hat{\Sigma}_{xx} = n_x^{-1}\tilde{X}^\top\tilde{X}$ and $\hat{\Sigma}_{yy} = n_x^{-1}\tilde{Y}^\top\tilde{Y}$):

$$\left(\frac{1}{n}\sum_{k=1}^n \mathrm{vec}(M_k)\mathrm{vec}(M_k)^\top - \hat{\Sigma}_{yy}\otimes\hat{\Sigma}_{xx}\right) = \frac{1}{n}(\tilde{Y}\otimes\tilde{X})^\top \mathrm{Diag}(\mathrm{vec}(S - n/n_x n_y))(\tilde{Y}\otimes\tilde{X}).$$

We can thus compute the squared Frobenius norm:

$$\begin{aligned}
&\left\| \frac{1}{n}\sum_{k=1}^n \mathrm{vec}(M_k)\mathrm{vec}(M_k)^\top - \hat{\Sigma}_{yy}\otimes\hat{\Sigma}_{xx} \right\|_F^2 \\
= \;& \frac{1}{n^2}\mathrm{tr}\,\mathrm{Diag}(\mathrm{vec}(S-n/n_xn_y))(\tilde{Y}\tilde{Y}^\top\otimes\tilde{X}\tilde{X}^\top)\,\mathrm{Diag}(\mathrm{vec}(S-n/n_xn_y))(\tilde{Y}\tilde{Y}^\top\otimes\tilde{X}\tilde{X}^\top) \\
= \;& \frac{1}{n^2}\sum_{i,j,i',j'} (S_{ij}-n/n_xn_y)(\tilde{Y}\tilde{Y}^\top\otimes\tilde{X}\tilde{X}^\top)_{ij,i'j'}(S_{i'j'}-n/n_xn_y)(\tilde{Y}\tilde{Y}^\top\otimes\tilde{X}\tilde{X}^\top)_{ij,i'j'}.
\end{aligned}$$

We have, by properties of sampling without replacement (Hoeffding, 1963):

$$\begin{aligned}
\mathbb{E}(S_{ij}-n/n_xn_y)(S_{i'j'}-n/n_xn_y) &= n/n_xn_y(1-n/n_xn_y) \text{ if } (i,j)=(i',j'), \\
\mathbb{E}(S_{ij}-n/n_xn_y)(S_{i'j'}-n/n_xn_y) &= -n/n_xn_y(1-n/n_xn_y)\frac{1}{n_xn_y-1} \text{ if } (i,j)\neq(i',j').
\end{aligned}$$

This implies

$$\begin{aligned}\mathbb{E}(\|\frac{1}{n}\sum_{k=1}^n \operatorname{vec}(M_k)\operatorname{vec}(M_k)^\top - \hat\Sigma_{yy}\otimes\hat\Sigma_{xx}\|_F^2|\tilde X,\tilde Y)&\\ = \frac{1}{n_x n_y n}\sum_{i,j}(\tilde Y\tilde Y^\top\otimes\tilde X\tilde X^\top)^2_{ij,ij} - \frac{1}{(n_xn_y-1)n_xn_yn}\sum_{(i,j)\neq(i',j')}(\tilde Y\tilde Y^\top\otimes\tilde X\tilde X^\top)^2_{ij,i'j'}&\\ \leqslant \frac{2}{n_xn_yn}\sum_{i,j}\|\tilde y_j\|^4\|\tilde x_i\|^4.&\end{aligned}$$

This finally implies that

$$\begin{aligned}&\mathbb{E}\left\|\frac{1}{n}\sum_{k=1}^n \operatorname{vec}(M_k)\operatorname{vec}(M_k)^\top - \Sigma_{yy}\otimes\Sigma_{xx}\right\|_F^2\\ \leqslant\ & \frac{4}{n}\sum_{i,j}\mathbb{E}\|x\|^4\mathbb{E}\|y\|^4 + 2\mathbb{E}\|\hat\Sigma_{xx}-\Sigma_{xx}\|_F^2\mathbb{E}\|\hat\Sigma_{yy}\|_F^2 + 2\mathbb{E}\|\hat\Sigma_{yy}-\Sigma_{yy}\|_F^2\|\Sigma_{xx}\|_F^2\\ \leqslant\ & C\mathbb{E}\|x\|^4\mathbb{E}\|y\|^4\times(\frac{1}{n}+\frac{1}{n_y}+\frac{1}{n_x}),\end{aligned}$$

for some constant $C>0$. This implies (**A2**). To prove the asymptotic normality in (**A3**), we use the martingale central limit theorem (Hall and Heyde, 1980) with sequence of σ-fields $\mathcal{F}_{n,k} = \sigma(\tilde X,\tilde Y,\varepsilon_1,\dots,\varepsilon_k,(i_1,j_1),\dots,(i_k,j_k))$ for $k\leqslant n$. We consider $\Delta_{n,k} = n^{-1/2}\varepsilon_{i_kj_k}y_{j_k}\otimes x_{i_k}\in\mathbb{R}^{pq}$ as the martingale difference. We have $\mathbb{E}(\Delta_{n,k}|\mathcal{F}_{n,k-1})=0$ and

$$\mathbb{E}(\Delta_{n,k}\Delta_{n,k}^\top|\mathcal{F}_{n,k-1}) = n^{-1}\sigma^2 y_{j_k}y_{j_k}^\top\otimes x_{i_k}x_{i_k}^\top,$$

with $\mathbb{E}(\|\Delta_{n,k})\|^4) = O(n^{-2})$ because of the finite fourth order moments. Moreover,

$$\sum_{k=1}^n \mathbb{E}(\Delta_{n,k}\Delta_{n,k}^\top|\mathcal{F}_{n,k-1}) = \sigma^2\hat\Sigma_{mm},$$

and thus tends in probability to $\sigma^2\Sigma_{yy}\otimes\Sigma_{xx}$ because of (**A2**). The assumptions of the martingale central limit theorem are met, we have that $\sum_{k=1}^n\operatorname{vec}(\Delta_{n,k})$ is asymptotically normal with mean zero and covariance matrix $\sigma^2\Sigma_{yy}\otimes\Sigma_{xx}$, which concludes the proof.

C.2 Proof of Proposition 4

We may first restrict minimization over the ball $\{W,\ \|W\|_*\leqslant\|\hat\Sigma_{mm}^{-1}\hat\Sigma_{Mz}\|_*\}$ because the optimum value is less than the value for $W=\hat\Sigma_{mm}^{-1}\hat\Sigma_{Mz}$. Since this random variable is bounded in probability, we can reduce the problem to a compact set. The sequence of continuous random functions $W\mapsto\frac{1}{2}\operatorname{vec}(W-\mathbf{W})^\top\hat\Sigma_{mm}\operatorname{vec}(W-\mathbf{W}) - \operatorname{tr}W^\top\hat\Sigma_{M\varepsilon}+\lambda_n\|W\|_*$ converges pointwise in probability to $W\mapsto\frac{1}{2}\operatorname{vec}(W-\mathbf{W})^\top\Sigma_{mm}\operatorname{vec}(W-\mathbf{W})+\lambda_0\|W\|_*$ with a unique global minimum (because Σ_{mm} is assumed invertible). We can thus apply standard result of consistency in M-estimation (Van der Vaart, 1998; Shao, 2003).

C.3 Proof of Proposition 7

We consider the result of Proposition 6: $\hat{\Delta} = n^{1/2}(\hat{W} - \mathbf{W})$ is asymptotically normal with mean zero and covariance $\sigma^2 \Sigma_{mm}^{-1}$. By Proposition 18 in Appendix B, if $\frac{4n^{-1/2}}{s_r}\|\hat{\Delta}\|_2^2 < \|\mathbf{U}_\perp^\top \hat{\Delta} \mathbf{V}_\perp\|_2$, then $\mathrm{rank}(\hat{W}) > \mathbf{r}$. For a random variable Θ with normal distribution with mean zero and covariance matrix $\sigma^2 \Sigma_{mm}^{-1}$, we let denote $f(C) = \mathbb{P}(\frac{4C^{-1/2}}{s_r}\|\Theta\|_2^2 < \|\mathbf{U}_\perp^\top \Theta \mathbf{V}_\perp\|_2)$. By the dominated convergence theorem, $f(C)$ converges to one when $C \to \infty$. Let $\varepsilon > 0$, thus there exists $C_0 > 0$ such that $f(C_0) > 1 - \varepsilon/2$. By the asymptotic normality result, $\mathbb{P}(\frac{4C_0^{-1/2}}{s_r}\|\hat{\Delta}\|_2^2 < \|\mathbf{U}_\perp^\top \hat{\Delta} \mathbf{V}_\perp\|_2)$ converges to $f(C_0)$ thus $\exists n_0 > 0$ such that $\forall n > n_0$, $\mathbb{P}(\frac{4C_0^{-1/2}}{s_r}\|\hat{\Delta}\|_2^2 < \|\mathbf{U}_\perp^\top \hat{\Delta} \mathbf{V}_\perp\|_2) > f(C_0) - \varepsilon/2 > 1 - \varepsilon$, which concludes the proof, because $\mathbb{P}(\frac{4n^{-1/2}}{s_r}\|\hat{\Delta}\|_2^2 < \|\mathbf{U}_\perp^\top \hat{\Delta} \mathbf{V}_\perp\|_2) \geqslant \mathbb{P}(\frac{4C_0^{-1/2}}{s_r}\|\hat{\Delta}\|_2^2 < \|\mathbf{U}_\perp^\top \hat{\Delta} \mathbf{V}_\perp\|_2)$ as soon as $n > C_0$.

C.4 Proof of Proposition 8

This is the same result as Fu and Knight (2000), but extended to the trace norm minimization, simply using the directional derivative result of Proposition 22 and the epiconvergence theorem from Geyer (1994, 1996). Indeed, if we denote $V_n(\Delta) = \mathrm{vec}(\Delta)^\top \hat{\Sigma}_{mm} \mathrm{vec}(\Delta) - \mathrm{tr}\Delta^\top n^{1/2} \hat{\Sigma}_{M\varepsilon} + \lambda_0 n^{1/2}(\|\mathbf{W} + n^{-1/2}\Delta\|_* - \|\mathbf{W}\|_*)$ and $V(\Delta) = \mathrm{vec}(\Delta)^\top \Sigma_{mm} \mathrm{vec}(\Delta) - \mathrm{tr}\Delta^\top A + \lambda_0 \left[\mathrm{tr}\mathbf{U}^\top \Delta \mathbf{V} + \|\mathbf{U}_\perp^\top \Delta \mathbf{V}_\perp\|_*\right]$, then for each Δ, $V_n(\Delta)$ converges in probability to $V(\Delta)$, and V is strictly convex, which implies that it has an unique global minimum; thus the epi-convergence theorem can be applied, which concludes the proof.

Note that a simpler analysis using regular tools in M-estimation leads to $\hat{W} = \mathbf{W} + n^{-1/2}\hat{\Delta} + o_p(n^{-1/2})$, where $\hat{\Delta}$ is the unique global minimizer of

$$\min_{\Delta \in \mathbb{R}^{p \times q}} \frac{1}{2} \mathrm{vec}(\Delta)^\top \Sigma_{mm} \mathrm{vec}(\Delta) - \mathrm{tr}\Delta^\top (n^{1/2} \hat{\Sigma}_{M\varepsilon}) + \lambda_0 \left[\mathrm{tr}\mathbf{U}^\top \Delta \mathbf{V} + \|\mathbf{U}_\perp^\top \Delta \mathbf{V}_\perp\|_*\right],$$

that is, we can actually take $A = n^{1/2} \hat{\Sigma}_{M\varepsilon}$ (which is asymptotically normal with correct moments).

C.5 Proof of Proposition 9

We let denote $\hat{\Delta} = n^{1/2}(\hat{W} - \mathbf{W})$. We first show that $\limsup_{n \to \infty} \mathbb{P}(\mathrm{rank}(\hat{W}) = \mathbf{r})$ is smaller than the proposed limit a. We consider the following events:

$$\begin{aligned} E_0 &= \{\mathrm{rank}(\hat{W}) = \mathbf{r}\} \\ E_1 &= \{\|n^{-1/2}\hat{\Delta}\|_2 < \mathbf{s}_r/2\} \\ E_2 &= \left\{ \frac{4n^{-1/2}}{s_r}\|\hat{\Delta}\|_2^2 < \|\mathbf{U}_\perp^\top \hat{\Delta} \mathbf{V}_\perp\|_2 \right\}. \end{aligned}$$

By Proposition 18 in Appendix B, we have $E_1 \cap E_2 \subset E_0^c$, and thus it suffices to show that $\mathbb{P}(E_1)$ tends to one, while $\limsup_{n \to \infty} \mathbb{P}(E_2^c) \leqslant a$. The first assertion is a simple consequence of Proposition 8.

Moreover, by Proposition 8, $\hat{\Delta}$ converges in distribution to the unique global optimum $\Delta(A)$ of an optimization problem parameterized by a vector A with normal distribution. For a given $\eta > 0$, we consider the probability $\mathbb{P}(\|\mathbf{U}_\perp^\top \Delta(A) \mathbf{V}_\perp\|_2 \leqslant \eta)$. For any A, when η tends to zero, the indicator function $1_{\|\mathbf{U}_\perp^\top \Delta(A) \mathbf{V}_\perp\|_2 \leqslant \eta}$ converges to $1_{\|\mathbf{U}_\perp^\top \Delta(A) \mathbf{V}_\perp\|_2 = 0}$, which is equal to $1_{\|\Lambda(A)\|_2 \leqslant \lambda_0}$, where

$$\mathrm{vec}(\Lambda(A)) = \Big((\mathbf{V}_\perp \otimes \mathbf{U}_\perp)^\top \Sigma_{mm}^{-1} (\mathbf{V}_\perp \otimes \mathbf{U}_\perp)\Big)^{-1} \Big((\mathbf{V}_\perp \otimes \mathbf{U}_\perp)^\top \Sigma_{mm}^{-1} ((\mathbf{V} \otimes \mathbf{U}) \mathrm{vec}(\mathbf{I}) - \mathrm{vec}(A))\Big).$$

By the dominated convergence theorem, $\mathbb{P}(\|\mathbf{U}_\perp^\top \Delta(A)\mathbf{V}_\perp\|_2 \leqslant \eta)$ converges to

$$a = \mathbb{P}(\|\Lambda(A)\|_2 \leqslant \lambda_0),$$

which is the proposed limit. This limit is in $(0,1)$ because of the normal distribution has an invertible covariance matrix and the set $\{\|\Lambda\|_2 \leqslant 1\}$ and its complement have non empty interiors.

Since $\hat{\Delta} = O_p(1)$, we can instead consider $E_3 = \{\frac{4n^{-1/2}}{s_r} M^2 < \|\mathbf{U}_\perp^\top \hat{\Delta} \mathbf{V}_\perp\|_2\}$ for a particular M, instead of E_2. Then following the same line or arguments than in Appendix C.3, we conclude that $\limsup_{n\to\infty} \mathbb{P}(E_3^c) \leqslant a$, which concludes the first part of the proof.

We now show that $\liminf_{n\to\infty} \mathbb{P}(\operatorname{rank}(\hat{W}) = \mathbf{r}) \geqslant a$. A sufficient condition for rank consistency is the following: we let denote $\hat{W} = USV^\top$ the singular value decomposition of $\hat{W}$ and we let denote U_o and V_o the singular vectors corresponding to all but the $\mathbf{r}$ largest singular values. Since we have simultaneous singular value decompositions, a sufficient condition is that $\operatorname{rank}(\hat{W}) \geqslant \mathbf{r}$ and $\left\|U_o^\top \left(\hat{\Sigma}_{mm}(\hat{W} - \mathbf{W}) - \hat{\Sigma}_{M\varepsilon}\right) V_o\right\|_2 < \lambda_n(1-\eta)$. If $\|\Lambda(n^{1/2}\hat{\Sigma}_{M\varepsilon})\| \leqslant \lambda_0(1-\eta)$, then, by Lemma 11, $\mathbf{U}_\perp^\top \Delta(n^{1/2}\hat{\Sigma}_{M\varepsilon})\mathbf{V}_\perp = 0$, and we get, using the proof of Proposition 8 and the notation $\hat{A} = n^{1/2}\hat{\Sigma}_{M\varepsilon}$:

$$U_o^\top \left(\hat{\Sigma}_{mm}(\hat{W} - \mathbf{W}) - \hat{\Sigma}_{M\varepsilon}\right) V_o \;=\; U_o^\top n^{-1/2} \left(\hat{\Sigma}_{mm}\Delta(\hat{A}) - \hat{A}\right) V_o + o_p(n^{-1/2}).$$

Moreover, because of regular consistency and a positive eigengap for $\mathbf{W}$, the projection onto the first $\mathbf{r}$ singular vectors of $\hat{W}$ converges to the projection onto the first $\mathbf{r}$ singular vectors of $\mathbf{W}$ (see Appendix A), which implies that the projection onto the orthogonal is also consistent, that is, $U_o U_o^\top$ converges in probability to $\mathbf{U}_\perp \mathbf{U}_\perp^\top$ and $V_o V_o^\top$ converges in probability to $\mathbf{V}_\perp \mathbf{V}_\perp^\top$. Thus:

$$\begin{aligned}
\left\|U_o^\top \left(\hat{\Sigma}_{mm}(\hat{W} - \mathbf{W}) - \hat{\Sigma}_{M\varepsilon}\right) V_o\right\|_2 &= \left\|U_o U_o^\top \left(\hat{\Sigma}_{mm}(\hat{W} - \mathbf{W}) - \hat{\Sigma}_{M\varepsilon}\right) V_o V_o^\top\right\|_2 \\
&= n^{-1/2}\|\mathbf{U}_\perp \mathbf{U}_\perp^\top (\hat{\Sigma}_{mm}\Delta(\hat{A}) - \hat{A})\mathbf{V}_\perp \mathbf{V}_\perp^\top\|_2 + o_p(n^{-1/2}) \\
&= n^{-1/2}\|\Lambda(A)\|_2 + o_p(n^{-1/2}).
\end{aligned}$$

This implies that

$$\liminf_{n\to\infty} \left\|U_o^\top \left(\hat{\Sigma}_{mm}(\hat{W} - \mathbf{W}) - \hat{\Sigma}_{M\varepsilon}\right) V_o\right\|_2 < \lambda_n(1-\eta) \geqslant \liminf_{n\to\infty} \mathbb{P}(\|\Lambda(\hat{A})\|_2 \leqslant \lambda_0(1-\eta))$$

which converges to a when η tends to zero, which concludes the proof.

C.6 Proof of Proposition 10

This is the same result as Fu and Knight (2000), but extended to the trace norm minimization, simply using the directional derivative result of Proposition 22. If we write $\hat{W} = \mathbf{W} + \lambda_n \hat{\Delta}$, then $\hat{\Delta}$ is defined as the global minimum of

$$\begin{aligned}
V_n(\Delta) &= \frac{1}{2}\operatorname{vec}(\Delta)^\top \hat{\Sigma}_{mm} \operatorname{vec}(\Delta) - \lambda_n^{-1} \operatorname{tr}\Delta^\top \hat{\Sigma}_{M\varepsilon} + \lambda_n^{-1}(\|\mathbf{W} + \lambda_n \Delta\|_* - \|\mathbf{W}\|_*) \\
&= \frac{1}{2}\operatorname{vec}(\Delta)^\top \Sigma_{mm} \operatorname{vec}(\Delta) + O_p(\zeta_n \|\Delta\|_2^2) + O_p(\lambda_n^{-1} n^{-1/2}) + \operatorname{tr}\Delta^\top \hat{\Sigma}_{M\varepsilon} \\
&\quad + \operatorname{tr}\mathbf{U}^\top \Delta \mathbf{V} + \|\mathbf{U}_\perp^\top \Delta \mathbf{V}_\perp\|_* + O_p(\lambda_n \|\Delta\|_2^2) \\
&= V(\Delta) + O_p(\zeta_n \|\Delta\|_2^2) + O_p(\lambda_n^{-1} n^{-1/2}) + O_p(\lambda_n \|\Delta\|_2^2).
\end{aligned}$$

More precisely, if $M\lambda_n < \mathbf{s}_r/2$,

$$\begin{aligned}\mathbb{E}\sup_{\|\Delta\|_2 \leqslant M} |V_n(\Delta) - V(\Delta)| &= \text{cst} \times \left(M^2 \mathbb{E}\|\hat{\Sigma}_{mm} - \Sigma_{mm}\|_F + M\lambda_n^{-1}\mathbb{E}(\|\hat{\Sigma}_{M\varepsilon}\|^2)^{1/2} + \lambda_n M^2 \right) \\ &= O(M^2\zeta_n + M\lambda_n^{-1}n^{-1/2} + \lambda_n M^2).\end{aligned}$$

Moreover, $V(\Delta)$ achieves its minimum at a bounded point $\Delta_0 \neq 0$. Thus, by Markov inequality the minimum of $V_n(\Delta)$ over the ball $\|\Delta\|_2 < 2\|\Delta_0\|_2$ is with probability tending to one strictly inside and is thus also the unconstrained minimum, which leads to the proposition.

C.7 Proof of Proposition 11

The optimal $\Delta \in \mathbb{R}^{p\times q}$ should be such that $\mathbf{U}_\perp^\top \Delta \mathbf{V}_\perp$ has low rank, where $\mathbf{U}_\perp \in \mathbb{R}^{p\times(p-\mathbf{r})}$ and $\mathbf{V}_\perp \in \mathbb{R}^{q\times(q-\mathbf{r})}$ are orthogonal complements of the singular vectors $\mathbf{U}$ and $\mathbf{V}$. We now derive the condition under which the optimal Δ is such that $\mathbf{U}_\perp^\top \Delta \mathbf{V}_\perp$ is actually equal to zero: we consider the minimum of $\frac{1}{2}\operatorname{vec}(\Delta)^\top \Sigma_{mm} \operatorname{vec}(\Delta) + \operatorname{vec}(\Delta)^\top \operatorname{vec}(\mathbf{U}\mathbf{V}^\top)$ with respect to Δ such that $\operatorname{vec}(\mathbf{U}_\perp^\top \Delta \mathbf{V}_\perp) = (\mathbf{V}_\perp \otimes \mathbf{U}_\perp)^\top \operatorname{vec}(\Delta) = 0$. The solution of that constrained optimization problem is obtained through the following linear system (Boyd and Vandenberghe, 2003):

$$\begin{pmatrix} \Sigma_{mm} & (\mathbf{V}_\perp \otimes \mathbf{U}_\perp) \\ (\mathbf{V}_\perp \otimes \mathbf{U}_\perp)^\top & 0 \end{pmatrix} \begin{pmatrix} \operatorname{vec}(\Delta) \\ \operatorname{vec}(\Lambda) \end{pmatrix} = \begin{pmatrix} -\operatorname{vec}(\mathbf{U}\mathbf{V}^\top) \\ 0 \end{pmatrix},$$

where $\Lambda \in \mathbb{R}^{(p-\mathbf{r})\times(q-\mathbf{r})}$ is the Lagrange multiplier for the equality constraint. We can solve explicitly for Δ and Λ which leads to

$$\operatorname{vec}(\Lambda) = \left((\mathbf{V}_\perp \otimes \mathbf{U}_\perp)^\top \Sigma_{mm}^{-1} (\mathbf{V}_\perp \otimes \mathbf{U}_\perp) \right)^{-1} \left((\mathbf{V}_\perp \otimes \mathbf{U}_\perp)^\top \Sigma_{mm}^{-1} (\mathbf{V} \otimes \mathbf{U}) \operatorname{vec}(\mathbf{I}) \right),$$

and

$$\operatorname{vec}(\Delta) = -\Sigma_{mm}^{-1} \operatorname{vec}(\mathbf{U}\mathbf{V}^\top - \mathbf{U}_\perp \Lambda \mathbf{V}_\perp^\top).$$

Then the minimum of the function $F(\Delta)$ in Eq. (2) is such that $\mathbf{U}_\perp^\top \Delta \mathbf{V}_\perp = 0$ (and thus equal to Δ defined above) if and only if for all $\Theta \in \mathbb{R}^{p\times q}$, the directional derivative of F at Δ in the direction Θ is nonnegative, that is:

$$\lim_{\varepsilon \to 0^+} \frac{F(\Delta + \varepsilon\Theta) - F(\Delta)}{\varepsilon} \geqslant 0.$$

By Proposition 22, this directional derivative is equal to

$$\begin{aligned}\operatorname{tr}\Theta^\top(\Sigma_{mm}\Delta + \mathbf{U}\mathbf{V}^\top) + \|\mathbf{U}_\perp^\top \Theta \mathbf{V}_\perp\|_* &= \operatorname{tr}\Theta^\top \mathbf{U}_\perp \Lambda \mathbf{V}_\perp + \|\mathbf{U}_\perp^\top \Theta \mathbf{V}_\perp\|_* \\ &= \operatorname{tr}\Lambda^\top \mathbf{U}_\perp^\top \Theta \mathbf{V}_\perp + \|\mathbf{U}_\perp^\top \Theta \mathbf{V}_\perp\|_*.\end{aligned}$$

Thus the directional derivative is always non negative if for all $\Theta' \in \mathbb{R}^{(p-\mathbf{r})\times(q-\mathbf{r})}$, $\operatorname{tr}\Lambda^\top \Theta' + \|\Theta'\|_* \geqslant 0$, that is, if and only if $\|\Lambda\|_2 \leqslant 1$, which concludes the proof.

C.8 Proof of Theorem 12

Regular consistency is obtained by Corollary 5. We consider the problem in Eq. (2) of Proposition 10, where $\lambda_n n^{1/2} \to \infty$ and $\lambda_n \to 0$. Since Eq. (5) is satisfied, the solution Δ indeed satisfies $\mathbf{U}_\perp^\top \Delta \mathbf{V}_\perp = 0$ by Lemma 11.

We have $\hat{W} = \mathbf{W} + \lambda_n \Delta + o_p(\lambda_n)$ and we now show that the optimality conditions are satisfied with rank $\mathbf{r}$. From the regular consistency, the rank of $\hat{W}$ is, with probability tending to one, larger than $\mathbf{r}$ (because the rank is lower semi-continuous function). We now need to show that it is actually equal to $\mathbf{r}$. We let denote $\hat{W} = USV^\top$ the singular value decomposition of $\hat{W}$ and we let denote U_o and V_o the singular vectors corresponding to all but the $\mathbf{r}$ largest singular values. Since we have simultaneous singular value decompositions, we simply need to show that, $\left\| U_o^\top \left(\hat{\Sigma}_{mm}(\hat{W} - \mathbf{W}) - \hat{\Sigma}_{M\varepsilon} \right) V_o \right\|_2 < \lambda_n$ with probability tending to one. We have:

$$\begin{aligned} U_o^\top \left(\hat{\Sigma}_{mm}(\hat{W} - \mathbf{W}) - \hat{\Sigma}_{M\varepsilon} \right) V_o &= U_o^\top \left(\lambda_n \hat{\Sigma}_{mm} \Delta + o_p(\lambda_n) - O_p(n^{-1/2}) \right) V_o \\ &= \lambda_n U_o^\top (\Sigma_{mm} \Delta) V_o + o_p(\lambda_n). \end{aligned}$$

Moreover, because of regular consistency and a positive eigengap for $\mathbf{W}$, the projection onto the first $\mathbf{r}$ singular vectors of $\hat{W}$ converges to the projection onto the first $\mathbf{r}$ singular vectors of $\mathbf{W}$ (see Appendix A), which implies that the projection onto the orthogonal is also consistent, that is, $U_o U_o^\top$ converges in probability to $\mathbf{U}_\perp \mathbf{U}_\perp^\top$ and $V_o V_o^\top$ converges in probability to $\mathbf{V}_\perp \mathbf{V}_\perp^\top$. Thus:

$$\begin{aligned} \left\| U_o^\top \left(\hat{\Sigma}_{mm}(\hat{W} - \mathbf{W}) - \hat{\Sigma}_{M\varepsilon} \right) V_o \right\|_2 &= \left\| U_o U_o^\top \left(\hat{\Sigma}_{mm}(\hat{W} - \mathbf{W}) - \hat{\Sigma}_{M\varepsilon} \right) V_o V_o^\top \right\|_2 \\ &= \lambda_n \| \mathbf{U}_\perp \mathbf{U}_\perp^\top (\Sigma_{mm} \Delta) \mathbf{V}_\perp \mathbf{V}_\perp^\top \|_2 + o_p(\lambda_n) \\ &= \lambda_n \| \Lambda \|_2 + o_p(\lambda_n). \end{aligned}$$

This implies that that the last expression is asymptotically of magnitude strictly less than one, which concludes the proof.

C.9 Proof of Theorem 13

We have seen earlier that if $n^{1/2}\lambda_n$ tends to zero and λ_n tends to zero, then Eq. (4) is necessary for rank-consistency. We just have to show that there is a subsequence that does satisfy this. If $\liminf \lambda_n > 0$, then we cannot have consistency (by Proposition 6), thus if we consider a subsequence, we can always assume that λ_n tends to zero.

We now consider the sequence $n^{1/2}\lambda_n$, and its accumulation points. If zero or $+\infty$ is one of them, then by Propositions 7 and 9, we cannot have rank consistency. Thus, for all accumulation points (which are finite and strictly positive), by considering a subsequence, we are in the situation where $n^{1/2}\lambda_n$ tends to $+\infty$ and λ_n tends to zero, which implies Eq. (4), by definition of Λ in Eq. (3) and Lemma 11.

C.10 Proof of Theorem 15

We let denote $U_{LS}^{\mathbf{r}}$ and $V_{LS}^{\mathbf{r}}$ the first $\mathbf{r}$ columns of U_{LS} and V_{LS} and U_{LS}^o and V_{LS}^o the remaining columns; we also denote $s_{LS}^{\mathbf{r}}$ the corresponding first $\mathbf{r}$ singular values and s_{LS}^o the remaining singular values. From Lemma 14 and results in the appendix, we get that $\|s_{LS}^{\mathbf{r}} - \mathbf{s}\|_2 = O_p(n^{-1/2})$ and $\|s_{LS}^o\|_2 = O_p(n^{-1/2})$ and $\|U_{LS}^{\mathbf{r}}(U_{LS}^{\mathbf{r}})^\top - \mathbf{U}\mathbf{U}^\top\|_2 = O_p(n^{-1/2})$ and $\|V_{LS}^{\mathbf{r}}(V_{LS}^{\mathbf{r}})^\top - \mathbf{V}\mathbf{V}^\top\|_2 = O_p(n^{-1/2})$. By writing $\hat{W}_A = \mathbf{W} + n^{-1/2}\hat{\Delta}_A$, $\hat{\Delta}_A$ is defined as the minimum of

$$\frac{1}{2} \operatorname{vec}(\Delta)^\top \hat{\Sigma}_{mm} \operatorname{vec}(\Delta) - n^{1/2} \operatorname{tr} \Delta^\top \hat{\Sigma}_{M\varepsilon} + n\lambda_n \left(\|A\mathbf{W}B + n^{-1/2} A\Delta B\|_* - \|A\mathbf{W}B\|_* \right).$$

We have:

$$\begin{aligned} A\mathbf{U} &= U_{LS}\,\mathrm{Diag}(s_{LS})^{-\gamma}U_{LS}^{\top}\mathbf{U} \\ &= U_{LS}^{\mathbf{r}}\,\mathrm{Diag}(s_{LS}^{\mathbf{r}})^{-\gamma}(U_{LS}^{\mathbf{r}})^{\top}\mathbf{U} + U_{LS}^{o}\,\mathrm{Diag}(s_{LS}^{o})^{-\gamma}(U_{LS}^{o})^{\top}\mathbf{U} \\ &= \mathbf{U}\,\mathrm{Diag}(\mathbf{s})^{-\gamma} + O_p(n^{-1/2}) + O_p(n^{-1/2}n^{\gamma/2}) \\ &= \mathbf{U}\,\mathrm{Diag}(\mathbf{s})^{-\gamma} + O_p(n^{-1/2}n^{\gamma/2}), \end{aligned}$$

and

$$\begin{aligned} A\mathbf{U}_{\perp} &= U_{LS}\,\mathrm{Diag}(s_{LS})^{-\gamma}U_{LS}^{\top}\mathbf{U}_{\perp} \\ &= U_{LS}^{\mathbf{r}}\,\mathrm{Diag}(s_{LS}^{\mathbf{r}})^{-\gamma}(U_{LS}^{\mathbf{r}})^{\top}\mathbf{U}_{\perp} + U_{LS}^{o}\,\mathrm{Diag}(s_{LS}^{o})^{-\gamma}(U_{LS}^{o})^{\top}\mathbf{U}_{\perp} \\ &= \mathbf{U}_{\perp}\,\mathrm{Diag}(s_{LS}^{o})^{-\gamma} + O_p(n^{\gamma/2-1/2}) \\ &= O_p(n^{\gamma/2}). \end{aligned}$$

Similarly we have: $B\mathbf{V} = \mathbf{V}\,\mathrm{Diag}(\mathbf{s})^{-\gamma} + O_p(n^{-1/2}n^{\gamma/2})$ and $B\mathbf{V} = O_p(n^{\gamma/2})$. We can decompose any $\Delta \in \mathbb{R}^{p\times q}$ as $\Delta = (\mathbf{U}\ \mathbf{U}_{\perp})\begin{pmatrix} \Delta_{\mathbf{rr}} & \Delta_{\mathbf{r}o} \\ \Delta_{o\mathbf{r}} & \Delta_{oo} \end{pmatrix}(\mathbf{V}\ \mathbf{V}_{\perp})^{\top}$. We have assumed that $\lambda_n n^{1/2}n^{\gamma/2}$ tends to infinity. Thus,

- if $\mathbf{U}_{\perp}^{\top}\Delta = 0$ and $\Delta\mathbf{V}_{\perp} = 0$ (i.e., if Δ is of the form $\mathbf{U}\Delta_{\mathbf{rr}}\mathbf{V}^{\top}$),

$$\begin{aligned} n\lambda_n\|A\mathbf{W}B + n^{-1/2}A\Delta B\|_* - \|A\mathbf{W}B\|_* &\leqslant \lambda_n n^{1/2}\|A\Delta B\|_* \\ &= \lambda_n n^{1/2}\|\,\mathrm{Diag}(\mathbf{s})^{-\gamma}\Delta_{\mathbf{rr}}\,\mathrm{Diag}(\mathbf{s})^{-\gamma}\|_* \\ &\qquad + O_p(\lambda_n n^{\gamma/2}) \\ &= O_p(\lambda_n n^{1/2}) \end{aligned}$$

  tends to zero.

- Otherwise, $n\lambda_n\|A\mathbf{W}B + n^{-1/2}A\Delta B\|_* - \|A\mathbf{W}B\|_*$ is larger than $\lambda_n n^{1/2}\|A\Delta B\|_* - 2\|A\mathbf{W}B\|_*$. The term $\|A\mathbf{W}B\|_*$ is bounded in probability because we can write $A\mathbf{W}B = \mathbf{U}\,\mathrm{Diag}(\mathbf{s})^{1-2\gamma}\mathbf{V}^{\top} + O_p(n^{-1/2+\gamma/2})$ and $\gamma \leqslant 1$. Besides, $\lambda_n n^{1/2}\|A\Delta B\|_*$ is tending to infinity as soons as any of $\Delta_{o\mathbf{r}}$, $\Delta_{\mathbf{r}o}$ or $\Delta_{\mathbf{rr}}$ are different from zero. Indeed, by equivalence of finite dimensional norms $\lambda_n n^{1/2}\|A\Delta B\|_*$ is larger than a constant times $\lambda_n n^{1/2}\|A\Delta B\|_F$, which can be decomposed in four pieces along $(\mathbf{U}, \mathbf{U}_{\perp})$ and $(\mathbf{V}, \mathbf{V}_{\perp})$, corresponding asymptotically to Δ_{oo}, $\Delta_{o\mathbf{r}}$, $\Delta_{\mathbf{r}o}$ or $\Delta_{\mathbf{rr}}$. The smallest of those terms grows faster than $\lambda_n n^{1/2+\gamma/2}$, and thus tends to infinity.

Thus, since Σ_{mm} is invertible, by the epi-convergence theorem of Geyer (1994, 1996), $\hat{\Delta}_A$ converges in distribution to the minimum of

$$\frac{1}{2}\,\mathrm{vec}(\Delta)^{\top}\Sigma_{mm}\,\mathrm{vec}(\Delta) - n^{1/2}\mathrm{tr}\Delta^{\top}\hat{\Sigma}_{M\varepsilon},$$

such that $\mathbf{U}_{\perp}^{\top}\Delta = 0$ and $\Delta\mathbf{V}_{\perp} = 0$. This minimum has a simple asymptotic distribution, namely $\Delta = \mathbf{U}\Theta\mathbf{V}^{\top}$ and Θ is asymptotically normal with mean zero and covariance matrix $\sigma^2\left[(\mathbf{V}\otimes\mathbf{U})^{\top}\Sigma_{mm}(\mathbf{V}\otimes\mathbf{U})\right]^{-1}$, which leads to the consistency and the asymptotic normality.

In order to finish the proof, we consider the optimality conditions which can be written as $A\Delta B$ and

$$A^{-1}\left(\hat{\Sigma}_{mm}\hat{\Delta}_A - n^{1/2}\hat{\Sigma}_{M\varepsilon}\right)B^{-1}$$

having simultaneous singular value decompositions with proper decays of singular values, that is, such that the first $\mathbf{r}$ are equal to $\lambda_n n^{1/2}$ and the remaining ones are less than $\lambda_n n^{1/2}$.

From the asymptotic normality we get that $\hat{\Sigma}_{mm}\hat{\Delta}_A - n^{1/2}\hat{\Sigma}_{M\varepsilon}$ is $O_p(1)$, we can thus consider matrices of the form $A^{-1}\Theta B^{-1}$ where Θ is bounded, the same way we considered matrices of the form $A\Delta B$.

We have:

$$\begin{aligned} A^{-1}\mathbf{U} &= U_{LS}\,\mathrm{Diag}(s_{LS})^{\gamma}U_{LS}^{\top}\mathbf{U} \\ &= U_{LS}^{\mathbf{r}}\,\mathrm{Diag}(s_{LS}^{\mathbf{r}})^{\gamma}(U_{LS}^{\mathbf{r}})^{\top}\mathbf{U} + U_{LS}^{o}\,\mathrm{Diag}(s_{LS}^{o})^{\gamma}(U_{LS}^{o})^{\top}\mathbf{U} \\ &= \mathbf{U}\,\mathrm{Diag}(\mathbf{s})^{\gamma} + O_p(n^{-1/2}), \end{aligned}$$

and

$$\begin{aligned} A^{-1}\mathbf{U}_{\perp} &= U_{LS}\,\mathrm{Diag}(s_{LS})^{\gamma}U_{LS}^{\top}\mathbf{U}_{\perp} \\ &= U_{LS}^{\mathbf{r}}\,\mathrm{Diag}(s_{LS}^{\mathbf{r}})^{\gamma}(U_{LS}^{\mathbf{r}})^{\top}\mathbf{U}_{\perp} + U_{LS}^{o}\,\mathrm{Diag}(s_{LS}^{o})^{\gamma}(U_{LS}^{o})^{\top}\mathbf{U}_{\perp} \\ &= O_p(n^{-1/2}) + \mathbf{U}_{\perp}\,\mathrm{Diag}(s_{LS}^{o})^{\gamma}, \end{aligned}$$

with similar expansions for $B^{-1}\mathbf{V}$ and $B^{-1}\mathbf{V}_{\perp}$. We obtain the first order expansion:

$$\begin{aligned} A^{-1}\Theta B^{-1} = \mathbf{U}\,\mathrm{Diag}(\mathbf{s})^{\gamma}\Theta_{\mathbf{rr}}\,\mathrm{Diag}(\mathbf{s})^{\gamma}\mathbf{V}^{\top} + \mathbf{U}_{\perp}\,\mathrm{Diag}(s_{LS}^{o})^{\gamma}\Theta_{o\mathbf{r}}\,\mathrm{Diag}(\mathbf{s})^{\gamma}\mathbf{V}^{\top} \\ + \mathbf{U}\,\mathrm{Diag}(\mathbf{s})^{\gamma}\Theta_{\mathbf{r}o}\,\mathrm{Diag}(s_{LS}^{o})^{\gamma}\mathbf{V}_{\perp}^{\top} + \mathbf{U}_{\perp}\,\mathrm{Diag}(s_{LS}^{o})^{\gamma}\Theta_{oo}\,\mathrm{Diag}(s_{LS}^{o})^{\gamma}\mathbf{V}_{\perp}^{\top} \end{aligned}$$

Because of the regular consistency, the first term is of the order of $\lambda_n n^{1/2}$ (so that the first $\mathbf{r}$ singular values of $\hat{W}$ are strictly positive), while the three other terms have norms less than $O_p(n^{-\gamma/2})$ which is less than $O_p(n^{1/2}\lambda_n)$ by assumption. This concludes the proof.

References

J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. Low-rank matrix factorization with attributes. Technical Report N24/06/MM, Ecole des Mines de Paris, 2006.

J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. Technical Report HAL-00250231, HAL, 2008.

Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. In *Proceedings of the International Conference on Machine Learning*, 2007.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems 19*, 2007.

F. R. Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

F. R. Bach, R. Thibaux, and M. I. Jordan. Computing regularization paths for learning multiple kernels. In *Advances in Neural Information Processing Systems 17*, 2004.

J. F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizbal. *Numerical Optimization Theoretical and Practical Aspects*. Springer, 2003.

J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization*. Number 3 in CMS Books in Mathematics. Springer-Verlag, 2000.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2003.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32: 407, 2004.

M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings American Control Conference*, volume 6, pages 4734–4739, 2001.

W. Fu and K. Knight. Asymptotics for lasso-type estimators. *Annals of Statistics*, 28(5):1356–1378, 2000.

C. J. Geyer. On the asymptotics of constrained *m*-estimation. *Annals of Statistics*, 22(4):1993–2010, 1994.

C. J. Geyer. On the asymptotics of convex stochastic optimization. Technical report, School of Statistics, University of Minnesota, 1996.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

P. Hall and C. C. Heyde. *Martingale Limit Theory and Its Application*. Academic Press, 1980.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

T. Kato. *Perturbation Theory for Linear Operators*. Springer-Verlag, 1966.

A. S. Lewis and H. S. Sendov. Twice differentiable spectral functions. *SIAM J. Mat. Anal. App.*, 23 (2):368–386, 2002.

Z. Lu, R. Monteiro, and M. Yuan. Convex optimization methods for dimension reduction and coefficient estimation in multivariate linear regression. Technical report, Optimization online, 2008. URL `http://www.optimization-online.org/DB_HTML/2008/01/1877.html`.

J. R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley, New York, 1998.

N. Meinshausen and B. Yu. Lasso-type recovery of sparse representations for high-dimensional data. Technical Report 720, Dpt of Statistics, UC Berkeley, 2006.

B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. Technical Report arXiv:0706.4138v1, arXiv, 2007.

J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the International Conference on Machine Learning*, 2005.

W. Rudin. *Real and complex analysis, Third edition.* McGraw-Hill, 1987.

J. Shao. *Mathematical Statistics*. Springer, 2003.

N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, 2005.

G. W. Stewart and J. Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal Royal Statististics*, 58(1): 267–288, 1994.

A. W. Van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 1998.

M. Yuan and Y. Lin. On the non-negative garrotte estimator. *Journal of The Royal Statistical Society Series B*, 69(2):143–161, 2007.

M. Yuan, A. Ekici, Z. Lu, and R. D. C. Monteiro. Dimension reduction and coefficient estimation in the multivariate linear regression. *Journal of the Royal Statistical Society, Series B*, 69(3): 329–346, 2007.

P. Zhao and B. Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429, December 2006.

Journal of Machine Learning Research 9 (2008) 1049-1082 Submitted 5/07; Revised 12/07; Published 6/08

Learning Similarity with Operator-valued Large-margin Classifiers

Andreas Maurer ANDREASMAURER@COMPUSERVE.COM
Adalbertstr. 55
D-80799 München
Germany

Editor: Peter Bartlett

Abstract

A method is introduced to learn and represent similarity with linear operators in kernel induced Hilbert spaces. Transferring error bounds for vector valued large-margin classifiers to the setting of Hilbert-Schmidt operators leads to dimension free bounds on a risk functional for linear representations and motivates a regularized objective functional. Minimization of this objective is effected by a simple technique of stochastic gradient descent. The resulting representations are tested on transfer problems in image processing, involving plane and spatial geometric invariants, handwritten characters and face recognition.

Keywords: learning similarity, similarity, transfer learning

1. Introduction

Similarity seems fundamental to perception and reasoning. The precise meaning of the word "similarity" however is elusive and a corresponding Google search reveals a plethora of definitions ranging from "the property of being similar" to the analyses of Wittgenstein, Russel or Carnap. This is not surprising: Pairs of triangles may or may not be similar, two poems may induce similar or radically different moods in similar people, pairs of wines, bird-calls, weather patterns, approaches to cognitive science, movies and definitions of similarity themselves may each be similar or not similar or similar to a varying degree. It is the very universality of the concept which explains the lack of a universal definition, beyond the structural feature that similarity is a property possessed by pairs of objects and the vague feeling that it is somehow related to geometric proximity.

What cannot be defined may still be learned. Even if we cannot explain the meaning of a concept, as long as it has observable manifestations we can hope to infer models to predict future observations. These models will generally be domain-dependent and not in the form of definitions, but rather vectors of synaptic efficacies, transformation coefficients or other constructions which often defy verbalization, they are more akin to feeling than to rationality and they will be judged by their predictive power rather than by logical clarity and comprehensibility.

This paper introduces a technique to learn and to represent similarity, an analysis of generalization performance, an algorithmic realization and some experimental results.

1.1 A General Framework

Assumption 1: There is a measurable space $\mathcal{X}$ and a probability measure ρ on $\mathcal{X}^2 \times \{-1,1\}$, the *pair oracle*.

The interpretation is as follows: $\mathcal{X}$ is the input space containing the objects in question. Whenever we call the oracle it will return the triplet $(x,x',r) \in \mathcal{X}^2 \times \{-1,1\}$ with probability $\rho(x,x',r)$. If the oracle returns $(x,x',1)$ it asserts that x and x' are *similar*, if it returns $(x,x',-1)$ it asserts that x and x' are *dissimilar*. The assumption that similarity is a binary property, which can only have the values of true or false, is a simplification which can in principle be removed (see Section 4.1 below). Beyond this restriction arbitrary definitions of similarity can be substituted, we do not require "obvious properties" such as $\rho(x,x',r) = \rho(x',x,r)$ or $\rho(x,x,-1) = 0$. We will however use an intuitive property of similarity, a kinship to closeness or geometric proximity, in the choice of our hypothesis spaces below.

Assumption 2: $\mathcal{X} \subset H$, where H is a real separable finite- or infinite-dimensional Hilbert space, and $diam(\mathcal{X}) \leq 1$.

Either the inputs are already members of some Euclidean space (vectors of neural activations, pixel vectors or vectors of feature-values), or we identify them with their images under some feature map, which may be realized by some fixed pre-processor such as a fixed weight neural network or by a positive definite kernel on $\mathcal{X}$. In the more detailed description of the proposed algorithm and experimental results we will be more explicit about these feature maps. The bound on the diameter of the input space is a convenience for the statement of our theoretical results.

Assumption 3: There is a training sample

$$S = \left(\left(x_1,x_1',r_1\right),...,\left(x_m,x_m',r_m\right)\right) \in \left(\mathcal{X}^2 \times \{-1,1\}\right)^m,$$

generated in m independent, identical trials of ρ, that is, $S \sim \rho^m$.

The assumptions of independence and stationarity are crucial and very strong: The oracle is without memory of our previous calls and not affected by the passage of time. It will not deliberately help nor mislead the learner. The training sample contains all the information available to the learner who wants to find a rule to predict the oracles behavior.

Definition 1: A pair hypothesis is a function $f : \mathcal{X}^2 \to \{-1,0,1\}$. Its risk is

$$R(f) = \Pr_{(x,x',r)\sim\rho}\left\{f\left(x,x'\right) \neq r\right\}.$$

The pair hypothesis attempts to predict the similarity value of a pair (x,x') and its risk is its error probability as measured by the pair oracle. We allowed the value 0 to account for the possibility that f may refuse to make a decision. Any such refusal is counted as an error by the risk functional.

1.2 Risk Bounds and Regularized Objectives

So far we have only used the structural condition that similarity is a property possessed by pairs of objects. In the choice of the hypothesis space we will follow the intuition that similarity is related to closeness or geometrical proximity. We will consider hypotheses from the set

$$\mathcal{H} = \left\{f_T : \left(x,x'\right) \mapsto sgn\left(1 - \left\|Tx - Tx'\right\|\right) : T \in \mathcal{L}_0(H)\right\},$$

where $\mathcal{L}_0(H)$ is the set of linear operators of finite rank on H. A transformation $T \in \mathcal{L}_0(H)$ thus defines a hypothesis f_T which regards a pair of inputs as similar if the distance between their respective images under T is smaller than one, and as dissimilar if this distance is larger than one. Fixing the threshold to one causes no loss of generality, because any other positive threshold could be absorbed as a factor of the transformation.

This choice of the hypothesis space combines the geometric connotation of similarity with the simplicity of linear representations. The transformations which parametrize our hypothesis space are in some ways more interesting and useful than the hypotheses themselves. A choice of $T \in \mathcal{L}_0(H)$ also implies a choice of the Mahalanobis distance $d^2(x,x') = \langle T^*T(x-x'), x-x' \rangle$ and the positive semidefinite kernel $\kappa(x,x') = \langle T^*Tx, x' \rangle$.

The risk of the hypothesis f_T induces a risk functional on $\mathcal{L}_0(H)$

$$R(T) = R(f_T) = \Pr_{(x,x',r)\sim\rho} \left\{ r\left(1 - \left\|Tx - Tx'\right\|^2\right) \leq 0 \right\}.$$

Since we can write $\|Tx - Tx'\|^2 = \langle T^*T(x-x'), x-x' \rangle = \langle T^*T, Q_{x-x'} \rangle_2$, where $Q_{x-x'}$ and $\langle .,. \rangle_2$ are respectively the outer product operator and the Hilbert-Schmidt inner product (see Section 2.1), the last expression is reminiscent of the risk of a classifier defined by a linear function thresholded at 1. This provides the intuition underlying the proposed technique: We will look for a linear large-margin classifier whose weight vector is the positive operator T^*T.

Let $\psi : \mathbb{R} \to \mathbb{R}$, $\psi \geq 1_{(-\infty,0]}$ with Lipschitz constant L. Given our training sample $S = ((x_1, x_1', r_1), \ldots, (x_m, x_m', r_m))$ we define the empirical risk estimate

$$\hat{R}_\psi(T,S) = \frac{1}{m} \sum_{i=1}^{m} \psi\left(r_i\left(1 - \left\|T\left(x_i - x_i'\right)\right\|^2\right)\right). \tag{1}$$

We then have the following theorem, a proof of which will be given in Section 3.

Theorem 1 *$\forall \delta > 0$, with probability greater $1-\delta$ in a sample $S \sim \rho^m$*
$\forall T \in \mathcal{L}_0(H)$ with $\|T^*T\|_2 \geq 1$

$$R(T) \leq \hat{R}_\psi(T,S) + \frac{4L\ \|T^*T\|_2 + \sqrt{(1/2)\ln(2\|T^*T\|_2/\delta)}}{\sqrt{m}}.$$

where $\|A\|_2 = Tr(A^*A)^{1/2}$ is the Hilbert-Schmidt- or Frobenius- norm of A.

The theorem gives a high-probability-bound on the true risk valid for all transformations in terms of the empirical risk estimate and a complexity term. Because of the uniform nature of the bound, a principled approach could search for $T \in \mathcal{L}_0(H)$ to minimize the right side of the bound. This is just what we propose to do, with two practical modifications:

- The term $\sqrt{(1/2)\ln(2\|T^*T\|_2/\delta)}$ is neglected, the one linear in $\|T^*T\|_2$ being regarded as the dominant contribution.
- The factor $4L$ is replaced by an adjustable regularization parameter $\lambda > 0$. This allows to compensate the fact that the difficult estimates in such generalization bounds overestimate the estimation error.

As a Lipschitz function ψ we use h_γ, the hinge-loss with margin γ, which has the value $1 - t/\gamma$ for $t < \gamma$, and the value 0 otherwise. This leads to the regularized objective function

$$\Lambda_{h_\gamma,\lambda}(T) := \frac{1}{m} \sum_{i=1}^{m} h_\gamma\left(r_i\left(1 - \left\|T\left(x_i - x_i'\right)\right\|^2\right)\right) + \frac{\lambda\|T^*T\|_2}{\sqrt{m}},$$

which is convex in T^*T. Details related to the minimization of $\Lambda_{\gamma,\lambda}$ are given in Section 3.3. It follows from the nature of the objective function that the minimizing operator will have rank $\leq m$.

1.3 The Multi-category Problem and Learning-to-learn

In many classification problems occurring in a complex environment a learner cannot hope to obtain examples for all potential categories. An example is furnished by the recognition of human faces, since nobody can be expected to ever see all the faces which possibly need to be distinguished in the future and the total number of categories is itself uncertain.

Human learning appears to cope with these empirically deficient problems: In addition to the recognition of the already known categories human learners develop meta-techniques to improve the learning and recognition of future categories. As a child learns to recognize parents, family, friends, neighbors, classmates and teachers it learns to memorize, recognize and distinguish faces in general, an ability which leads to a reliable recall often already on the basis of a single training image. The earlier learning process leading to an improvement of future learning performance is often referred to as *learning-to-learn* or *meta-learning*.

The practical utility of such mechanisms for machine-learning is obvious, and theoretically well founded models of learning-to-learn may also be interesting from the point of view of cognitive science or psychology (see Robins, 1998; Thrun, 1998; Baxter, 1998 for surveys, theoretical and experimental contributions).

Here we exploit the fact that partial empirical knowledge of some of the categories of a domain implies partial empirical knowledge of an underlying principle of similarity. By a very crude operational definition similarity is a property of two phenomena which makes them belong to the same category of some domain, and dissimilarity is the negation of similarity. Following this idea we can define a pair oracle which regards pairs of inputs as similar if and only if they come with the same label, use this oracle to generate a large number of examples and train our algorithm to obtain a similarity rule, together with a representing operator T.

Subsequently, once a novel category is represented by a first example, any new phenomenon can be classified as belonging to the given category if and only if it is similar to the representing example. Let us call this decision rule the *elementary verifier* generated from the example. It follows from our analysis, that we can be confident that a single randomly chosen example for a future (possibly previously unseen) category will give an elementary verifier with expected error bounded by the bound in Theorem 1. These constructions and some related questions will be discussed in Section 5.

A related concept is *transfer*, where a representation trained from the data-set of a training-task is applied to facilitate the learning of another, presumably related, target-task. Corresponding experiments were carried out on a number of problems in image recognition, such as the recognition of handwritten characters, rotation and scale invariant character recognition and the recognition of human faces. In all these cases the representations generated from the training-tasks yielded a considerable performance improvement for single-sample nearest neighbor classifiers on the target-tasks.

This paper is organized as follows: Section 2 gives notation and theoretical background, Section 3 introduces operator valued large-margin classifiers and a corresponding algorithm to train representations, Section 4 derives an alternative algorithm related to PCA, Section 5 discusses the application of our method to meta-learning and transfer, Section 6 describes experimental results and Section 7 gives a brief review of some related approaches in the literature.

A precursor of this paper appeared in the NIPS'06 workshop on "Learning to Compare Examples" (Maurer, 2006c).

2. Notation, Definitions and Preliminary Results

In this section we introduce some notation and necessary theoretical background. For the readers convenience there is also an appendix with a tabular summary of most of the notation used in this paper.

2.1 Hilbert Space and Hilbert Schmidt Operators

The letter H denotes a finite- or infinite-dimensional real, separable Hilbert space, with inner product $\langle .,. \rangle$ and norm $\|.\|$. Inner products and norms on other spaces will be identified with subscripts. If $\left(\mathcal{S}, \|.\|_{\mathcal{S}}\right)$ is a generic normed space and $E \subseteq \mathcal{S}$, we will use the notation

$$\|E\|_{\mathcal{S}} = \sup_{x \in E} \|x\|_{\mathcal{S}}.$$

If H' is another Hilbert space, with inner product $\langle .,. \rangle'$ and norm $\|.\|'$, then $\mathcal{L}_\infty(H,H')$ denotes the Banach space of linear transformations $T : H \to H'$ such that

$$\|T\|_\infty = \sup_{x \in H, \|x\| \leq 1} \|Tx\|' < \infty.$$

For $T \in \mathcal{L}_\infty(H,H')$, $Ker(T)$ is the subspace $\{x : Tx = 0\}$ and T^* is the unique member of $\mathcal{L}_\infty(H',H)$ satisfying $\langle Tx, y \rangle' = \langle x, T^* y \rangle$, $\forall x \in H, y \in H'$. If $S \subset H$ then $S^\perp$ is the subspace $\{x : \langle x, y \rangle = 0, \forall y \in S\}$. A transformation $U \in \mathcal{L}_\infty(H,H')$ is called a partial isometry if $\|Ux\| = \|x\|$, $\forall x \in Ker(U)^\perp$.

We write $\mathcal{L}_\infty(H) = \mathcal{L}_\infty(H,H)$. An operator $T \in \mathcal{L}_\infty(H)$ is called symmetric if $T = T^*$ and positive if it is symmetric and $\langle Tx, x \rangle \geq 0$, $\forall x \in H$. We use $\mathcal{L}_\infty^*(H)$ and $\mathcal{L}_\infty^+(H)$ to denote the sets of symmetric and positive members of $\mathcal{L}_\infty(H)$ respectively. For every $T \in \mathcal{L}_\infty^+(H)$ there is some unique $T^{1/2} \in \mathcal{L}_\infty^+(H)$ with $T = T^{1/2} T^{1/2}$. Evidently for every $T \in \mathcal{L}_\infty(H,H')$ we have $T^*T \in \mathcal{L}_\infty^+(H)$ and we denote with $|T|$ the positive operator $(T^*T)^{1/2}$. For every $T \in \mathcal{L}_\infty(H,H')$ there is a polar decomposition $T = U|T|$ for a unique partial isometry $U \in \mathcal{L}_\infty(H,H')$.

For $\mathcal{V} \subseteq \mathcal{L}_\infty(H)$ we use the notation $\mathcal{V}^*\mathcal{V} = \left\{T^*T : T \in \mathcal{V}\right\}$. The set of finite rank operators on H is denoted by $\mathcal{L}_0(H)$. Any orthonormal basis establishes a one-to-one correspondence between $\mathcal{L}_0(H)$ and $\bigcup_{n=1}^\infty \{T : H \to \mathbb{R}^n : T \text{ linear}\}$.

With $\mathcal{L}_2(H)$ we denote the real vector space of operators $T \in \mathcal{L}_\infty(H)$ satisfying $\sum_{i=1}^\infty \|Te_i\|^2 \leq \infty$ for every orthonormal basis $(e_i)_{i=1}^\infty$ of H. The members of $\mathcal{L}_2(H)$ are compact and called *Hilbert Schmidt operators*. For $S, T \in \mathcal{L}_2(H)$ and an orthonormal basis (e_i) the series $\sum_i \langle Se_i, Te_i \rangle$ is absolutely summable and independent of the chosen basis. The number

$$\langle S, T \rangle_2 = \sum_i \langle Se_i, Te_i \rangle$$

defines an inner product on $\mathcal{L}_2(H)$, making it a Hilbert space. We denote the corresponding norm with $\|.\|_2$ (see Reed and Simon, 1980 for background on functional analysis). $\mathcal{L}_2^*(H)$ and $\mathcal{L}_2^+(H)$ denote the sets of symmetric and positive members of $\mathcal{L}_2(H)$ respectively. For every member of $\mathcal{L}_2^*(H)$ there is a complete orthonormal basis of eigenvectors, and for $T \in \mathcal{L}_2^*(H)$ the norm $\|T\|_2$ is just the ℓ_2-norm of its sequence of eigenvalues. In the finite dimensional case the norm $\|T\|_2$ is the Frobenius norm of the matrix of T in an orthonormal representation.

The set of d-dimensional, orthogonal projections in H is denoted with $\mathcal{P}_d$. It is easy to verify that $\mathcal{P}_d \subset \mathcal{L}_2^+(H)$ and if $P \in \mathcal{P}_d$ then $\|P\|_2 = \sqrt{d}$ and $P^2 = P$.

Definition 2 *For $x \in H$ define an operator Q_x on H by $Q_x z = \langle z, x \rangle x$, $\forall z \in H$.*

The map $x \to Q_x$ is an embedding of H in $\mathcal{L}_2^+(H)$ which is homogeneous of degree two (i.e., $Q_{\lambda x} = \lambda^2 Q_x$, $\lambda \in \mathbb{R}$, $x \in H$). In matrix terminology Q_x is the 'outer product' of x with itself. The following simple lemma is crucial for our proofs (see also Maurer, 2006b).

Lemma 3 *Let $x, y \in H$ and $T \in \mathcal{L}_2(H)$. Then*
(i) $Q_x \in \mathcal{L}_2^+(H)$ and $\|Q_x\|_2 = \|x\|^2$.
(ii) $\langle Q_x, Q_y \rangle_2 = \langle x, y \rangle^2$.
(iii) $\langle T, Q_x \rangle_2 = \langle Tx, x \rangle$.
(iv) $\langle T^* T, Q_x \rangle_2 = \|Tx\|^2$.
(v) For $\alpha \in \mathbb{R}$, $Q_{\alpha x} = \alpha^2 Q_x$.
(vi) For $P \in \mathcal{P}_d$ we have $\|PTP\|_2 \leq \|T\|_2$.

Proof For $x = 0$ all assertions are trivial. Otherwise extend $x/\|x\|$ to an orthonormal basis of H and use this basis in the definition of $\langle T, Q_x \rangle_2$ to obtain $\langle T, Q_x \rangle_2 = \langle Tx/\|x\|, Q_x x/\|x\| \rangle = \|x\|^{-2} \langle Tx, \langle x, x \rangle x \rangle = \langle Tx, x \rangle$, which is (iii). (ii),(iv) and the second half of (i) follow immediately, the first part of (i) then follows from (iii) with $T = Q_x$. (v) is trivial. To prove (vi) complete a basis $\{e_i\}_{i=1,..,d}$ for the range of P to a basis e_i for H to get $\|PTP\|_2^2 = \sum_{ij} \langle PTPe_i, e_j \rangle^2 = \sum_{i,j \leq d} \langle Te_i, e_j \rangle^2 \leq \sum_{ij} \langle Te_i, e_j \rangle^2 = \|T\|_2^2$. ■

2.2 Rademacher Complexities

To derive the uniform laws of large numbers we need for Theorem 1 we will use Rademacher averages as complexity measures for function classes:

Definition 4 *Let $\mathcal{F}$ be a real-valued function class on a space $\mathcal{X}$. Let $\{\sigma_i : i \in \{1, ..., m\}\}$ be a collection of independent random variables, distributed uniformly in $\{-1, 1\}$. The empirical Rademacher complexity of $\mathcal{F}$ is the function $\hat{\mathcal{R}}_m(\mathcal{F})$ defined on $\mathcal{X}^m$ by*

$$\hat{\mathcal{R}}_m(\mathcal{F})(\mathbf{x}) = \mathbb{E}_\sigma \left[\sup_{\mathbf{f} \in \mathcal{F}} \frac{2}{m} \sum_{i=1}^m \sigma_i f(x_i) \right].$$

If $\mathbf{X} = (X_i)_{i=1}^m$ is a vector of $\mathcal{X}$-valued independent random variables then the expected Rademacher complexity of $\mathcal{F}$ is

$$\mathcal{R}_m(\mathcal{F}) = \mathbb{E}_{\mathbf{X}} \left[\hat{\mathcal{R}}_m(\mathcal{F})(\mathbf{X}) \right].$$

Theorem 5 *Let $\mathcal{F}$ be a $[0,1]$-valued function class on a space $\mathcal{X}$, and $\mathbf{X} = (X_i)_{i=1}^m$ a vector of $\mathcal{X}$-valued independent, identically distributed random variables. Fix $\delta > 0$.*

With probability greater than $1 - \delta$ we have for all $f \in \mathcal{F}$

$$E[f(X_1)] \leq \frac{1}{m} \sum_{i=1}^m f(X_i) + \mathcal{R}_m(\mathcal{F}) + \sqrt{\frac{\ln(1/\delta)}{2m}}.$$

We also have with probability greater than $1-\delta$ *for all* $f \in \mathcal{F}$*, that*

$$E\left[f\left(X_1\right)\right] \leq \frac{1}{m}\sum_{i=1}^{m} f\left(X_i\right) + \hat{\mathcal{R}}_m\left(\mathcal{F}\right)\left(\mathbf{X}\right) + \sqrt{\frac{9\ln\left(2/\delta\right)}{2m}}.$$

For a proof see Bartlett and Mendelson (2002) or Maurer (2006b). We will also use the following result from Bartlett et al. (2005):

Theorem 6 *Let* $\mathcal{F}$ *be a class of real-valued functions on a space* $\mathcal{X}$ *and suppose that* $\psi : \mathbb{R} \to \mathbb{R}$ *has Lipschitz constant* L*. Let* $\psi \circ \mathcal{F} = \{\psi \circ f : f \in \mathcal{F}\}$*. Then* $\hat{\mathcal{R}}_m\left(\psi \circ \mathcal{F}\right) \leq L\,\hat{\mathcal{R}}_m\left(\mathcal{F}\right)$*.*

2.3 Bounds for Vector-valued Processes

In this section we review some bounds for vector valued processes and linear classifiers. The bounds are not at all original (they are taken from Koltchinskii and Panchenko, 2002; Bartlett and Mendelson, 2002; Shawe-Taylor and Christianini, 2003), nor are they necessarily the tightest possible, because they are uniformly valid on the chosen function classes (compare with Bartlett et al., 2005). Because the proofs are easy we provide them for the readers convenience.

Using Lemma 3 all these results can be easily transferred from the Hilbert space H to the Hilbert space $\mathcal{L}_2\left(H\right)$. This simple step, together the geometrical interpretation implied by Lemma 3 (iv), is the principal theoretical contribution of this paper.

Lemma 7 *Let* $V \subset H$ *and* $\mathcal{F} = \{x \in H \mapsto \langle x, v\rangle : v \in V\}$*. Then for any* $\mathbf{x} = (x_1, ..., x_m) \in H^m$

$$\hat{\mathcal{R}}_m\left(\mathcal{F}\right)\left(\mathbf{x}\right) \leq \frac{2\left\|V\right\|}{m}\left(\sum_{i=1}^{m}\left\|x_i\right\|^2\right)^{1/2}.$$

Proof From Schwartz' and Jensen's inequality and linearity we obtain

$$\begin{aligned}\hat{\mathcal{R}}_m\left(\mathcal{F}\right)\left(\mathbf{x}\right) &= \mathbb{E}_\sigma\left[\sup_{v\in V}\frac{2}{m}\sum_{i=1}^{m}\sigma_i\left\langle x_i, v\right\rangle\right] = \mathbb{E}_\sigma\left[\sup_{v\in V}\frac{2}{m}\left\langle\sum_{i=1}^{m}\sigma_i x_i, v\right\rangle\right] \\ &\leq \frac{2\left\|V\right\|}{m}\mathbb{E}_\sigma\left[\left\|\sum_{i=1}^{m}\sigma_i x_i\right\|\right] \leq \frac{2\left\|V\right\|}{m}\left(\mathbb{E}_\sigma\left[\left\|\sum_{i=1}^{m}\sigma_i x_i\right\|^2\right]\right)^{1/2},\end{aligned}$$

but by the properties of the σ_i we have $\mathbb{E}_\sigma\left[\sigma_i\sigma_j\right] = \delta_{ij}$, so we get

$$\mathbb{E}_\sigma\left[\left\|\sum_{i=1}^{m}\sigma_i x_i\right\|^2\right] = \sum_{i=1}^{m}\sum_{j=1}^{m}\mathbb{E}_\sigma\left[\sigma_i\sigma_j\right]\left\langle x_i, x_j\right\rangle = \sum_{i=1}^{m}\left\|x_i\right\|^2.$$

■

Theorem 8 *Let (X,r) be a random variable with values in $H\times\{-1,1\}$ and let $(\mathbf{X},\mathbf{r})=((X_1,r_1),\ldots,(X_m,r_m))$ be a vector of m iid copies of (X,r). Let $\psi:\mathbb{R}\to\mathbb{R}$, $\psi\geq 1_{(-\infty,0]}$ with Lipschitz constant L and $V\subset H$. For $v\in V$ denote $err(v)=\Pr\{sign(1-\langle X,v\rangle)\neq r\}$ and*

$$\hat{err}_\psi(v,(\mathbf{X},\mathbf{r}))=\frac{1}{m}\sum_{i=1}^m\psi(r_i(1-\langle X_i,v\rangle)).$$

Let $\delta>0$. Then with probability greater than $1-\delta$ we have for all $v\in V$

$$err(v)\leq\hat{err}_\psi(v,(\mathbf{X},\mathbf{r}))+\frac{2L\,\|V\|}{m}\left(\sum_{i=1}^m\|X_i\|^2\right)^{1/2}+\sqrt{\frac{9\ln(2/\delta)}{2m}}.$$

If $\|X\|\leq 1$ a.s. then with probability greater than $1-\delta$ we have for all $v\in V$ that

$$err(v)\leq\hat{err}_\psi(v,(\mathbf{X},\mathbf{r}))+\frac{2L\,\|V\|}{\sqrt{m}}+\sqrt{\frac{\ln(1/\delta)}{2m}}.$$

Proof If we can prove the Theorem for $\psi:\mathbb{R}\to[0,1]$, then it will follow for general ψ, because it will also be true for $\min\{\psi,1\}$ which has Lipschitz constant bounded by L and $\hat{\text{er}}_{\min\{\psi,1\}}(v,(\mathbf{X},\mathbf{r}))\leq\hat{\text{er}}_\psi(v,(\mathbf{X},\mathbf{r}))$. We can thus assume $\psi:\mathbb{R}\to[0,1]$ and since $\psi\geq 1_{(-\infty,0]}$ we have $R(v)=\mathbb{E}\left[1_{(-\infty,0]}(r(1-\langle X,v\rangle))\right]\leq\mathbb{E}[\psi(r(1-\langle X,v\rangle))]$. In view of Theorem 5 it therefore suffices to prove that

$$\hat{\mathcal{R}}_m(\psi\circ\mathcal{F})(\mathbf{X})\leq\frac{2L\,\|V\|}{m}\left(\sum_{i=1}^m\|X_i\|^2\right)^{1/2},\tag{2}$$

where $\mathcal{F}$ is the function class

$$\mathcal{F}=\{(x,r)\in H\times\{-1,1\}\mapsto r(1-\langle x,v\rangle):v\in V\}.$$

By Theorem 6 we have $\hat{\mathcal{R}}_m(\psi\circ\mathcal{F})\leq L\,\hat{\mathcal{R}}_m(\mathcal{F})$ and one verifies easily that $\hat{\mathcal{R}}_m(\mathcal{F})=\hat{\mathcal{R}}_m(x\mapsto\langle x,v\rangle:v\in V)$, so (2) follows from Lemma 7. ■

To derive our bounds for hyperbolic PCA in Section 4 we need the following lemma. A similar statement can be found in Shawe-Taylor and Christianini (2003).

Lemma 9 *Let $V,W\subset H$ be and suppose that $X_1,\ldots,X_m$ are independent, identically distributed, zero-mean random variables with values in W. Then for ε and m such that $\|W\|\,\|V\|<\sqrt{m}\varepsilon$ we have*

$$\Pr\left\{\sup_{v\in V}\left|\frac{1}{m}\sum_{i=1}^m\langle v,X_i\rangle\right|>\varepsilon\right\}\leq\exp\left(\frac{-\left(\sqrt{m}\varepsilon-\|V\|\,\|W\|\right)^2}{2\left|\langle V,W\rangle\right|^2}\right).$$

Proof Consider the average $\bar{\mathbf{X}}=(1/m)\sum_1^m X_i$. With Jensen's inequality and using independence we obtain

$$\left(\mathbb{E}\left[\left\|\bar{\mathbf{X}}\right\|\right]\right)^2\leq\mathbb{E}\left[\left\|\bar{\mathbf{X}}\right\|^2\right]=\frac{1}{m^2}\sum_{i=1}^m\mathbb{E}\left[\|X_i\|^2\right]\leq\|W\|^2/m.$$

Now let $f : W^m \to \mathbb{R}$ be defined by $f(\mathbf{x}) = \sup_{v \in V} |(1/m) \sum_1^m \langle v, x_i \rangle|$. We have to bound the probability that $f > \varepsilon$. By Schwartz' inequality and the above bound we have

$$\mathbb{E}[f(\mathbf{X})] = \mathbb{E}\left[\sup_{v \in V} \left|\left\langle v, \bar{\mathbf{X}} \right\rangle\right|\right] \leq \|V\| \, \mathbb{E}\left[\left\|\bar{\mathbf{X}}\right\|\right] \leq \left(1/\sqrt{m}\right) \|V\| \, \|W\|. \tag{3}$$

Let $\mathbf{x} \in W^m$ be arbitrary and $\mathbf{x}' \in W^m$ be obtained by modifying a coordinate x_k of $\mathbf{x}$ to be an arbitrary $x'_k \in W$. Then

$$\left|f(\mathbf{x}) - f\left(\mathbf{x}'\right)\right| \leq \frac{1}{m} \sup_{v \in V} \left|\langle v, x_k \rangle - \left\langle v, x'_k \right\rangle\right| \leq \frac{2}{m} \left|\langle V, W \rangle\right|.$$

By (3) and the bounded-difference inequality (see McDiarmid, 1998) we obtain for $t > 0$

$$\Pr\left\{ f(\mathbf{X}) > \frac{\|V\| \, \|W\|}{\sqrt{m}} + t \right\} \leq \Pr\{ f(\mathbf{X}) - \mathbb{E}[f(\mathbf{X})] > t \} \leq \exp\left(\frac{-mt^2}{2\left|\langle V, W \rangle\right|^2} \right).$$

The conclusion follows from setting $t = \varepsilon - (1/\sqrt{m}) \|V\| \, \|W\|$ ■

We will also use the following model-selection lemma taken from (Anthony and Bartlett, 1999, Lemma 15.5):

Lemma 10 *Suppose*

$$\{F(\alpha_1, \alpha_2, \delta) : 0 < \alpha_1, \alpha_2, \delta \leq 1\}$$

is a set of events such that:

(i) For all $0 < \alpha \leq 1$ *and* $0 < \delta \leq 1$,

$$\Pr\{F(\alpha, \alpha, \delta)\} \leq \delta.$$

(ii) For all $0 < \alpha_1 \leq \alpha \leq \alpha_2 \leq 1$ *and* $0 < \delta_1 \leq \delta \leq 1$,

$$F(\alpha_1, \alpha_2, \delta_1) \subseteq F(\alpha, \alpha, \delta).$$

Then for $0 < a, \delta < 1$,

$$\Pr\left(\bigcup_{\alpha \in (0,1]} F(\alpha a, \alpha, \delta\alpha(1-a)) \right) \leq \delta.$$

3. Operator Valued Large Margin Classifiers

In this section we derive our algorithm. We give a proof of Theorem 1, then discuss the derivation of an objective functional and finally some issues related to its minimization.

3.1 Generalization Bounds

We first give a version of Theorem 1 for a fixed hypothesis space given by Hilbert-Schmidt operators of uniformly bounded norm and then derive from it a regularized version applying to all Hilbert-Schmidt operators.

Recall that the pair oracle is a probability measure ρ on $\mathcal{X}^2 \times \{-1,1\}$, the set of labeled pairs (sometimes also called equivalence constraints; Bar-Hillel et al., 2005), and that we assumed the input space $\mathcal{X}$ to be embedded in the Hilbert space H such that $diam(\mathcal{X}) \leq 1$. We will apply Theorem 8 to the Hilbert space $\mathcal{L}_2(H)$ instead of H and replace the random variable (X,r) of Theorem 8 by the random variable $(Q_{x-x'},r)$ with values in $\mathcal{L}_2(H) \times \{-1,1\}$, where (x,x',r) are distributed according to the pair oracle ρ. The training sample $S = ((x_1,x_1',r_1),...,(x_m,x_m',r_m))$ corresponds to m independent realizations $\left(\left(Q_{x_1-x_1'},r_1\right),...,\left(Q_{x_m-x_m'},r_m\right)\right)$ of this random variable. Since $diam(\mathcal{X}) \leq 1$ we have $\|Q_{x-x'}\|_2 = \|x-x'\|^2 \leq 1$ a.s. by virtue of Lemma 3 (i).

Recall the definition of the risk associated with a transformation T. Using Lemma 3 (iv) we have

$$\begin{aligned} R(T) &= \Pr\left\{r\left(1-\left\|Tx-Tx'\right\|^2\right) \leq 0\right\} = \Pr\left\{r\left(1-\langle Q_{x-x'}, T^*T\rangle_2\right) \leq 0\right\} \\ &= \operatorname{err}(T^*T). \end{aligned}$$

Here $\operatorname{err}(T^*T)$ (see Theorem 8) is the expected error of the linear classifier defined by the vector T^*T in $\mathcal{L}_2(H)$ thresholded at the value 1 and applied to a random labeled data point $(Q_{x-x'},r) \in \mathcal{L}_2(H) \times \{-1,1\}$. Also note that the empirical estimator (1) can be rewritten

$$\hat{R}_\psi(T,S) = \frac{1}{m}\sum_{i=1}^m \psi\left(r_i\left(1-\left\langle Q_{x_i-x_i'}, T^*T\right\rangle_2\right)\right).$$

With corresponding substitutions Theorem 8 becomes

Theorem 11 *Let $\psi : \mathbb{R} \to \mathbb{R}$, $\psi \geq 1_{(-\infty,0]}$ with Lipschitz constant L and $\mathcal{V} \subset \mathcal{L}_2(H)$. Let $\delta > 0$.*

(i) With probability greater than $1-\delta$ we have for all $T \in \mathcal{L}_\infty(H)$ such that $T^*T \in \mathcal{V}$

$$R(T) \leq \hat{R}_\psi(T,S) + \frac{2L\ \left\|\mathcal{V}\right\|_2}{m}\left(\sum_{i=1}^m \left\|X_i - X_i'\right\|^4\right)^{1/2} + \sqrt{\frac{9\ln(2/\delta)}{2m}}.$$

(ii) With probability greater than $1-\delta$ we have for all $T \in \mathcal{L}_\infty(H)$ such that $T^*T \in \mathcal{V}$

$$R(T) \leq \hat{R}_\psi(T,S) + \frac{2L\ \left\|\mathcal{V}\right\|_2}{\sqrt{m}} + \sqrt{\frac{\ln(1/\delta)}{2m}}.$$

Typically $\mathcal{V}$ would be a ball of some fixed radius, say c, about the origin in $\mathcal{L}_2(H)$, so that $\left\|\mathcal{V}\right\|_2 = c$. Our application of the vector valued generalization Theorem 8 introduced a certain looseness: Theorem 8 gives bounds on $\operatorname{err}(A)$ for all $A \in \mathcal{V}$, while we only require the bounds for those $A \in \mathcal{V}$ which are of the form $A = T^*T$, that is we are only using the linear functionals in $\mathcal{V} \cap \mathcal{L}_2^+(H)$. This raises the question if we might not get much better bounds for $\mathcal{V} \cap \mathcal{L}_2^+(H)$ than for $\mathcal{V}$. The following proposition shows that this will not work using Rademacher averages.

Proposition 12 *Let $\mathcal{F}$ and $\mathcal{F}^+$ be the function classes on $\mathcal{L}_2(H)$ given by*

$$\begin{aligned}\mathcal{F} &= \{B \in \mathcal{L}_2(H) \mapsto \langle B,A\rangle_2 : \|A\|_2 \le 1\},\\ \mathcal{F}^+ &= \left\{B \in \mathcal{L}_2(H) \mapsto \langle B,A\rangle_2 : \|A\|_2 \le 1,\ A \in \mathcal{L}_2^+(H)\right\}.\end{aligned}$$

Then $\hat{\mathcal{R}}_m(\mathcal{F}^+) \le \hat{\mathcal{R}}_m(\mathcal{F}) \le 4\hat{\mathcal{R}}_m(\mathcal{F}^+)$.

Proof The first inequality is obvious since $\mathcal{F}^+ \subset \mathcal{F}$ (see Theorem 12 in Bartlett and Mendelson, 2002). Suppose $\|A\|_2 \le 1$. With $A_1 = (A+A^*)/2$ and $A_2 = (A+A^*)/2$ we can write $A = A_1 + A_2$ with A_i symmetric and $\|A_i\|_2 \le 1$. By symmetry of the A_i and the spectral theorem we can write $A_i = A_{i1} - A_{i2}$ with $A_{ij} \in \mathcal{L}_2^+(H)$ and $\|A_{ij}\|_2 \le 1$. So any $f \in \mathcal{F}$ can be written in the form $f = f_{11} - f_{12} + f_{21} - f_{22}$ with $f_{ij} \in \mathcal{F}^+$, or $\mathcal{F} = \mathcal{F}^+ - \mathcal{F}^+ + \mathcal{F}^+ - \mathcal{F}^+$, whence the second inequality also follows from Theorem 12 in Bartlett and Mendelson (2002), or from an application of the triangle inequality. ■

By stratification over balls in $\mathcal{L}_2(H)$ we arrive at a generalization bound valid for *all* bounded operators. If specialized to $\mathcal{L}_0(H)$, the second conclusion below becomes Theorem 1 in the introduction.

Theorem 13 *Let $\psi : \mathbb{R} \to \mathbb{R}$, $\psi \ge 1_{(-\infty,0]}$ with Lipschitz constant L and $\delta > 0$.*

1. With probability greater than $1-\delta$ we have for all $T \in \mathcal{L}_\infty(H)$

$$R(T) \le \hat{R}_\psi(T,S) + \frac{4L\ \|T^*T\|_2}{m}\left(\sum_{i=1}^m \left\|X_i - X_i'\right\|^4\right)^{1/2} + \sqrt{\frac{9\ln(4\|T^*T\|_2/\delta)}{2m}}.$$

2. With probability greater than $1-\delta$ we have for all $T \in \mathcal{L}_\infty(H)$

$$R(T) \le \hat{R}_\psi(T,S) + \frac{4L\ \|T^*T\|_2}{\sqrt{m}} + \sqrt{\frac{\ln(2\|T^*T\|_2/\delta)}{2m}}.$$

Proof We will use Lemma 10. For $\alpha \in (0,1]$ let $\mathcal{V}(\alpha) = \{T \in \mathcal{L}_\infty(H) : \|T^*T\|_2 \le 1/\alpha\}$ and consider the events

$$F(\alpha_1,\alpha_2,\delta) = \Big\{\exists T \in \mathcal{V}(\alpha_2) \text{ such that } R(T) > \hat{R}(T,S) + \frac{2L}{\alpha_1 m}\left(\sum_{i=1}^m \left\|Q_{X_i - X_i'}\right\|^2\right)^{1/2} + \sqrt{\frac{9\ln(2/\delta)}{2m}}\Big\}.$$

By the first conclusion of Theorem 11 the events $F(\alpha_1,\alpha_2,\delta)$ satisfy hypothesis (i) of Lemma 10, and it is easy to see that (ii) also holds. If we set $a = 1/2$ and replace α by $1/\|T^*T\|_2$, then the conclusion of Lemma 10 becomes the first conclusion above. The second conclusion is proved similarly. ■

3.2 The Objective Functional

Because of the complicated estimates involved, risk bounds such as Theorem 13 have a tendency to be somewhat loose, so that a learning algorithm relying on naive minimization of the bounds may end up with suboptimal hypotheses. On the other hand in the absence of other helpful information such a bound provides a valuable guiding principle, as long as it is transformed to an objective functional which can be minimized in practice and allows for a flexible parametrization of the slack suspected in the bound.

Departing from the simpler of the two conclusions of Theorem 13, a naive approach would look for some $T \in \mathcal{L}_0(H)$ to minimize the objective functional

$$\hat{R}_\psi(T,S) + \frac{4L\ \|T^*T\|_2}{\sqrt{m}} + \sqrt{\frac{\ln\left(2\|T^*T\|_2/\delta\right)}{2m}}.$$

Our first modification is to discard the last term on the grounds that it will be dominated by second one. This is only justified if we exclude extremely small values of the confidence parameter δ and work with operators of reasonably large norm, so that $\|T^*T\|_2$ is substantially greater than $\sqrt{\ln\|T^*T\|_2/\delta}$. The new objective functional reads

$$\hat{R}_\psi(T,S) + \frac{4L\ \|T^*T\|_2}{\sqrt{m}}.$$

Our second modification is to replace the factor $4L$ in the second term by an adjustable regularization parameter $\lambda > 0$. On the one hand this just absorbs the Lipschitz constant L of the function ψ (which is yet to be fixed), on the other hand it expresses the belief that $\|T^*T\|_2/\sqrt{m}$ gives the right order of the true estimation error. We will stick to this belief, even though it can be successfully argued that it is naive, because the estimation error may decay much more rapidly than $m^{-1/2}$ as shown by Bartlett et al. (2005) and several other works. There were in fact experimental indications, that in our case the decay is not much better than $m^{-1/2}$, because the same value of λ appeared to work very well for different applications with rather different values of m (see Section 6.1).

The objective functional now depends on ψ and λ and has the form

$$\Lambda_{\psi,\lambda}(T,S) = \hat{R}_\psi(T,S) + \frac{\lambda\|T^*T\|_2}{\sqrt{m}}. \tag{4}$$

We still have to fix the Lipschitz function ψ, satisfying $\psi \geq 1_{(-\infty,0]}$. We want it to be as small as possible to reduce slack, but it should be convex for practical reasons. It is easy to show that any convex function ψ with $\psi \geq 1_{(-\infty,0]}$ and Lipschitz constant L satisfies $\psi \geq h_{L^{-1}}$, where h_γ is the hinge loss with margin $\gamma > 0$, defined by

$$h_\gamma(t) = \begin{cases} 1 - t/\gamma & \text{if} \quad t \leq \gamma \\ 0 & \text{if} \quad t > \gamma \end{cases}.$$

We settle for the hinge loss, not only because it is optimal with respect to convexity, the Lipschitz condition and the lower bound constraint, but because of its simplicity. Our final objective function thus depends on the two parameters λ and γ and reads

$$\Lambda_{h_\gamma,\lambda}(T,S) = \frac{1}{m}\sum_{i=1}^{m} h_\gamma\left(r_i\left(1 - \left\|Tx_i - Tx_i'\right\|^2\right)\right) + \frac{\lambda\|T^*T\|_2}{\sqrt{m}},$$

for a sample $S = ((x_1, x_1', r_1), ..., (x_m, x_m', r_m))$ generated in m independent, identical trials of the pair oracle ρ. The proposed algorithm searches for $T \in \mathcal{L}_\infty(H)$ to minimize $\Lambda_{h_\gamma,\lambda}(T,S)$.

While the hinge loss is used in most of our experiments, there are other choices. It is inherent to the problem of similarity learning that one is led to consider an asymmetric response to similar and dissimilar examples (see, for example, the approaches of Bar-Hillel et al., 2005 and Xing et al., 2002). This is implemented by making the Lipschitz function ψ dependent on the parameter r which indicates similarity or dissimilarity. We thus consider two functions $\psi_1, \psi_{-1} : \mathbb{R} \to \mathbb{R}$, $\psi_i \geq 1_{(-\infty,0]}$ and the objective functional

$$\Lambda_{\psi_1,\psi_{-1},\lambda}(T,S) = \frac{1}{m}\sum_{i=1}^{m} \psi_{r_i}\left(r_i\left(1 - \left\|T\left(x_i - x_i'\right)\right\|^2\right)\right) + \frac{\lambda\|T^*T\|_2}{\sqrt{m}}.$$

Note that our generalization bounds remain valid for the empirical risk estimate using ψ_1 and ψ_{-1} as long as we use for L the Lipschitz constant of $\min\{\psi_1, \psi_{-1}\}$. Using $\psi_i = h_{\gamma_i}$ we are effectively considering an *inside margin* γ_1, which applies to similar pairs, and an *outside margin* γ_{-1}, which applies to dissimilar pairs.

The use of different margins, or more generally different functions ψ_i in response to similar and dissimilar examples is nonsensical from the point of view of our bounds, which would always assume a smaller value if we used $\min\{\psi_1, \psi_{-1}\}$ to begin with. These bounds however were imported from the inherently symmetric vector valued case to a situation which is inherently asymmetric, because $\langle Q_{X-X'}, T^*T\rangle_2 = \|TX - TX'\|^2 \geq 0$ for all possible solutions T. The asymmetric margins may therefore have their merit and an instance of asymmetric margins is described in Section 4.

Even with symmetric margins the response will be asymmetrical: If we use $\gamma = 1$ (as in fact we did in our experiments) every similar pair (x_i, x_i') will make a contribution to the objective function unless $Tx_i = Tx_i'$, whereas dissimilar pairs will only contribute if $\|Tx_i - Tx_i'\|^2 < 2$, which can exclude many examples of dissimilarity, in particular if the regularization parameter λ is small.

Why not use the trace-norm $\|T^*T\|_1 = \|T\|_2^2$ as a regularizer? Since $\|T^*T\|_2 \leq \|T^*T\|_1$ the trace-norm could be substituted in our bounds and the regularization part of the algorithm in Table 1 below would simplify considerably if we used $\|T\|_2^2$ instead of $\|T^*T\|_2$. Regularization with the trace-norm is also believed to enforce sparsity in the sense of a low rank of T.

The trace-norm was not used for three reasons:

1. The bounds become looser upon substitution of $\|T^*T\|_1$. While this is obvious, one could argue, that there may be other bounds which work well with $\|T^*T\|_1$. Besides, the idea of minimizing bounds has to be approached with caution, so this argument is not decisive.

2. The trace-norm does not work as well in practice. In all experiments the trace norm was tried and performance found to be slightly inferior (while still comparable) to the use of the Hilbert-Schmidt norm $\|T^*T\|_2$ (see Section 6).

3. Regularization with the trace norm can cause too much sparsity and instability of the learning algorithm. This can be seen by simplifying the empirical part of the objective to be linear in $V = T^*T$ (that this is a valid approximation is shown in Proposition 15 below). Then there is an empirical operator A (made explicit in Section 4.2) such that an objective functional can be written as
$$-\langle V, A\rangle_2 + \lambda\|T^*T\|_p^p,$$

to be minimized with positive operators V. In this case the minimizers can be given explicitly in terms of A. For $p=2$ (the Hilbert-Schmidt case) one finds that the minimizer is a multiple of the positive part A_+ of the empirical operator A (the source of the sparsity observed in our experiments), and stable under small perturbations of the eigenvalues of A. For $p=1$ however the minimizer V will be a multiple of the projection to the subspace spanned by the eigenvectors corresponding to the largest positive eigenvalue of A. This space will be generically one-dimensional, making it useless for many data-representations. If it is more than one-dimensional then it will be unstable under perturbations of the eigenvalues of A.

3.3 Minimization of the Objective Functional

Throughout this section fix a sample $S=((x_1,x_1',r_1),...,(x_m,x_m',r_m))$ and assume that $\psi:\mathbb{R}\to\mathbb{R}$ is convex, satisfying $\psi\geq 1_{(-\infty,0]}$. For $\lambda>0$ consider the functional $\Omega_{\psi,\lambda}:\mathcal{L}_2^+(H)\to\mathbb{R}$

$$\Omega_{\psi,\lambda}(A)=\frac{1}{m}\sum_{i=1}^m\psi\left(r_i\left(1-\left\langle Q_{x_i-x_i'},A\right\rangle_2\right)\right)+\frac{\lambda\|A\|_2}{\sqrt{m}}. \tag{5}$$

Comparison with (4) reveals that $\Lambda_{\psi,\lambda}(T)=\Omega_{\psi,\lambda}(T^*T)$, so that any operator T is a minimizer of $\Lambda_{\psi,\lambda}$ if and only if T^*T is a minimizer of $\Omega_{\psi,\lambda}$ in $\mathcal{L}_2^+(H)$, a simple fact which has several important consequences. Note that $\Omega_{\psi,\lambda}$ is convex if ψ is convex, and since $\mathcal{L}_2^+(H)$ is a convex set we obtain a convex optimization problem. The situation somewhat resembles that of an SVM (in particular if ψ is the hinge loss), but solutions cannot be sought in all of $\mathcal{L}_2(H)$ but must lie in the cone $\mathcal{L}_2^+(H)$, a positivity constraint which makes the optimization problem quite different.

Denote with M the linear span of $\{x_i-x_i':i=1,...,m\}$ in H and define a map from M^m to $\mathcal{L}_2^+(H)$ by

$$A_{\mathbf{v}}=\sum_{i=1}^m Q_{v_i}\text{ where }\mathbf{v}=(v_1,...,v_m)\in M^m.$$

That $A_{\mathbf{v}}\in\mathcal{L}_2^+(H)$ follows from Lemma 3. We also define a linear transformation $T_{\mathbf{v}}:H\to\mathbb{R}^m$ by setting

$$(T_{\mathbf{v}}z)_k=\langle z,v_k\rangle\text{ for }k=1,...,m\text{ and }z\in H. \tag{6}$$

Then we have $T_{\mathbf{v}}^*T_{\mathbf{v}}=A_{\mathbf{v}}$. Also note that $\mathbf{v}\leftrightarrow T_{\mathbf{v}}$ establishes a continuous bijection between M^m and the set of all linear transformations $T:H\to\mathbb{R}^m$ with $M^\perp\subseteq Ker(T)$.

Suppose we can find some $\mathbf{v}\in M^m$ such that $\Omega_{\psi,\lambda}(A_{\mathbf{v}})\leq\Omega_{\psi,\lambda}(A)$ for all $A\in\mathcal{L}_2^+(H)$. Then $\Lambda_{\psi,\lambda}(T_{\mathbf{v}})=\Omega_{\psi,\lambda}(T_{\mathbf{v}}^*T_{\mathbf{v}})=\Omega_{\psi,\lambda}(A_{\mathbf{v}})$ is also optimal, so $T_{\mathbf{v}}$ will be an optimal pre-processor and we are done. To find such an optimal $\mathbf{v}\in M^m$ we plan to do gradient descent of $\Omega_{\psi,\lambda}(A_{\mathbf{v}})$ in the parameter $\mathbf{v}$ which automatically ensures the positivity constraint and keeps us comfortable in an at most m-dimensional environment. Since $\Omega_{\psi,\lambda}(A_{\mathbf{v}})$ is not generally convex in $\mathbf{v}$, even if $\Omega_{\psi,\lambda}$ is convex, one might worry about the existence of local minima.[1] The following theorem excludes this possibility:

Theorem 14 *Assume that ψ is convex. For $\mathbf{v}\in M^m$ define $\Phi(\mathbf{v})=\Omega_{\psi,\lambda}(A_{\mathbf{v}})$. If Φ has a stable local minimum at $\mathbf{v}\in M^m$ then $A_{\mathbf{v}}$ is a global minimizer of $\Omega_{\psi,\lambda}$ in $\mathcal{L}_2^+(H)$.*

1. Recall that a real function f on a topological space $\mathcal{X}$ has a local minimum at $x\in\mathcal{X}$ if there is an open set $O\ni x$ such that $f(x)\leq f(y)\ \ \forall y\in O$

Proof Abbreviate $\Omega_{\psi,\lambda}$ to Ω and use P to denote the orthogonal projection onto M. We first claim that

$$\forall A \in \mathcal{L}_2^+(H) \text{ we have } \Omega(PAP) \leq \Omega(A). \tag{7}$$

This follows from $\left\langle Q_{x_i-x_i'}, PAP \right\rangle_2 = \left\langle Q_{x_i-x_i'}, A \right\rangle_2$ and Lemma 3 (vi) by inspection of (5). Next consider the identity

$$\{A_{\mathbf{v}} : \mathbf{v} \in M^m\} = \left\{PAP : A \in \mathcal{L}_2^+(H)\right\}. \tag{8}$$

First note that the inclusion from left to right follows from $A_{\mathbf{v}} = PA_{\mathbf{v}}P$ for $\mathbf{v} \in M^m$. On the other hand for any $A \in \mathcal{L}_2^+(H)$ all the eigenvectors of PAP with nonzero eigenvalues have to lie in M, and enumerating the eigenvectors e_i of PAP beginning with those in M (of which there can be at most m) we have

$$PAPz = \sum_{i=1}^{m} \lambda_i \langle z, e_i \rangle e_i = \sum_{i=1}^{m} \left\langle z, \lambda_i^{1/2} e_i \right\rangle \lambda_i^{1/2} e_i = \sum_{i=1}^{m} Q_{\left(\lambda_i^{1/2} e_i\right)} z = A_{\mathbf{v}} z$$

for all $z \in H$, so that $PAP \in \{A_{\mathbf{v}} : \mathbf{v} \in M^m\}$ which proves (8).

To prove the theorem let Φ attain a local minimum at $\mathbf{v} \in M^m$. We will assume that Ω does not attain a global minimum at $A_{\mathbf{v}}$ and derive a contradiction. We can write $A_{\mathbf{v}} = T_{\mathbf{v}}^* T_{\mathbf{v}}$, using (6). Since Ω is convex it cannot even attain a local minimum at $A_{\mathbf{v}}$, so there is a sequence $A_n \in \mathcal{L}_2^+(H)$ such that $A_n \to A_{\mathbf{v}}$ and $\Omega(A_n) < \Omega(A_{\mathbf{v}})$. By continuity of multiplication also $PA_nP \to PA_{\mathbf{v}}P = A_{\mathbf{v}}$, by (7) $\Omega(PA_nP) \leq \Omega(A_n) < \Omega(A_{\mathbf{v}})$ and by (8) there exists $\mathbf{v}_n \in M^m$ such that $A_{\mathbf{v}_n} = PA_nP$. We thus have $A_{\mathbf{v}_n} \to A_{\mathbf{v}}$ (this does not imply that $\mathbf{v}_n \to \mathbf{v}$!) and $\Omega(A_{\mathbf{v}_n}) < \Omega(A_{\mathbf{v}})$. By continuity of the square-root $A_{\mathbf{v}_n}^{1/2} \to A_{\mathbf{v}}^{1/2} = |T_{\mathbf{v}}|$. By polar decomposition we can write $T_{\mathbf{v}} = U|T_{\mathbf{v}}|$, where U is a partial isometry, and define $T_n : M \to \mathbb{R}^m$ by $T_n = UA_n^{1/2}$. Then $T_n \to U_{\mathbf{v}}|T_{\mathbf{v}}| = T_{\mathbf{v}}$, so if $\mathbf{w}_n$ is chosen such that $T_n = T_{\mathbf{w}_n}$ we have $\mathbf{w}_n \to \mathbf{v}$, but also $\Phi(\mathbf{w}_n) = \Omega(T_n^* T_n) = \Omega\left(A_n^{1/2} U^* U A_n^{1/2}\right) = \Omega(A_n) < \Omega(A_{\mathbf{v}}) = \Phi(\mathbf{v})$, so Φ cannot attain a local minimum at $\mathbf{v}$. ■

Observe that this result justifies the gradient descent method in the presence of positivity constraints also for other convex loss functions besides the hinge loss. Nevertheless, it does not exclude the existence of points with vanishing gradients away from the global minimum. While the probability of arriving at these points during gradient descent is vanishing, the algorithm can still be slowed down considerably in their neighborhood, a possibility which we have to be prepared for (although it doesn't seem to happen in practice). The theorem therefore only proves that the gradient descent works, but not that it is efficient.

There is an alternative technique (see Xing et al., 2002) of iterative projections which avoids the problem of vanishing gradients and stays more closely to the original convex optimization problem. Briefly, one extends the functional Ω to all of $\mathcal{L}_2(H)$, so that the problem becomes equivalent to an SVM, and then alternates between gradient descent in Ω, which is convex, and projections onto $\mathcal{L}_2^+(H)$. The projection of an operator $A \in \mathcal{L}_2(H)$ to $\mathcal{L}_2^+(H)$ is effected by an eigen-decomposition and reconstruction with all the negative eigenvalues set to zero, so that only the positive part of A is retained. This method would also work for our objective functional, in fact for any convex objective constrained to positive operators. Here this technique was not chosen, because it appeared that the effort of the repeated eigen-decomposition might cancel the advantages of the method. Also the proposed gradient descent, which is easily converted to an online algorithm, appeared more elegant

Given sample S, regularization parameter λ, margin γ, learning rate θ
set $\lambda' = \lambda/\sqrt{|S|}$ and $d = m$
randomly initialize $v = (v_1, ..., v_d)$
repeat
  Compute $\|A_v\|_2 = \left(\sum_{ij} \langle v_i, v_j \rangle^2\right)^{1/2}$
  For $i = 1, ..., d$ compute $w_i = 2\|A_v\|_2^{-1} \sum_j \langle v_i, v_j \rangle v_i$
  Fetch (x, x', r) randomly from sample S
  For $i = 1, ..., d$ compute $a_i \leftarrow \langle v_i, x - x' \rangle$
  Compute $b \leftarrow \sum_{i=1}^d a_i^2$
  If $r(1-b) < \gamma$
    then for $i := 1, ..., d$ do $v_i \leftarrow v_i - \theta\left(\frac{r}{\gamma} a_i (x - x') + \lambda' w_i\right)$
    else for $i := 1, ..., d$ do $v_i \leftarrow v_i - \theta\lambda' w_i$
until convergence

Table 1: Learning algorithm

in its implicit adherence to the positivity constraint. The ultimate reason to stay with the proposed technique was of course its practical success.

So our algorithm will randomly initialize the vector $\mathbf{v} \in M^m$ and then follow the negative gradient of Φ, either for a specified number of steps or until some heuristic convergence criterion on the value of Φ is met. Straightforward differentiation gives for the k-th component of the gradient of Φ at $\mathbf{v} \in M^m$ the expression

$$\begin{aligned}(\nabla\Phi)_k(\mathbf{v}) &= \frac{-2}{m}\sum_{i=1}^m \psi'\left(r_i\left(1 - \sum_{j=1}^m a_{ij}^2\right)\right) r_i a_{ik}\left(x_i - x_i'\right) \\ &\quad + \frac{2\lambda}{\|A_{\mathbf{v}}\|_2 \sqrt{m}} \sum_{j=1}^m \langle v_k, v_j \rangle v_j,\end{aligned}$$

where $a_{ik} = \langle x_i - x_i', v_k \rangle$ and $\|A_{\mathbf{v}}\|_2 = \left(\sum_{i,j} \langle v_i, v_j \rangle^2\right)^{1/2}$. In Table 1 we give a corresponding algorithm of stochastic gradient descent for the case that ψ is the hinge-loss with margin γ.

In a simplified view, which disregards the regularization term (or if $\lambda = 0$), the algorithm will modify T in an attempt to bring Tx_i and Tx_i' closer if x_i and x_i' are similar (i.e., $r_i = 1$) and their distance exceeds $1 - \gamma$, it will attempt to move Tx_i and Tx_i' further apart if x_i and x_i' are dissimilar (i.e., $r_i = -1$) and their distance is less than $1 + \gamma$, and it will be indifferent to all other cases. This procedure can also be interpreted in terms of the effect which the individual gradient steps have on the level ellipsoid of the quadratic form induced by T^*T. Figure 1 tries to shows the simple geometrical intuition behind this construction.

There are two heuristic approximations to accelerate this algorithm. A simple time-saver is the observation that the contribution of the regularization term to the gradient changes only very little with small updates $\mathbf{v}$. It therefore doesn't need to be recomputed on every iteration, but it suffices to compute it intermittently.

Also in the experiments reported below a singular value decomposition of the optimal operator revealed that it was dimensionally sparse, in the sense that very few (generally less than 50) singular

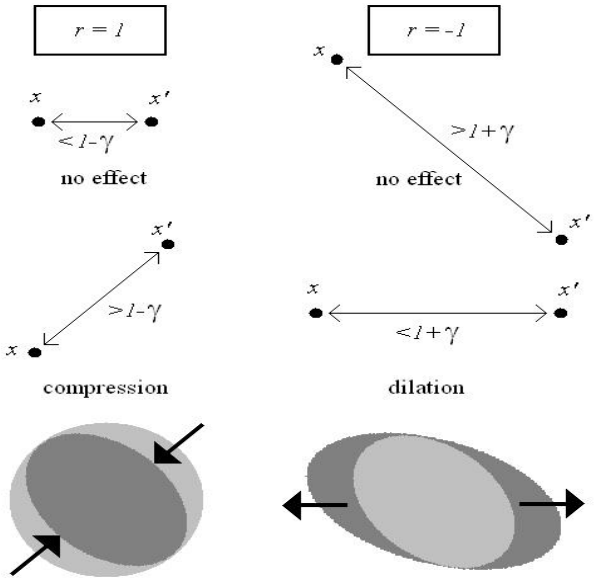


Figure 1: The effect of similar and dissimilar pairs on the level ellipsoid of the quadratic form induced by T^*T in the case of hinge loss with margin γ. If $\|Tx - Tx'\| \leq 1-\gamma$ for similar or $\|Tx - Tx'\| \geq 1+\gamma$ for dissimilar pairs there is no effect. If $\|Tx - Tx'\| < 1+\gamma$ for similar pairs, the ellipsoid is compressed in a direction parallel to the line between x and x'. If $\|Tx - Tx'\| > 1-\gamma$ for dissimilar pairs, the ellipsoid is dilated. Regularization corresponds to a shrinking of the ellipsoid.

values were significantly different from zero. This implies that $A_{\mathbf{v}}$ can be well approximated by some $A_{\mathbf{w}}$ where $\mathbf{w} \in M^d$ with $d \ll m$, and shows that the proposed algorithm effects a dimensional reduction. It also suggests that one might try gradient descent in M^d instead of M^m for $d < m$, which causes a considerable acceleration if $d \approx 10^2$ and $m \geq 10^3$. Of course the argument in Theorem 14 is then no longer valid, because finite dimensional constraints are not convex, so that local minima might become a problem. In practice this never happened for $d \approx 10^2$ and was observed only for $d \leq 5$. The heuristic is implemented by accordingly modifying the initialization $d = m$ in Table 1. In our experiments we used $d = 100$.

There is a simple practical use to an eigen-decomposition of the optimal $A_{\mathbf{v}}$ returned by the algorithm. The v_i will in general not be orthogonal, the algorithm does not enforce this in any way). If the decomposition reveals that only few eigenvalues of $A_{\mathbf{v}}$ are significantly different from zero, we can restrict ourselves to the span of the corresponding eigenvalues. This yields a representation operator T with very low dimensional range, which is easier to compute and facilitates subsequent processing.

By virtue of our dimension free bounds the algorithm is well suited for kernel implementations. Note that the search space consists of vectors $\mathbf{v} = (v_1, ..., v_d) \in M^d$ which admit a linear representation

$$v_i = \sum_{j=1}^{m} \alpha_j^i \left(x_j - x_j'\right)$$

in terms of the training sample. To add two such $\mathbf{v}$ we just add the corresponding matrices α_j^i, to compute inner products we just use the kernel function on the input space. Substituting these rules one finds that there is no problem in kernelization of the algorithm.

4. Similarity Regression and Hyperbolic PCA

In this section we consider some alternatives. The first is rather obvious and extends the method described above to continuous similarity values. The other method is derived from the risk functional R and has already been described in Maurer (2006a).

4.1 Similarity Regression

The bounds in Section 3.1 have straightforward extensions to the case, when the oracle measure is not supported on $\mathcal{X}^2 \times \{-1,1\}$ but on $\mathcal{X}^2 \times [0,1]$, corresponding to a continuum of similarity values, and $\ell : [0,1] \times \mathbb{R} \to \mathbb{R}$ is a loss function such that $\ell(y,.)$ has Lipschitz constant at most L for all $y \in [0,1]$. The corresponding risk functional to be minimized would be

$$R'(T) = \mathbb{E}_{(x,x',r)\sim\rho}\left[\ell\left(r, \left\|Tx - Tx'\right\|^2\right)\right].$$

If ℓ has the appropriate convexity properties, then an obvious modification of the proposed algorithm can be used. With a least-squares loss function the square of the norm (an unavoidable feature of our method) will lead to an overemphasis of large distances, probably an undesirable feature which can be partially compensated by a redefinition of the loss function at the expense of a large Lipschitz constant (e.g., with $\ell(y,t) = (y-t)^2/(y-y_0)^2$ for some $y_0 > 0$).

The possibilities of this type of similarity regression (or learning of metrics) remain to be explored.

4.2 Risk Bounds with Affine Loss Functions and Hyperbolic PCA

Consider again the case of a binary oracle (similar/dissimilar) and the task of selecting an operator from some set $\mathcal{V} \subset \mathcal{L}_2(H)$.

Proposition 15 *Suppose* $1 < \left\|\mathcal{V}\right\|_\infty = c < \infty$ *and set* $\eta_1 = -1$ *and* $\eta_{-1} = 1/\left(c^2-1\right)$. *Then for all* $T \in \mathcal{V}$

$$R(T) \leq 1 + \mathbb{E}_{(x,x',r)\sim\rho}\left[\eta_r\right] - \left\langle T^*T, \mathbb{E}_{(x,x',r)\sim\rho}\left[\eta_r Q_{x-x'}\right]\right\rangle_2.$$

For a balanced oracle this becomes

$$R(T) \leq \frac{c^2}{2\left(c^2-1\right)} - \left\langle T^*T, \mathbb{E}_{(x,x',r)\sim\rho}\left[\eta_r Q_{x-x'}\right]\right\rangle_2.$$

Proof Define real functions ψ_1, ψ_{-1} by $\psi_1(t) = 1-t$ and $\psi_{-1}(t) = 1 - t/\left(c^2-1\right)$. Since $\|x-x'\| \leq 1$ a.s. and by the definition of c we have for $T \in \mathcal{V}$

$$1_{(-\infty,0]}\left(r\left(1 - \left\|TX - TX'\right\|^2\right)\right) \leq \psi_r\left(r\left(1 - \left\|TX - TX'\right\|^2\right)\right) \text{ a.s.}$$

We also have for $r \in \{-1,1\}$ that $\psi_r(t) = 1 + r\eta_r t$, whence

$$\begin{aligned} R(T) &= \mathbb{E}_{(x,x',r)\sim\rho}\left[1_{(-\infty,0]}\left(r\left(1-\left\|TX-TX'\right\|^2\right)\right)\right] \\ &\leq \mathbb{E}_{(x,x',r)\sim\rho}\left[\psi_r\left(r\left(1-\left\|TX-TX'\right\|^2\right)\right)\right] \\ &= \mathbb{E}_{(x,x',r)\sim\rho}\left[1+\eta_r\left(1-\left\|TX-TX'\right\|^2\right)\right] \\ &= 1+\mathbb{E}_{(x,x',r)\sim\rho}\left[\eta_r\right] - \left\langle T^*T, \mathbb{E}_{(x,x',r)\sim\rho}\left[\eta_r Q_{x-x'}\right]\right\rangle_2. \end{aligned}$$

Since for a balanced oracle $1+\mathbb{E}_{(x,x',r)\sim\rho}[\eta_r] = c^2/\left(2\left(c^2-1\right)\right)$, the second conclusion is immediate. ■

Of course we can use the risk bounds of the previous section for an empirical estimator constructed from the Lipschitz functions ψ_1, ψ_{-1} used in the proof above, corresponding to an inner margin of 1 and an outer margin of c^2-1 (see Section 3.1 and 3.2). The affine nature of these functions however allows a different, more direct analysis and a different algorithmic implementation.

The operator

$$A := \mathbb{E}_{(x,x',r)\sim\rho}\left[\eta_r Q_{x-x'}\right]$$

is the expectation of the operator-valued random variable $(x,x',r) \mapsto \eta_r Q_{x-x'}$. Minimizing the bounds in Proposition 15 is equivalent to maximizing $\langle T^*T, A\rangle_2$, which is the only term depending on the operator T. Given the sample $S = ((x_1,x_1',r_1),\ldots,(x_m,x_m',r_m))$ the obvious way to try this is by maximizing the empirical counterpart $\left\langle T^*T, \hat{A}\right\rangle_2$ where $\hat{A}$ is the *empirical operator*

$$\hat{A}(S) = \frac{1}{m}\sum_{i=1}^{m} \eta_{r_i} Q_{x_i - x_i'}. \tag{9}$$

The next result addresses the issue of estimation. The result is similar to Theorem 13 but can be obtained without the use of Rademacher averages.

Theorem 16 *Under the above assumptions and if $c \geq 2$ we have for $\delta > 0$ with probability greater $1-\delta$ in the sample S that for all $T \in \mathcal{V}$*

$$\left|\left\langle T^*T, \hat{A}\right\rangle_2 - \langle T^*T, A\rangle_2\right| \leq \frac{2}{\sqrt{m}}\left(\sup_{T\in\mathcal{V}} \|T^*T\|_2 + c^2\sqrt{2\ln(1/\delta)}\right).$$

Proof Apply Lemma 9 to the $\mathcal{L}_2(H)$-valued random variable $\eta_r Q_{x-x'} - \mathbb{E}[\eta_r Q_{x-x'}]$. The corresponding substitutions give

$$\Pr\left\{\left|\left\langle T^*T, \hat{A}\right\rangle_2 - \langle T^*T, A\rangle_2\right| > \varepsilon\right\} \leq \exp\left(\frac{-\left(\sqrt{m}\varepsilon - 2\sup_{T\in\mathcal{V}}\|T^*T\|_2\right)^2}{8c^4}\right),$$

and the result follows from equating the right side to δ. ■

Of course an analogous result could have been obtained from Theorem 11.

We now specialize the space of candidate operators to $\mathcal{V} = c\mathcal{P}_d$ where $\mathcal{P}_d$ is the set of d-dimensional orthogonal projections. Then $\sup_{T\in\mathcal{V}} \|T^*T\|_2 = c^2\sqrt{d}$ in the bound above. The optimization problem now is to maximize $\left\langle P,\hat{A}\right\rangle_2$ for $P \in \mathcal{P}_d$ which is done by projecting onto a maximal eigenspace of $\hat{A}$, as shown by the following proposition.

Proposition 17 *Suppose $A \in \mathcal{L}_2(H)$ is symmetric with eigenvectors e_i and corresponding eigenvalues λ_i. Suppose that $d \in \mathbb{N}$ and that the sum in the eigen-expansion of A can be ordered in such a manner that $\lambda_i \geq \lambda_j$ for all $i \leq d < j$. Then*

$$\max_{P\in\mathcal{P}_d} \langle A,P\rangle_2 = \sum_{i=1}^{d} \lambda_i,$$

the maximum being attained by the projection onto the span of $(e_i)_{i=1}^d$.

Proof let $P \in \mathcal{P}_d$ with $v_1,...,v_d$ being an orthonormal basis for the range of P. Then

$$\begin{aligned}
\langle A,P\rangle_2 &= \sum_{j=1}^{d}\sum_{i=1}^{d} \lambda_i \left\langle v_j,e_i\right\rangle^2 + \sum_{j=1}^{d}\sum_{i=d+1}^{\infty} \lambda_i \left\langle v_j,e_i\right\rangle^2 \\
&\leq \sum_{i=1}^{d}\lambda_i \sum_{j=1}^{d} \left\langle v_j,e_i\right\rangle^2 + \lambda_d \sum_{j=1}^{d}\left(\sum_{i=d+1}^{\infty} \left\langle v_j,e_i\right\rangle^2\right) \\
&= \sum_{i=1}^{d}\lambda_i \sum_{j=1}^{d} \left\langle v_j,e_i\right\rangle^2 + \lambda_d \sum_{i=1}^{d}\left(1-\sum_{j=1}^{d} \left\langle v_j,e_i\right\rangle^2\right) \\
&\leq \sum_{i=1}^{d}\lambda_i \sum_{j=1}^{d} \left\langle v_j,e_i\right\rangle^2 + \sum_{i=1}^{d}\lambda_i\left(1-\sum_{j=1}^{d} \left\langle v_j,e_i\right\rangle^2\right) = \sum_{i=1}^{d}\lambda_i,
\end{aligned}$$

which proves $\sup_{P\in\mathcal{P}_d} \langle A,P\rangle_2 \leq \sum_{i=1}^{d}\lambda_i$ (this also follows directly from Horn's theorem; Simon, 1979, Theorem 1.15). If P is the projection onto the span of $(e_i)_{i=1}^d$ we can set $v_j = e_j$ above and obtain an equality. ■

This gives an alternative algorithm to the one described in Section 3: Fix a quantity $c > 2$ and construct a matrix representation of the empirical operator $\hat{A}$ given in (9). For some fixed target-dimension d find the projection onto a d-dimensional dominant eigenspace of $\hat{A}$. We omit the rather straightforward technical details concerning to the representation of $\hat{A}$ and the implementation of a kernel. Some of these issues are discussed in Maurer (2006a) where corresponding experiments are reported.

There is an intuitive interpretation to this algorithm, which could be called hyperbolic PCA: The empirical objective functional is proportional to

$$\begin{aligned}
m\left\langle P,\hat{A}\right\rangle_2 &= \sum_{i=1}^{m} \eta_{r_i} \left\|Px_i - Px_i'\right\|^2 \\
&= \frac{1}{c^2-1}\sum_{i:x_i,x_i' \text{ dissimilar}} \left\|Px_i - Px_i'\right\|^2 - \sum_{i:x_i,x_i' \text{ similar}} \left\|Px_i - Px_i'\right\|^2.
\end{aligned}$$

When similarity and dissimilarity are defined by class memberships, then maximizing this expression corresponds to maximizing inter-class- and minimizing intra-class variance, where the parameters $1/\left(c^2-1\right)$ and the proportions of similar and dissimilar pairs control the trade-off between these potentially conflicting goals. A similar proposal can be found in Thrun (1998). Typically we have $c \gg 1$ so that $1/\left(c^2-1\right) \ll 1$, so that dissimilar pairs ('negative equivalence constraints') receive a much smaller weight than similar pairs, corresponding to the intuitive counting argument given ba Bar-Hillel et al. (2005).

The method is similar to principal component analysis insofar as it projects to a principal eigenspace of a symmetric operator. In contrast to PCA, where the operator in question is the empirical covariance operator, which is always nonnegative, we will project to an eigenspace of an empirical operator which is a linear combination of empirical covariances and generally not positive. While the quadratic form associated with the covariance has elliptic level sets, the empirical operator induces hyperbolic level sets.

5. Applications to Multi-category Problems and Learning to Learn

In this section we apply similarity learning to problems where the nature of the application task is partially or completely unknown, so that the available data is used for a preparatory learning process, to facilitate future learning.

5.1 Classification Problems Involving a Large Number of Categories

A multi-category task τ with input space $\mathcal{X}$ is a pair $\tau = (\mathcal{Y}, \mu)$ where $\mathcal{Y}$ is a finite or countable alphabet of labels and μ a probability measure on $\mathcal{X} \times \mathcal{Y}$. We interpret $\mu(x,y)$ as the probability to encounter the pattern x with label y. As usual we assume $\mathcal{X}$ to be embedded in the Hilbert space H.

Let $\tau = (\mathcal{Y}, \mu)$ be such a task, $T \in \mathcal{L}_0(H)$ and suppose that we are given a *single* labeled example $(x,y) \in \mathcal{X} \times \mathcal{Y}$. Any classifier trained on this example alone and applied to another pattern $x' \in \mathcal{X}$ can sensibly only make the decisions "x' *is of type* y" or "x' *is not of type* y" or no decision at all. Face verification is a case where such classifiers can be quite important in practice: Anyone having to verify the identity of a person on the basis of a single photograph has to learn and generalize on the basis of a single example image. A simple classifier using only the pseudo-metric induced by T is the *elementary verifier* $\varepsilon_T(x,y)$ which decides

$$\begin{array}{cc} x' \textit{ is of type } y & \text{if} \quad \|T(x-x')\| < 1 \\ \textit{undecided} & \text{if} \quad \|T(x-x')\| = 1 \\ x' \textit{ is not of type } y & \text{if} \quad \|T(x-x')\| > 1 \end{array} .$$

Relative to the task $\tau = (\mathcal{Y}, \mu)$ it has the error probability (counting 'undecided' as an error)

$$\mathrm{err}_\tau(\varepsilon_T(x,y)) = \Pr_{(x',y') \sim \mu} \left\{ r\left(y,y'\right)\left(1 - \left\|T\left(x-x'\right)\right\|\right) \leq 0 \right\},$$

where the function $r : \mathcal{Y} \times \mathcal{Y} \to \{-1,1\}$ quantifies equality and inequality: $r(y,y') = 1$ if $y = y'$ and $r(y,y') = -1$ if $y \neq y'$.

There is a canonical pair oracle derived from the task τ. It is the probability measure ρ_τ on $\mathcal{X}^2 \times \{-1,1\}$ given by

$$\rho_\tau(A) = \Pr_{((x,y),(x',y')) \sim \mu^2} \left\{ \left(x, x', r\left(y,y'\right)\right) \in A \right\} \text{ for } A \subseteq \mathcal{X}^2 \times \{-1,1\}. \tag{10}$$

To generate a draw (x,x',r) from ρ_τ make two independent draws of (x,y) and (x',y') from μ and then return $(x,x',1)$ if $y=y'$ and $(x,x',-1)$ if $y\neq y'$. Then

$$\mathbb{E}_{(x,y)\sim\mu}\left[\mathrm{err}_\tau\left(\varepsilon_T\left(x,y\right)\right)\right]=R_{\rho_\tau}\left(T\right), \tag{11}$$

so we can use the risk bounds in Sections 3.1 or 4.2 to bound the expected error of the elementary verifier $\varepsilon_T\left(x,y\right)$ under a random draw of the training example (x,y).

If there are many labels appearing approximately equally likely, then dissimilar pairs will be sampled much more frequently than similar ones, resulting in a negative bias of elementary classifiers. Similar unwanted effects have been noted by Xing et al. (2002) and Bar-Hillel et al. (2005). This does not mean that our bounds are paradoxical, because the biased sampling corresponds to an equally biased error measure: Under these circumstances the non-verifier which always asserts "x' *is not of type* y" would already have a small error.

This problem can be avoided by a simple balancing technique: In the computation of the error of the elementary classifier $\varepsilon_d\left(x,y\right)$ we assign different weights to the cases of false rejection and false acceptance. Define a balanced error

$$\begin{aligned}\overline{\mathrm{err}}_\tau\left(\varepsilon_d\left(x,y\right)\right) &= \frac{1}{2C_1}\Pr_{(x',y')\sim\mu}\left\{\left\|T\left(x-x'\right)\right\|\geq 1 \text{ and } y'=y\right\}\\ &\quad+\frac{1}{2C_{-1}}\Pr_{(x',y')\sim\mu}\left\{\left\|T\left(x-x'\right)\right\|\leq 1 \text{ and } y'\neq y\right\},\end{aligned}$$

where $C_1=\mu^2\left\{\left(\left(x,y\right),\left(x',y'\right)\right):y'=y\right\}$ and $C_{-1}=\mu^2\left\{\left(\left(x,y\right),\left(x',y'\right)\right):y'=y\right\}$ are the probabilities to obtain examples with equal and unequal labels respectively in two independent draws of μ. If we define a balanced pair oracle $\overline{\rho}$ by

$$\overline{\rho}_\tau\left(A\right)=\frac{\rho_\tau\left(A\cap\mathcal{X}^2\times\{1\}\right)}{2\rho_\tau\left(\mathcal{X}^2\times\{1\}\right)}+\frac{\rho_\tau\left(A\cap\mathcal{X}^2\times\{-1\}\right)}{2\rho_\tau\left(\mathcal{X}^2\times\{-1\}\right)},$$

where ρ_τ is defined in (10) then one again verifies that

$$\mathbb{E}_{(x,y)\sim\mu}\left[\overline{\mathrm{err}}_\tau\left(\varepsilon_d\left(x,y\right)\right)\right]=R_{\overline{\rho}_\tau}\left(T\right) \tag{12}$$

and that $\overline{\rho}_\tau$ returns similar and dissimilar pairs with equal probability. To generate a draw of (x,x',r) from $\overline{\rho}_\tau$ first draw (x,y) from μ and then flip a fair coin. On heads draw x' from the class conditional distribution of y and return $(x,x',1)$, on tails draw x' from the conditional distribution for $\mathcal{Y}\setminus\{y\}$ (or continue to draw $(x',y')\sim\mu$ until $y'\neq y$) and return $(x,x',-1)$.

Whichever of the two oracles we use: If we make m independent draws from it to obtain a sample $S=\left(\left(x_1,x_1',r_1\right),\ldots,\left(x_m,x_m',r_m\right)\right)$ from which we generate the operator T according to the algorithm derived in Section 3, then we are essentially minimizing a bound on the expected error of elementary verifiers trained on future examples. In this way the proposed algorithm can be considered an algorithm of learning to learn.

This is particularly interesting if $m<|\mathcal{Y}|$, because we obtain performance guarantees for categories which we haven't seen before. Consider for example the case where $\mathcal{Y}$ stands for a large population of human individuals, and the inputs correspond to facial images. From a sample $S=\left(\left(x_1,x_1',r_1\right),\ldots,\left(x_m,x_m',r_m\right)\right)$ we train an operator T and then use the elementary verifiers ε_T for face-verification on the entire population. With methods similar to the balancing technique described above the oracle can be also modified to achieve different desired penalties for false rejection and false acceptance.

5.2 Similarity as a Vehicle for Transfer

The face-verification system proposed above is somewhat naive from an economical point of view, because the example-pairs have to be obtained independently, which means that we will need images of approximately $3m/2$ individuals (with balancing), all of which will probably have to be paid to cooperate. It would be preferable to collect the data independently but conditional to a smaller subpopulation $\mathcal{Y}' \subset \mathcal{Y}$ with $|\mathcal{Y}'| < m$, so that multiple images can be gathered from each individual. This effectively replaces the original task $\tau = (\mathcal{Y}, \mu)$ with a subtask $\tau' = (\mathcal{Y}', \mu')$, where

$$\mu'(A) = \frac{\mu(A)}{\mu(\mathcal{X} \times \mathcal{Y}')} \text{ for } A \subseteq \mathcal{X} \times \mathcal{Y}'.$$

Of course events which are independent w.r.t. μ are not independent w.r.t. μ' and vice versa, so if we sample independently from μ' our generalization guarantees will only work for the task τ'. The operator T generated from the sample drawn from μ' may nonetheless work for the original task τ corresponding to the entire population.

This points to a different method to apply the proposed algorithm: Use one task $\tau = (\mathcal{Y}, \mu)$, the training task, to draw the training sample and train the representation T, and apply T to the data of the application task $\tau' = (\mathcal{Y}', \mu')$.

As the application task is unknown at the time of training T, this transfer mechanism may of course fail. Deciding between success or failure however does not have the sample complexity of learning, which is affected by a complexity penalty of the function class, but only the sample complexity of validation, which can be determined from Hoeffdings inequality. If T has been trained from τ and is subsequently tested on $\tau' = (\mathcal{Y}', \mu')$ then with probability greater $1 - \delta$ in an i.i.d. sample S' of size m drawn from μ' we have

$$R_{\rho_{\tau'}}(T) \leq \hat{R}_{1_{(-\infty,0]}}(T, S') + \sqrt{\frac{\ln(1/\delta)}{2m}},$$

which is of course much better than the bounds in Theorem 11 (here $\hat{R}_{1_{(-\infty,0]}}(T, S')$ is the empirical risk of T on S').

Of course this type of transfer can be attempted between any two binary classification tasks, but its success normally requires a similarity of the pattern classes themselves. A classifier trained to distinguish images of the characters "a" and "b" will be successfully applied to two classes of images if these pattern classes have some resemblance of "a" and "b". A representation of similarity however can be successfully transferred if there is a similar notion of similarity applicable to both problems. It is this higher conceptual level of similarity concepts that allows them to transcend task-boundaries. This kind of 'meta-similarity" is for example present, when there is a common process of data generation, as is the case when the tasks share an invariance property.

Suppose now that the learner has trained operators $T_1, ..., T_K$ from the samples $S_1, ..., S_K$ gathered in past experience, with each S_k drawn iid from a corresponding multi-category task τ_k, and that the learner is currently confronted with a new task τ', for which a new sample S' is available. A simple union bound yields from Hoeffdings inequality the following standard result: $\forall \delta > 0$ with probability greater $1 - \delta$ in S' we have that for every $k \in \{1, ..., K\}$

$$R_{\rho_{\tau'}}(T_k) \leq \hat{R}_{1_{(-\infty,0]}}(T_k, S') + \sqrt{\frac{\ln K + \ln(1/\delta)}{2m}}.$$

Let us assume that K is not too large. Minimizing $\hat{R}_{1_{(-\infty,0]}}(T_k, S')$ over k is a simple algorithm which does two things:

1. It finds the operator T_{k^*} which optimally represents the data of τ' for the purpose of recognition on the basis of single training examples.

2. It classifies the new task τ into one of the K meta-categories defined by the old tasks $\tau_1, ..., \tau_K$. The classification is essentially carried out on the basis of compatibility of underlying notions of similarity.

The more obvious practical aspect is the first, because T_{k^*} can be put to use right away. There are however interesting strategies for "life long learning" (Thrun, 1998) where the mechanism of task classification is also useful. If the optimal empirical risk $\hat{R}_{1_{(-\infty,0]}}(T_k, S')$ is too large one could use S' to train T' which is appended to $(T_1, ..., T_K)$ to reflect the fact that a new type of task has been discovered. If $\hat{R}_{1_{(-\infty,0]}}(T_k, S')$ is small on the other hand, one could merge the data S' of the new task with the data S_{k^*} of the most closely matching task and retrain on $S' \cup S_{k^*}$ to obtain an operator T'' to replace T_{k^*} for better generalization due to the larger sample size, and keep the number K constant.

In this context it should be noted, that "multi-task learning", the idea of pooling the data from various tasks to achieve a smaller estimation error (see Caruana 1998; Baxter, 2000; Evgeniou et al., 2004; Maurer 2006b), is easily implemented in the case of similarity learning by just concatenating the samples $S_1 \cup ... \cup S_K$ and training. The concatenated sample corresponds to a draw from the mixed pair oracle $\rho = \sum c_k \rho_k$ with $c_k = |S_k| / \sum_i |S_i|$.

The proposed method has been derived from the objective of minimizing the risk functional R which is connected to classification through the identities (11) and (12). It is therefore a principled technique to train representations for future learning on the basis of a *single example*. While the representation T can be used to preprocess data for other algorithms operating on larger future training examples, there is reason to believe that it will be no longer optimal if there are more examples. It is a challenging problem to define risk functionals giving optimality for algorithms operating on other future sample sizes. While the algorithm proposed in Argyriou et al. (2006) appears to have such properties, corresponding risk bounds are lacking and the relationship to the current work remains to be explored.

6. Experiments

All the experiments reported concern transfer in machine vision where a representation trained on one task is applied to another one. The experiments involved the recognition of randomly rotated-, randomly scaled-, randomly rotated and scaled-, and handwritten characters, spatially rotated objects and face recognition. Below we briefly describe the parametrization of the training algorithm, the various tasks tried and the parameters recorded in testing. An executable to reproduce most of the experiments will be made available at the web-site www.andreas-maurer.eu.

6.1 Parametrization and Experimental Setup

All the experiments with character recognition used gray-scale images of 28×28 pixels, corresponding to 784-dimensional vectors. The images of the COIL100 database had 64×64, the images of the ATT face database had 92×112 pixels. Pixel vectors were normalized to unity, otherwise

there was no preprocessing. The embedding in the Hilbert space H was effected by the Gaussian RBF-kernel

$$\langle x,y\rangle = \kappa(x,y) = (1/2)\exp\left(-4\left|\frac{x}{|x|}-\frac{y}{|y|}\right|^2\right),$$

where x and y are two images, $|.|$ is the euclidean norm on pixel vectors and $\langle .,.\rangle$ is the inner product of the embedded vectors in the RKHS H (see, for example, Christianini and Shawe-Taylor, 2000, for kernel techniques).

For all tasks involved in the experiments corresponding data-sets were generated and processed in this kernel-representation.

In every transfer experiment there was a training task and an application- or test task, represented by corresponding labeled data-sets. On the data-set of the training task the algorithm given in Table 1 was used, together with the accelerating heuristics in Section 3.3, to generate a representation T. All the experiments reported below were carried out with margin $\gamma = 1$ and the regularization parameter either $\lambda = 0.005$ (for the Hilbert-Schmidt norm $\|T^*T\|_2$) or $\lambda = 0.001$ (for $\|T^*T\|_1 = \|T\|_2^2$). These values were determined by cross validation for problem of handwritten character recognition below and reused in all the other experiments. Separately optimizing parameters for each experiment using cross validation would only lead to an improvement of the results.

The gradient descent was carried out for 10^6 steps with a constant learning rate $\theta = 0.01$. For the training task we report the final values of the objective function Λ and the empirical risk $\hat{R}(T) = \hat{R}_{1_{(-\infty,0]}}(T)$. Another interesting property of T is its 'essential rank' as the number of singular values appreciably larger than 0. In the results below this is referred to as 'sparsity' and given as the number of singular values larger than 2% of the spectral norm.

The representation T is applied to the data-set of the application task, with pixel vectors equi-dimensional to those in the training data-set. Application and training data-sets had no overlapping categories.

On the application task we measured three properties related to the quality of the representation:

1. The empirical risk $\hat{R}(T) = \hat{R}_{1_{(-\infty,0]}}(T)$ as an estimator for the true risk $R(T)$. This relates to the theory above and to the performance of elementary verifiers.

2. The area under the ROC-curve (ROC area T) for the distance as a detector of class-equality. This can be regarded as an estimator for the probability that a pattern pair with equal labels is represented at a closer distance than an independently chosen pair with different labels.

3. The error (error T) of nearest neighbor classifiers when each category of the application task is represented by a *single* example, averaged over 100 runs with randomly chosen examples.

Both of the latter two quantities were also measured for the unrepresented but normalized input pixel vectors (ROC area input, error input). The values obtained using the trace-norm regularization are given in parentheses.

It has been pointed out by a referee that the theory is not exactly applicable to the experiments, because training sets and the test sets are not chosen iid from the same distribution. The experiments were carried out in the present form to illustrate the utility of similarity for transfer. Separating the same data-sets into sets of training- and test pairs would certainly have produced even better results.

6.2 Training and Application Tasks

In some of the transfer experiments the training and application tasks shared a definitive class of invariants, so that similarity of two pattern corresponds the existence of a transformation in the class of invariants, which (roughly) maps one pattern to the other.

Rotation invariant character recognition. Randomly rotated images of printed alpha characters were used for the training set and randomly rotated images of printed digits were used for the test set, the digit 9 being omitted for obvious reasons.

Scale invariant character recognition. Randomly scaled images of printed alpha characters for the training, randomly scaled images of printed digits were used for the test set. Scaling ranged over a factor of 2.

Rotation and scale invariant character recognition. Randomly rotated and scaled characters for training, randomly rotated and scaled images of printed digits were used for the test set, the digit 9 being omitted for obvious reasons. Scaling ranged over a factor of 2, the digit 9 is omitted in the test set.

Spatially rotation invariant object recognition. The COIL100 database contains images of objects rotated about an axis at an angle 60° to the optical axis. Here the invariance transformations which relate similar patterns cannot be explicitly computed from the images. The first 80 objects of the database were taken for training, the remaining 20 for testing.

In the remaining experiments the underlying notion of similarity cannot be explicitly specified and corresponds to a Gestalt-property of the domain in question.

Handwritten character recognition. The images of handwritten alpha characters from the NIST database were used for training, the handwritten digits from the MNIST database for testing.

Face recognition. The first 35 images of the ATT database for training, the last 5 for testing. Unfortunately the ATT database is at the same time very small and very clean and easy, so that the results are not very conclusive. Attempts to obtain the potentially more interesting Purdue database failed.

The images for the first three experiments are available on the web-site www.andreas-maurer.eu, the others are publicly available.

6.3 Results for Transfer

The results are summarized in Tables 2 and 3. The various row headings will be explained in the sequel.

In all experiments the representation T brings an improvement in recognition rates. This improvement is moderate (54% error downto 33% in the case of handwritten characters) to spectacular (72% downto $< 1\%$ for plane rotations). The results on the COIL and ATT databases slightly improve on the corresponding results in Fleuret and Blanchard, (2005) and Chopra et al. (2005), but the margin is so small, that this may well be a statistical artifact. What is more remarkable is that our results are at all comparable, because our method makes no assumption on the specific properties of image data, such as high correlations for neighboring pixels: In contrast to the approaches described in Chopra et al. (2005) and Fleuret and Blanchard (2005), our method would yield the same results if the images were subjected to any fixed but unknown permutation of pixel indices.

type of experiment	rotation invariance	scale invariance	rot.+scale invariance
training set	alpha	alpha	alpha
nr of categories	20	52	20
nr of examples	2000	1560	4000
$\hat{R}(T)$	0.019	0.005	0.058
$\Lambda(T)$	0.074	0.033	0.185
sparsity of T	9	42	7
test set	digits $\setminus$ 9	digits	digits $\setminus$ 9
nr of categories	9	10	9
nr of examples	900	300	1800
$\hat{R}(T)$	0.55	0.061	0.128
ROC area input	0.597	0.69	0.54
ROC area T	0.999 (0.999)	0.995 (0.99)	0.982 (0.972)
error input	0.716	0.508	0.822
error T	0.009 (0.011)	0.019 (0.035)	0.097 (0.127)

Table 2:

type of experiment	spatial rot. invariance	handw. Chars	face recognition
training set	COIL $\leq$80	NIST	ATT 1-35
nr of categories	80	52	35
nr of examples	2880	4160	350
$\hat{R}(T)$	0.003	0.038	0
$\Lambda(T)$	0.024	0.314	0.022
sparsity of T	46	19	35
test set	COIL $\geq$81	MNIST	ATT 36-40
nr of categories	20	10	5
nr of examples	720	10000	50
$\hat{R}(T)$	0.379	0.183	0.045
ROC area input	0.845	0.728	0.934
ROC area T	0.989 (0.984)	0.9 (0.891)	0.997 (0.997)
error input	0.375	0.549	0.113
error T	0.093 (0.123)	0.335 (0.383)	0 (0)

Table 3:

6.4 One Experiment in Detail

To illustrate these experiments and results we will consider the example of combined rotation and scale-invariance, corresponding to the last column in Table 2. Figure 2 shows some typical training examples, of which there are 200 representing each of the 20 categories, making a total number of 4000.

The oracle presents the learner with pairs of these images together with a similarity value $r \in \{-1,1\}$. Similar pairs are chosen from the same category (the same column in Fig. 2) dis-

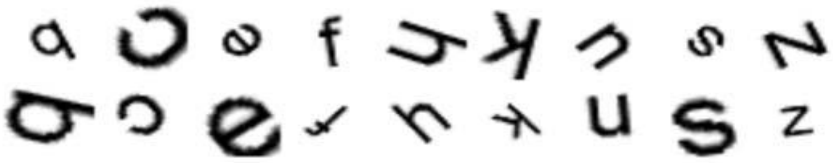


Figure 2: Randomly rotated and scaled alpha-characters

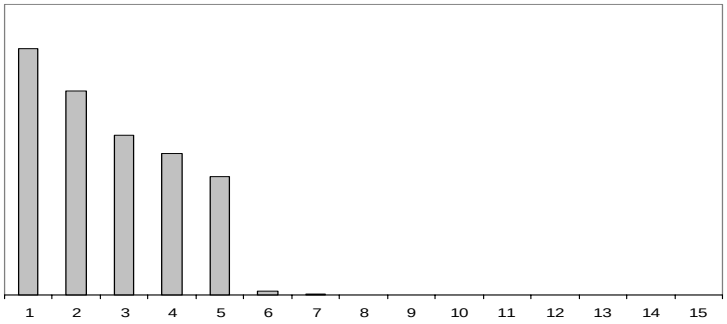


Figure 3: The 15 largest eigenvalues of T^*T in proportion

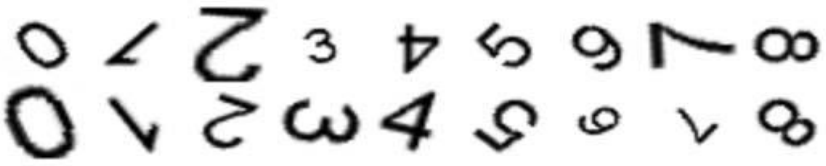


Figure 4: Randomly rotated and scaled digits

similar ones from different categories (different columns). The pairs are chosen at random under the constraint that similar pairs appear with equal frequency as dissimilar ones. These pairs are fed as input to the stochastic gradient descent algorithm in Table 1. While there are $O\left(10^6\right)$ pairs potentially generated, only about 2000 of these can be statistically independent in terms of the generation of the original sample.

The spectrum of T^*T (Fig. 3) of the resulting operator T shows a marked decrease of singular values, allowing the conclusion that the data characterizing rotation and scale invariant character categories in the chosen Gaussian kernel representation is essentially only 5-dimensional. This observed sparsity is not a consequence of the regularization with the Hilbert-Schmidt norm, because an increase in the regularization parameter λ increases the essential rank of T (a different behavior would be expected with a 1-norm regularizer), but an intrinsic property of the data which the algorithm discovers.

The representation T is then applied to the recognition of digits. Fig. 4 exemplifies the test data.

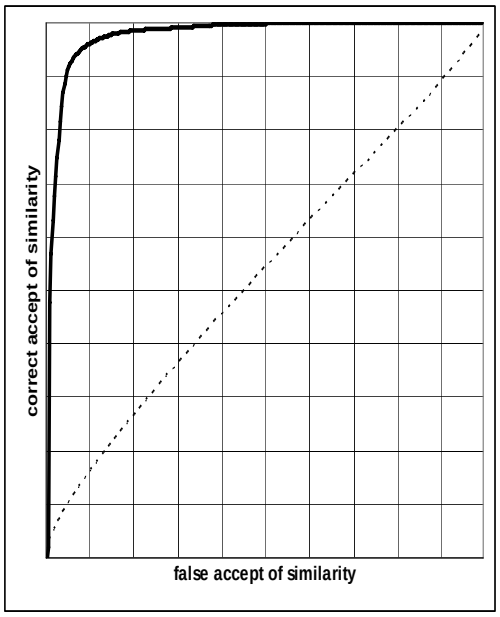


Figure 5: ROC curve for the metric as a feature for similarity. Transformed data solid, input data dotted.

To measure the empirical risk $\hat{R}(T) = \hat{R}_{1_{(-\infty,0]}}(T)$ we generate a random sequence of pairs as we did with the training task above and average the relative rates of dissimilar pairs with $\|Tx - Tx'\|_H \leq 1$ and the rate of similar pairs with $\|Tx - Tx'\|_H \geq 1$. Here the subscript H refers to the distance in the RKHS. This gives the entry 0.128 in the row labeled by 'Risk T', so the representation T organizes the data of rotation and scale invariant character categories into balls of diameter 1, up to an error of about 13%.

To parametrize a potential verification system an ROC curve for the utility of the metric as a feature for class equality is useful. Fig 5 shows these curves dotted for the metric $\|x - x'\|_{28\times 28}$ (where the distance is measured on the raw normalized pixel vectors in $\mathbb{R}^{28\times 28}$) in red and solid for $\|Tx - Tx'\|_H$. The areas under these curves correspond to the rows labeled 'ROC area input' and 'ROC area T' respectively. The values in parentheses correspond to regularization with $\|T\|_2^2$, which givel slightly inferior results.

Finally we measure the performance of a single-nearest neighbor classifier. A prototype is selected randomly from each category. The images in either one of the rows in Fig. 4 could represent such a 'training sample'. We then measure the performance of the corresponding 1-NN classifier on the test set with the training data omitted. The error rates are averaged over 100 random selections of the prototype set. These computations are carried out (with equal prototypes) for the metric $\|x - x'\|_{28\times 28}$ and the metric $\|Tx - Tx'\|_H$ and give the entries in the rows 'error input' and 'error T' respectively. Again the values in parentheses correspond to regularization with $\|T\|_2^2$.

6.5 Classification of Tasks

We report some simple results concerning the recognition of task-families on the basis of the similarity of underlying similarity concepts, as proposed in Section 5.2. In Table 4 we consider task-

families with 28×28 pixel data. These families share the properties of rotation invariance, scale invariance, combined rotation and scale invariance and the property of being handwritten respectively. The columns correspond to alpha-characters used to train representation, the rows to digits used for testing. Each entry is the empirical risk $\hat{R}(T)$ of the operator T trained from the alpha-task heading the column, measured on the digit-task heading the row. The minimum in each row occurs

	rotation	scaling	rot.+scaling	handwritten
rotation	0.055	0.38	0.089	0.374
scaling	0.36	0.061	0.11	0.304
rot.+scaling	0.375	0.39	0.128	0.434
handwritten	0.4	0.336	0.35	0.18

Table 4:

on the diagonal, which shows that the underlying similarity property or invariance of a data-set is reliably recognized. The margin of this minimum is weakened only for separate rotation and scale invariances, because the representation for combined rotation and scale invariance also performs reasonably well on these data-sets, although not as well as the specialized representations. Scale invariant representations perform better on handwritten data than rotation invariant ones, probably because scale invariance is more related to the latent invariance properties of handwritten characters than full rotation invariance.

Given the nature of the algorithm, these results are perhaps not surprising, but they seem to point in interesting new directions for the design of more autonomous systems of pattern recognition.

7. Related Work

A lot of work has been done to develop learning algorithms for data-representations with optimal metric properties. Typically a heuristically derived objective function is optimized, and often there is no discussion of generalization performance for high dimensional input data.

The classical method seems to be *Linear Discriminant Analysis* (LDA, Fukunaga, 1990) which projects onto a dominant subspace of the matrix quotient of the inter-class and intra-class empirical covariances. This can only work if the intra-class covariance operator is non-singular, and it will work poorly even if it is non-singular, but has very small eigenvalues (a generic situation in high dimensions), whence there have been several efforts to remedy the situation by optimization within the null-space of the intra-class covariance operator (NLDA) or by simultaneous diagonalisation of the intra- and inter-class covariances (OLDA). The stability problem inherent to the quotient approach is approached by a heuristic regularization prescription (ROLDA). These various extensions to LDA are presented by Ye and Xiong (2006) and have been tested in an experimental situation where the trained representation is combined with K-NN classification on the same task where the representation was trained. The performance appears to be comparable to SVM.

LDA and its extensions can be best compared to the algorithm of hyperbolic PCA presented in Section 4. In contrast to LDA hyperbolic PCA projects onto a dominant eigenspace of a weighted difference and not the quotient of the inter- and intra-class covariances. For this reason hyperbolic PCA is free of the stability problems of LDA and generalization guarantees are easily obtained.

An interesting technique has been proposed by Goldberger et al (2004). The desired representation is chosen to optimize the performance of a stochastic variant of K-NN classification on the

represented data. The method, called *Neighborhood Component Analysis* (NCA) appears to admit a regularized version, with essentially the same regularizer as in this work, and it seems possible to obtain dimension-free generalization guarantees for the regularized version. Unfortunately the optimization problem underlying NCA is not convex.

There is a certain kinship of NCA to the technique presented in this work, because both approaches depart from an objective defined by performance requirements of algorithms operating on the represented data. The stochastic K-NN classifiers of NCA correspond to the elementary verifiers (see Section 5) in our approach.

The problem of similarity learning from pair oracles similar to this paper has been considered by several authors.

In the work of Bar-Hillel et al. (2005) the triplets (x,x',r) generated by the oracle are called equivalence constraints, positive if $r=1$ and negative if $r=-1$. Their algorithm, called RCA for *Relevant Component Analysis*, does not use negative equivalence constraints on the grounds that these are less informative than positive ones, a claim supported by a simple counting argument. The objective of RCA is essentially entropy maximization under the constraint that "chunklets" of data points belonging to the same class, as inferred from the positive equivalence constraints, remain confined to balls of a fixed diameter. Under Gaussian assumptions there is a bound on the variance of the RCA-estimator, but in general it is unclear if a representation optimizing the objective for one data-set will also be nearly optimal for another one drawn from the same distribution.

Xing et al. (2002) use both positive and negative equivalence constraints and pose the following optimization problem (in our notation) for a sample $((x_1,x'_1,r_1),...,(x_m,x'_m,r_m))$

$$\min_T \sum_{i:r_i=1} \left\|Tx_i - Tx'_i\right\|^2 \text{ such that } \sum_{i:r_i=-1} \left\|Tx_i - Tx'_i\right\| \geq 1.$$

To solve this problem they propose an algorithm which enforces the positivity constraint for T^*T by alternating gradient descent and projection to the cone of positive operators by eigenvalue decompositions, a method which seems generally applicable when a convex objective is to be optimized under a positivity constraint.

It is surprising that both in Bar-Hillel et al. (2005) and Xing et al. (2002) the existence of a pair oracle to generate a sample of labeled pairs (or equivalence constraints) is implicitly assumed, without attempting to directly predict this oracles behavior. If there is a process which generates labeled examples (and this is what the equivalence constraints are) it seems natural to learn to predict the labels.

This idea has already been proposed by Thrun and Mitchell (1995) (see also Thrun, 1998), where also the obvious connection to transfer and meta-learning is mentioned. This work combines this with the idea of representation learning, as also proposed by Thrun (1998).

Some authors (Chopra et al., 2005; Fleuret and Blanchard 2005) have considered the utility of representation learning for the purpose of multi-category and multi-task pattern recognition on the basis of single training examples. In contrast to the more general method introduced above, these approaches are tailored to the special domain of image processing.

8. Conclusion

This work presented a technique to represent pattern similarity on the basis of data generated by a domain dependent oracle. The method works well in multi-task and multi-category environments

as a preparation for future learning with minimal training sets, such as a single training example or a single training example per category.

A major theoretical problem is to explain the good performance of the method in the context of transfer, a phenomenon which doesn't seem to be completely understood.

Another important development would be a learning algorithm of optimal representations for larger future sample sizes. It is conceivable that, in the context of learning-to-learn, the learner can choose from a catalogue of previously trained representations on the basis of the size of the available training sample. The representations trained by the proposed algorithm would then constitute only one extreme entry in this catalogue, corresponding to a minimal sample size of one.

Appendix A. Notation Table

Notation	Short Description	Section
$\mathcal{X}$	input space, $\mathcal{X} \subset H, diam(\mathcal{X}) \leq 1$	1.1
ρ	pair oracle. P-measure on $\mathcal{X}^2 \times \{-1,1\}$	1.1
(x,x',r)	generic triplet $(x,x',r) \in \mathcal{X}^2 \times \{-1,1\}$	1.1
S	training sample $S \in \left(\mathcal{X}^2 \times \{-1,1\}\right)^m, S \sim \rho^m$	1.1
$R(T)$	risk of operator $= \Pr\left\{r\left(1-\|Tx-Tx'\|^2\right) \leq 0\right\}$	1.2
$\hat{R}_\psi(T,S)$	empirical risk estimate for $\psi \geq 1_{(-\infty,0]}$ $= \frac{1}{m}\sum_{i=1}^m \psi\left(r_i\left(1-\|T(x_i-x_i')\|^2\right)\right)$	1.2
$\hat{R}(T,S)$	empirical risk, $\hat{R}(T,S) = \hat{R}_{1_{(-\infty,0]}}(T,S)$	6.1
h_γ	hinge-loss with margin γ	1.2
λ	regularization parameter	3.2
$\Lambda_{\psi,\lambda}(T,S)$	objective functional $\Lambda_{\psi,\lambda}(T,S) = \hat{R}_\psi(T,S) + m^{-1/2}\lambda\|T^*T\|_2$	3.2
Ω	convex objective, $\Omega(T^*T) = \Lambda(T)$	3.3
H	real, separable Hilbert space	2.1
$\langle .,. \rangle$ and $\|.\|$	inner product and norm on H	2.1
$\|T\|_\infty$	operator norm $\|T\|_\infty = \sup_{x\in H, \|x\|\leq 1}\|Tx\|$	2.1
T^*	adjoint of the operator T	2.1
$\mathcal{L}_\infty(H)$	set of operators on H with $\|T\|_\infty < \infty$	2.1
$\mathcal{L}_\infty^*(H)$	$\{T \in \mathcal{L}_\infty(H) : T^* = T\}$	2.1
$\mathcal{L}_\infty^+(H)$	$\{T \in \mathcal{L}_\infty^*(H) : \langle Tx,x\rangle \geq 0, \forall x \in H\}$	2.1
$\mathcal{L}_0(H)$	set of finite-rank operators on H	2.1
$\|T\|_2$	Hilbert-Schmidt norm	2.1
$\mathcal{L}_2(H)$	set of operators on H with $\|T\|_2 < \infty$	2.1
$\langle T,S\rangle_2$	inner product in $\mathcal{L}_2(H)$	2.1
$\mathcal{L}_2^*(H)$, $\mathcal{L}_2^+(H)$	$\mathcal{L}_\infty^*(H) \cap \mathcal{L}_2(H)$ and $\mathcal{L}_\infty^+(H) \cap \mathcal{L}_2(H)$ resp.	2.1
$\mathcal{P}_d$	d-dimensional orthogonal projections in H	2.1
Q_x, for $x \in H$	the operator $Q_x(z) = \langle z,x\rangle x$	2.1
$\|E\|_{\mathcal{S}}, E \subseteq \mathcal{S}$	maximal norm in E, that is, $\sup_{x\in E}\|x\|_{\mathcal{S}}$	2.1
$\hat{\mathcal{R}}_m(\mathcal{F})$	empirical Rademacher complexity of $\mathcal{F}$	2.2
σ_i	Rademacher variables, uniform on $\{-1,1\}$	2.2

References

M. Anthony and P. Bartlett. *Learning in Neural Networks: Theoretical Foundations*. Cambridge University Press, 1999.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems*, 2006.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 2002.

P. Bartlett, O. Bousquet and S. Mendelson. Local Rademacher complexities. Available online: http://www.stat.berkeley.edu/˜bartlett/papers/bbm-lrc-02b.pdf.

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a Mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6: 937–965, 2005.

J. Baxter. Theoretical models of learning to learn. In *Learning to Learn*, S. Thrun and L. Pratt, Eds., Springer, 1998.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12: 149–198, 2000.

R. Caruana. Multitask learning. In *Learning to Learn*, S. Thrun and L. Pratt, Eds., Springer, 1998.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

N. Cristianini and J. Shawe-Taylor. *Support Vector Machines*. Cambridge University Press, 2000.

T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proc. Conference on Knowledge Discovery and Data Mining*, 2004.

F. Fleuret and G. Blanchard. Pattern recognition from one example by chopping. In *Advances in Neural Information Processing Systems*, 2005.

K. Fukunaga. *Introduction to Statistical Pattern Classification.* Academic Press, 1990.

J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood component analysis. In *Advances in Neural Information Processing Systems*, 2004.

V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1): 1–50, 2002.

A. Maurer. Generalization bounds for subspace selection and hyperbolic PCA. In *Subspace, Latent Structure and Feature Selection. LNCS* 3940: 185–197, Springer, 2006a.

A. Maurer. Bounds for linear multi-task learning. *Journal of Machine Learning Research*, 7:117–139, 2006b.

A. Maurer. Learning to compare using operator-valued large-margin classifiers. In *Advances in Neural Information Processing Systems*, 2006c.

C. McDiarmid. Concentration. In *Probabilistic Methods of Algorithmic Discrete Mathematics*, pages 195–248. Springer, Berlin, 1998.

M. Reed and B. Simon. Functional analysis. In *Methods of Mathematical Physics,* Academic Press, 1980.

A. Robins. Transfer in cognition. In *Learning to Learn*, S. Thrun and L. Pratt, Eds., Springer, 1998.

J. Shawe-Taylor and N. Christianini. Estimating the moments of a random vector. In *Proceedings of GRETSI 2003 Conference I*, pages 47–52, 2003.

B. Simon. *Trace Ideals and Their Applications*. Cambridge University Press, London, 1979.

S. Thrun and T. M. Mitchell. Learning one more thing. In *Proceedings of IJCAI*, 1995.

S. Thrun. Lifelong learning algorithms. In *Learning to Learn*, S. Thrun and L. Pratt, Eds., Springer, 1998.

E. P. Xing, A. Y. Ng, M. I. Jordan and S. Russell. Distance metric learning, with application to clustering with side information. In S. Becker, S. Thrun, and K. Obermayer, Eds., *Advances in Neural Information Processing Systems* 14, MIT Press, Cambridge, MA, 2002.

J. Ye and T. Xiong. Computational and theoretical analysis of null-space and orthogonal linear discriminant analysis. *Journal of Machine Learning Research*, 7:1183–1204, 2006.

Journal of Machine Learning Research 9 (2008) 1083-1114 Submitted 11/07; Revised 4/08; Published 6/08

Ranking Categorical Features Using Generalization Properties*

Sivan Sabato SIVANS@IL.IBM.COM
IBM Haifa Research Lab
Haifa University Campus
Haifa 31905, Israel

Shai Shalev-Shwartz SHAI@TTI-C.ORG
Toyota Technological Institute at Chicago
University Press Building
1427 East 60th Street, Second Floor
Chicago, Illinois 60637, USA

Editor: Yoav Freund

Abstract

Feature ranking is a fundamental machine learning task with various applications, including feature selection and decision tree learning. We describe and analyze a new feature ranking method that supports categorical features with a large number of possible values. We show that existing ranking criteria rank a feature according to the *training* error of a predictor based on the feature. This approach can fail when ranking categorical features with many values. We propose the Ginger ranking criterion, that estimates the *generalization* error of the predictor associated with the Gini index. We show that for almost all training sets, the Ginger criterion produces an accurate estimation of the true generalization error, regardless of the number of values in a categorical feature. We also address the question of finding the optimal predictor that is based on a single categorical feature. It is shown that the predictor associated with the misclassification error criterion has the minimal expected generalization error. We bound the bias of this predictor with respect to the generalization error of the Bayes optimal predictor, and analyze its concentration properties. We demonstrate the efficiency of our approach for feature selection and for learning decision trees in a series of experiments with synthetic and natural data sets.

Keywords: feature ranking, categorical features, generalization bounds, Gini index, decision trees

1. Introduction

In this paper we address the problem of supervised feature ranking in the presence of categorical features. Feature ranking mechanisms have various applications; For instance, they can be used to define a filter for feature selection or as a splitting criterion for growing decision trees. In the feature ranking task we order a given set of features according to their relevance for predicting a target label. As in other supervised learning tasks, the ranking of the features is generated based on an input training set. Examples of widely used feature ranking criteria are the Gini index, the misclassification error, and Information Gain, also termed 'cross-entropy' (Hastie et al., 2001). The focus of this paper is feature ranking in the presence of *categorical* features. We show that a direct application of existing ranking criteria might lead to poor results in the presence of categorical

*. A preliminary version of this paper appeared at the 20th Annual Conference on Learning Theory under the title "Prediction by Categorical Features: Generalization Properties and Application to Feature Ranking"

features that can take many values. We propose an adaptation of existing ranking criteria that copes with these difficulties.

Many feature ranking methods are equivalent to the following two-phase process: First, each individual feature is used to construct a predictor of the label. Then, the features are ranked based on the errors of these predictors. Most current approaches use the same training set both for constructing the predictor and for evaluating its error. When dealing with binary features, the training error is likely to be close to the generalization error, and therefore the ranking generated by current methods works rather well. However, this is not the case when dealing with categorical features that can take a large number of values. To illustrate this fact, consider the problem of predicting whether someone is unemployed, based on their social security number (SSN). A predictor constructed using any finite training set would have zero error on the training set but a large generalization error. Therefore, a ranking criterion that supports categorical features should employ a more robust estimation of the generalization error.

The first contribution of this paper is an estimator for the generalization error of the predictor associated with the Gini index. This estimator can be calculated from the training set and we propose to use it instead of the original Gini index criterion in the presence of categorical features. We prove that regardless of the underlying distribution, our estimation is close to the true value of the generalization error for almost all training sets.

Based on our perspective of ranking criteria as estimators of the generalization error of a certain predictor, a natural question that arises is which predictor to use. Among all predictors that are based on a single feature, we ultimately would like to use the one whose generalization error is minimal. We prove that the best predictor in this sense is the predictor associated with the misclassification error criterion. We analyze the difference between the expected generalization error of this predictor and the error of the Bayes optimal hypothesis. Finally, we show a concentration result for the generalization error of this predictor.

Feature ranking criteria have been extensively studied in the context of decision trees (Mingers, 1989; Kearns and Mansour, 1996; Quinlan, 1993). The failure of existing feature ranking criteria in the presence of categorical features with a large number of possible values has been previously discussed in Quinlan (1993) and Mitchell (1997). Quinlan suggested the Information Gain Ratio as a correction to the Information Gain criterion. In a broader context, information-theoretic measures are commonly used for feature ranking (see for example Torkkola, 2006, and the references therein). One justification for their use is the existence of bounds on the Bayes optimal error that are based on these measures (Torkkola, 2006). However, obtaining estimators for the entropy or mutual information seems to be difficult in the general case (Antos and Kontoyiannis, 2001). Another ranking criterion designed to address the above difficulty is a distance-based measure introduced by de Mantaras (1991).

The problem we address shares some similarities with the problem of estimating the missing mass of a sample, typically encountered in language modeling (Good, 1953; McAllester and Schapire, 2000; Drukh and Mansour, 2005). The missing mass of a sample is the total probability mass of the values not occurring in the sample. Indeed, in the aforementioned example of the SSN feature, the value of the missing mass will be close to one. In some of our proofs we borrow ideas from McAllester and Schapire (2000) and Drukh and Mansour (2005). However, our problem is more involved, as even for a value that we do observe in the sample, if it appears only a small number of times then the training error is likely to diverge from the generalization error. Finally, we would like to note that classical VC theory for bounding the difference between the training error

and the generalization error is not applicable here. This is because the VC dimension grows with the number of values a categorical feature may take, and in our framework this number is unbounded.

This paper is organized as follows. In Sec. 2 we formally describe our problem setting. We introduce our main results in Sec. 3 and prove them in Sec. 4. We present experimental results in Sec. 5 and concluding remarks are given in Sec. 6.

2. Problem Setting

In this section we establish the notation used throughout the paper and formally describe our problem setting. In the supervised feature ranking setting we are provided with k categorical features and with a label. Each categorical feature is a random variable that takes values from a finite set. We denote a feature by X and the set of values X can take by V. We make no assumptions on the identity of V for each X nor on its size. The label is a binary random variable, denoted Y, that takes values from $\{0,1\}$.

Generally speaking, the goal of supervised feature ranking is to rank the features based on their merit in constructing an accurate classification rule. The features are ranked according to their "relevance" to the label. Different criteria exist for assessing the relevance of a feature to the label. Since relevance is assessed for each feature separately, let us ignore the fact that we have k features and from now on focus on defining a relevance measure for a single feature X. We denote by V the set of values that X can take. To simplify our notation we denote

$$p_v \stackrel{\Delta}{=} \Pr[X = v] \quad \text{and} \quad q_v \stackrel{\Delta}{=} \Pr[Y = 1 | X = v].$$

In practice, the probabilities $\{p_v\}$ and $\{q_v\}$ are unknown. Instead, it is assumed that we have a training set $S = \{(x_i, y_i)\}_{i=1}^m$, which is sampled i.i.d. according to the joint probability distribution $\Pr[X,Y]$. Based on S, the probabilities $\{p_v\}$ and $\{q_v\}$ are usually estimated as follows. Let $c_v = |\{i : x_i = v\}|$ be the number of examples in S for which the feature takes the value v and let $c_v^+ = |\{i : x_i = v \wedge y_i = 1\}|$ be the number of examples in which the value of the feature is v and the label is 1. Then $\{p_v\}$ and $\{q_v\}$ are estimated as follows:

$$\hat{p}_v \stackrel{\Delta}{=} \frac{c_v}{m} \quad \text{and} \quad \hat{q}_v \stackrel{\Delta}{=} \begin{cases} \frac{c_v^+}{c_v} & c_v > 0 \\ \frac{1}{2} & c_v = 0. \end{cases}$$

Note that $\hat{p}_v$ and $\hat{q}_v$ are implicit functions of the training set S.

Two popular relevance criteria (Hastie et al., 2001) are the misclassification error

$$\sum_{v \in V} \hat{p}_v \min\{\hat{q}_v, (1 - \hat{q}_v)\} , \tag{1}$$

and the Gini index

$$2 \sum_{v \in V} \hat{p}_v \hat{q}_v (1 - \hat{q}_v) . \tag{2}$$

In these criteria, smaller values indicate more relevant features.

Both the misclassification error and the Gini index were found to work rather well in practice when $|V|$ is small. However, for categorical features with a large number of possible values, we might end up with a poor feature ranking criterion. As an example (see Mitchell, 1997), suppose that Y indicates whether a person is unemployed and we have two features: X_1 is the person's SSN

and X_2 is 1 if the person has a mortgage and 0 otherwise. For the first feature, V is the set of all the SSNs. Because the SSN alone determines the target label, we have that $\hat{q}_v$ is either 0 or 1 for any v such that $\hat{p}_v > 0$. Thus, both the misclassification error and the Gini index are zero for this feature. For the second feature, it can be shown that with high probability over the choice of the training set, the two criteria mentioned above take positive values. Therefore, both criteria prefer the first feature over the second. In contrast, for our purposes X_2 is much better than X_1. This is because X_2 can be used later for learning a reasonable classification rule based on a finite training set, while X_1 will suffer from over-fitting.

It would have been natural to attribute the failure of the relevance criteria to the fact that we use estimated probabilities instead of the true (unknown) probabilities. However, note that in the above example, the same problem would arise even if we used $\{p_v\}$ and $\{q_v\}$ in Eq. (1) and Eq. (2). The aforementioned problem was previously underscored in the context of the Information Gain criterion (Quinlan, 1993; de Mantaras, 1991; Mitchell, 1997). In that context, Quinlan (1993) suggested an adaptation of the Information Gain, called Information Gain Ratio, which was found rather effective in practice.

In this paper we take a different approach, and propose to interpret a feature ranking criterion as the generalization error of a classification rule that can be inferred from the training set. To do so, let us first introduce some additional notation. A probabilistic hypothesis is a function $h : V \to [0,1]$, where $h(v)$ is the probability to predict the label 1 given the value v. The generalization error of h is the probability to incorrectly predict the label,

$$\ell(h) \stackrel{\Delta}{=} \sum_{v \in V} p_v \left(q_v \left(1 - h(v)\right) + \left(1 - q_v\right) h(v) \right) \quad . \tag{3}$$

We now define two hypotheses based on the training set S. The first one is

$$h_S^{\text{Gini}}(v) = \hat{q}_v \quad . \tag{4}$$

As its name indicates, h_S^{Gini} is closely related to the Gini index filter given in Eq. (2). To see this, we note that the generalization error of h_S^{Gini} is

$$\ell(h_S^{\text{Gini}}) = \sum_{v \in V} p_v \left(q_v \left(1 - \hat{q}_v\right) + \left(1 - q_v\right) \hat{q}_v \right) \quad .$$

If the estimated probabilities $\{\hat{p}_v\}$ and $\{\hat{q}_v\}$ coincide with the true probabilities $\{p_v\}$ and $\{q_v\}$, then $\ell(h_S^{\text{Gini}})$ is identical to the Gini index defined in Eq. (2). This will be approximately true, for example, when $m \gg |V|$. In other words, the Gini index is the training error of h_S^{Gini}. When the training set is small, using $\ell(h_S^{\text{Gini}})$ is preferable to using the Gini index given in Eq. (2), because $\ell(h_S^{\text{Gini}})$ takes into account the fact that the estimated probabilities might be skewed.

The second hypothesis we define is

$$h_S^{\text{Bayes}}(v) \;=\; \begin{cases} 1 & \hat{q}_v > \frac{1}{2} \\ 0 & \hat{q}_v < \frac{1}{2} \\ \frac{1}{2} & \hat{q}_v = \frac{1}{2} \end{cases} \quad . \tag{5}$$

Note that if $\{\hat{q}_v\}$ coincide with $\{q_v\}$ then h_S^{Bayes} is the Bayes optimal classifier, which we denote by h_∞^{Bayes}. If in addition $\{\hat{p}_v\}$ and $\{p_v\}$ are the same, then $\ell(h_S^{\text{Bayes}})$ is identical to the misclassification

error defined in Eq. (1). Here again, the misclassification error might differ from $\ell(h_S^{\text{Bayes}})$ for small training sets.

To illustrate the advantage of $\ell(h_S^{\text{Gini}})$ and $\ell(h_S^{\text{Bayes}})$ over their counterparts given in Eq. (2) and Eq. (1), we return to the example mentioned above. For X_1, the SSN feature we have $\ell(h_S^{\text{Gini}}) = \ell(h_S^{\text{Bayes}}) = \frac{1}{2}M_0$, where $M_0 \stackrel{\Delta}{=} \sum_{v:c_v=0} p_v$. In general, we denote

$$M_k \stackrel{\Delta}{=} \sum_{v:c_v=k} p_v \ . \tag{6}$$

The quantity M_0 is known as the missing mass (Good, 1953; McAllester and Schapire, 2000) and for the SSN feature, $M_0 \geq (|V|-m)/|V|$. Therefore, the generalization error of both h_S^{Gini} and h_S^{Bayes} would be close to 1 for a reasonable m. On the other hand, for X_2, the feature of having a mortgage, it can be verified that both $\ell(h_S^{\text{Bayes}})$ and $\ell(h_S^{\text{Gini}})$ are likely to be small. Therefore, using $\ell(h_S^{\text{Gini}})$ or $\ell(h_S^{\text{Bayes}})$ yields a correct ranking for this naive example.

We have proposed a modification of the Gini index and the misclassification error that uses the generalization error and therefore is suitable even when m is smaller than $|V|$. In practice, however, we cannot directly use the generalization error criterion since it depends on the unknown probabilities $\{p_v\}$ and $\{q_v\}$. To overcome this obstacle, we must derive estimators for the generalization error that can be calculated from the training set. In the next section we discuss the problem of estimating $\ell(h_S^{\text{Gini}})$ and $\ell(h_S^{\text{Bayes}})$ based on the training set. Additionally, we analyze the difference between $\ell(h_S^{\text{Bayes}})$ and the error of the Bayes optimal hypothesis.

3. Main Results

We start this section with a derivation of an estimator for $\ell(h_S^{\text{Gini}})$, which can serve as a new feature ranking criterion. We show that for most training sets, this estimator will be close to the true value of $\ell(h_S^{\text{Gini}})$. We then shift our attention to $\ell(h_S^{\text{Bayes}})$. First, we prove that among all predictors with no prior knowledge on the distribution $\Pr[X,Y]$, the generalization error of h_S^{Bayes} is smallest in expectation. Next, we bound the difference between the generalization error of h_S^{Bayes} and the error of the Bayes optimal hypothesis. Finally, we prove a concentration bound for $\ell(h_S^{\text{Bayes}})$. Regretfully, we could not find a good estimator for $\ell(h_S^{\text{Bayes}})$. Nevertheless, we believe that our concentration results can be used for finding such an estimator. This task is left for future research.

We propose the following estimator for the generalization error of h_S^{Gini}:

$$\hat{\ell} \stackrel{\Delta}{=} \frac{|\{v : c_v = 1\}|}{2m} + \sum_{v:c_v>1} \frac{2c_v}{c_v - 1} \hat{p}_v \hat{q}_v (1 - \hat{q}_v) \ . \tag{7}$$

This estimator can be derived using a leave-one-out technique (see for instance Wasserman, 2004). In the next section we show a different derivation, based on a conditional cross-validation technique. We suggest to use the estimation of $\ell(h_S^{\text{Gini}})$ given in Eq. (7) rather than the original Gini index given in Eq. (2) as a feature ranking criterion. Let us compare these two criteria: First, for values v that appear many times in the training set we have that $\frac{c_v}{c_v-1} \approx 1$. If for all $v \in V$ we have that the size of the training set is much larger than $1/p_v$, then all values in V are likely to appear many times in the training set and thus the definitions in Eq. (7) and Eq. (2) consolidate. The two definitions differ when there are values that appear rarely in the training set. For such values, the correction term is larger than 1. Special consideration is given to values that appear exactly once in the training

set. For such values we estimate the generalization error to be $\frac{1}{2}$, which is the highest possible error. Intuitively, since one example provides us with no information as to the variance of the label Y given $X = v$, we cannot have a more accurate estimation for the contribution of this value to the total generalization error. Furthermore, the fraction of values that appear exactly once in the training set is an estimator for the probability mass of those values that do not appear at all in the training set (see also Good, 1953; McAllester and Schapire, 2000).

We now turn to analyze the quality of the proposed estimator. We first show in Thm. 1 that the bias of this estimator is small. Then, in Thm. 2, we prove a concentration bound for the estimator, which holds for any joint distribution of $\Pr[X,Y]$ and does not depend on the size of V. Specifically, we show that for any $\delta \in (0,1)$, in a fraction of at least $1-\delta$ of the training sets the error of the estimator is $O(\frac{\ln(m/\delta)}{\sqrt{m}})$.

Theorem 1 *Let S be a set of m examples sampled i.i.d. according to the probability measure $\Pr[X,Y]$. Let h_S^{Gini} be the Gini hypothesis given in Eq. (4) and let $\ell(h_S^{Gini})$ be the generalization error of h_S^{Gini}, where ℓ is as defined in Eq. (3). Let $\hat{\ell}$ be the estimation of $\ell(h_S^{Gini})$ as given in Eq. (7). Then, $\left|\mathbb{E}[\ell(h_S^{Gini})] - \mathbb{E}[\hat{\ell}]\right| \leq \frac{1}{2m}$, where expectation is taken over all samples S of m examples.*

The next theorem shows that for most training sets, our estimator is close to the true generalization error of h_S^{Gini}.

Theorem 2 *Under the same assumptions as in Thm. 1, let δ be an arbitrary scalar in $(0,1)$. Then, with probability of at least $1-\delta$ over the choice of S, we have*

$$\left|\ell(h_S^{Gini}) - \hat{\ell}\right| \leq O\left(\frac{\ln(m/\delta)\sqrt{\ln(1/\delta)}}{\sqrt{m}}\right) \ .$$

Based on the above theorem, $\hat{\ell}$ can be used as a ranking criterion. The convergence rate shown can be used to establish confidence intervals on the true Gini generalization error. The proofs of Thm. 1 and Thm. 2 are given in the next section.

So far we have derived an estimator for the generalization error of the Gini hypothesis and shown that it is close to the true Gini error. The Gini hypothesis has the advantage of being highly concentrated around its mean. This is important especially when the sample size is fairly small. However, the Gini hypothesis does not produce the lowest generalization error in expectation. We now turn to show that the hypothesis h_S^{Bayes} defined in Eq. (5) is optimal in this respect, but that its concentration might be weaker. These two facts are characteristic of the well known bias-variance tradeoff commonly found in estimation and prediction tasks.

Had we known the underlying distribution of our data, we could have used the Bayes optimal hypothesis, h_∞^{Bayes}, that achieves the smallest possible generalization error. When the underlying distribution is unknown, the training set is used to construct the hypothesis. Thm. 3 below shows that among all hypotheses that can be learned from a finite training set, h_S^{Bayes} achieves the smallest generalization error in expectation. More precisely, h_S^{Bayes} is optimal among all the hypotheses that are symmetric with respect to both $|V|$ and the label values. Clearly, symmetric hypotheses cannot exploit prior knowledge on the underlying distribution $\Pr[X,Y]$. Formally, let $\mathcal{F}$ be the set of all symmetric functions over $\mathbb{N}\times\mathbb{N}$, that is,

$$\mathcal{F} \ = \ \{\, f : \mathbb{N}\times\mathbb{N} \to [0,1] \mid \forall n_1, n_2 \in \mathbb{N},\ f(n_1,n_2) = 1 - f(n_1, n_1 - n_2) \,\}$$

and let H be the following set of mappings from samples of size m to hypotheses:

$$H = \left\{ h : (V \times \{0,1\})^m \to V^{[0,1]} \;\middle|\; \exists f \in \mathcal{F} \text{ s.t. } \forall S \in (V \times \{0,1\})^m, \forall v \in V, \quad h[S](v) = f(c_v(S), c_v^+(S)) \right\} . \tag{8}$$

That is, H is the set of mappings that given a sample, generate a hypothesis based solely on the sample. Thus, hypotheses that rely on any prior knowledge on $\Pr[X,Y]$ are excluded.

The following theorem establishes the optimality of h_S^{Bayes} and bounds the difference between the Bayes optimal error and the error achieved by h_S^{Bayes}.

Theorem 3 *Let S be a set of m examples sampled i.i.d. according to the probability measure $\Pr[X,Y]$. For any hypothesis h, let $\ell(h)$ be the generalization error of h, as defined in Eq. (3). Let h_S^{Bayes} be the hypothesis given in Eq. (5), let h_∞^{Bayes} be the Bayes optimal hypothesis, and let H be the set of hypothesis mappings defined in Eq. (8). Then*

$$\mathbb{E}[\ell(h_S^{Bayes})] = \min_{h \in H} \mathbb{E}[\ell(h[S])], \tag{9}$$

and

$$\mathbb{E}[\ell(h_S^{Bayes})] - \ell(h_\infty^{Bayes}) \leq \frac{1}{2}\mathbb{E}[M_0] + \frac{1}{8}\mathbb{E}[M_1] + \frac{1}{8}\mathbb{E}[M_2] + \sum_{k=3}^{m} \frac{1}{\sqrt{ek}}\mathbb{E}[M_k], \tag{10}$$

where M_k is as defined in Eq. (6). Furthermore,

$$\lim_{m\to\infty} \left(\frac{1}{2}\mathbb{E}[M_0] + \frac{1}{8}\mathbb{E}[M_1] + \frac{1}{8}\mathbb{E}[M_2] + \sum_{k=3}^{m} \frac{1}{\sqrt{ek}}\mathbb{E}[M_k] \right) = 0. \tag{11}$$

Note that the first term in the difference between $\mathbb{E}[\ell(h_S^{\text{Bayes}})]$ and $\ell(h_\infty^{\text{Bayes}})$ is exactly half the expectation of the missing mass. This is expected, because we cannot improve our prediction over the baseline error of $\frac{1}{2}$ for values not seen in the training set, as exemplified in the SSN example described in the previous section. Subsequent terms in the bound can be attributed to the fact that even for values observed in the training set, a wrong prediction might be generated if there is a small number of examples.

We have shown that h_S^{Bayes} has the smallest generalization error in expectation, but this does not guarantee a small generalization error on a given sample. Thm. 4 below bounds the concentration of $\ell(h_S^{\text{Bayes}})$. This concentration along with Thm. 3 provides us with a bound on the difference between h_S^{Bayes} and the Bayes optimal error that is true for most samples.

Theorem 4 *Under the same assumptions of Thm. 3, assume that $m \geq 8$ and let δ be an arbitrary scalar in $(0,1)$. Then, with probability of at least $1-\delta$ over the choice of S, we have*

$$|\ell(h_S^{Bayes}) - \mathbb{E}[\ell(h_S^{Bayes})]| \leq O\left(\frac{\ln(m/\delta)\sqrt{\ln(1/\delta)}}{m^{1/6}} \right).$$

The concentration bound for $\ell(h_S^{\text{Bayes}})$ is weaker than the concentration bound for $\ell(h_S^{\text{Gini}})$, suggesting that indeed the choice between h_S^{Gini} and h_S^{Bayes} is not trivial. To use $\ell(h_S^{\text{Bayes}})$ as a ranking criterion, an estimator for this quantity is needed. However, at this point we cannot provide such an estimator. We conjecture that based on Thm. 4 an estimator with a small bias but a weak concentration can be constructed. We leave this task to further work. Finally, we would like to note that Antos et al. (1999) have shown that the Bayes optimal error cannot be estimated based on a finite training set. Finding an estimator for $\ell(h_S^{\text{Bayes}})$ would allow us to approximate the Bayes optimal error up to the bias term quantified in Thm. 3.

4. Proofs of Main Results

In this section we provide the full proofs of the theorems presented above.

4.1 Proof of Thm. 1

In the previous section, an estimator for the generalization error of the Gini hypothesis was presented. We stated that for most training sets this estimation is reliable. In this section, we first derive the estimator $\hat{\ell}$ given in Eq. (7) using a conditional cross-validation technique, and then use this interpretation of $\hat{\ell}$ to prove Thm. 1 and Thm. 2.

To derive the estimator given in Eq. (7), let us first rewrite $\ell(h_S^{\text{Gini}})$ as the sum $\sum_v \ell_v(h_S^{\text{Gini}})$, where $\ell_v(h_S^{\text{Gini}})$ is the amount of error due to value v and is formally defined as

$$\ell_v(h) \stackrel{\Delta}{=} \Pr[X=v]\Pr[h(X)\neq Y \mid X=v] = p_v\left(q_v(1-h(v))+(1-q_v)h(v)\right) .$$

We now estimate the two factors $\Pr[X=v]$ and $\Pr[h_S^{\text{Gini}}(X)\neq Y \mid X=v]$ independently. Later on we multiply the two estimations. The resulting local estimator of $\ell_v(h)$ is denoted $\hat{\ell}_v$ and our global estimator is $\hat{\ell} \stackrel{\Delta}{=} \sum_v \hat{\ell}_v$.

To estimate $\Pr[X=v]$, we use the straightforward estimator $\hat{p}_v$. Turning to the estimation of $\Pr[h_S^{\text{Gini}}(X)\neq Y \mid X=v]$, recall that h_S^{Gini}, defined in Eq. (4), is a probabilistic hypothesis where $\hat{q}_v$ is the probability to return the label 1 given that the value of X is v. Equivalently, we can think of the label that $h_S^{\text{Gini}}(v)$ returns as being generated based on the following process: Let $S(v)$ be the set of those indices in the training set in which the feature takes the value v, namely, $S(v)=\{i : x_i = v\}$. Then, to set the label $h_S^{\text{Gini}}(v)$ we randomly choose an index $i \in S(v)$ and return the label y_i. Based on this interpretation, a natural path for estimating $\Pr[h_S^{\text{Gini}}(X)\neq Y \mid X=v]$ is through cross-validation: Select an $i \in S(v)$ to determine $h_S^{\text{Gini}}(v)$, and estimate the generalization error to be the fraction of the examples whose label is different from the label of the selected example. That is, the estimation is $\frac{1}{c_v-1}\sum_{j\in S(v):j\neq i}\mathbf{1}_{y_i\neq y_j}$. Obviously, this procedure cannot be used if $c_v=1$. We handle this case separately later on. To reduce the variance of this estimation, this process can be repeated, selecting each single example from $S(v)$ in turn and validating each time using the rest of the examples in $S(v)$. It is then possible to average over all the choices of the examples. The resulting estimation therefore becomes

$$\sum_{i\in S(v)}\frac{1}{c_v}\left(\frac{1}{c_v-1}\sum_{j\in S(v):j\neq i}\mathbf{1}_{y_i\neq y_j}\right) = \frac{1}{c_v(c_v-1)}\sum_{i,j\in S(v):i\neq j}\mathbf{1}_{y_i\neq y_j} .$$

Thus, we estimate $\Pr[h_S^{\text{Gini}}(X)\neq Y \mid X=v]$ based on the fraction of differently-labeled pairs of examples in $S(v)$. Multiplying this estimator by $\hat{p}_v$ we obtain the following estimator for $\ell_v(h_S^{\text{Gini}})$,

$$\begin{aligned}\hat{\ell}_v &= \hat{p}_v\frac{1}{c_v(c_v-1)}\sum_{i,j\in S(v),i\neq j}\mathbf{1}_{y_i\neq y_j} \\ &= \hat{p}_v\frac{2c_v^+(c_v-c_v^+)}{c_v(c_v-1)} = \hat{p}_v\frac{2c_v^2\hat{q}_v(1-\hat{q}_v)}{c_v(c_v-1)} = \hat{p}_v\cdot\frac{2c_v}{c_v-1}\hat{q}_v(1-\hat{q}_v).\end{aligned} \tag{12}$$

Finally, for values v that appear only once in the training set, the above cross-validation procedure cannot be applied, and we therefore estimate their generalization error to be $\frac{1}{2}$, the highest possible

error. The full definition of $\hat{\ell}_v$ is thus:

$$\hat{\ell}_v = \begin{cases} \hat{p}_v \cdot \frac{1}{2} & c_v \leq 1 \\ \hat{p}_v \cdot \frac{2c_v}{c_v - 1} \hat{q}_v (1 - \hat{q}_v) & c_v \geq 2. \end{cases} \tag{13}$$

The resulting estimator $\hat{\ell}$ defined in Eq. (7) is exactly the sum $\sum_v \hat{\ell}_v$.

Based on the above derivation of $\hat{\ell}_v$, we now turn to prove Thm. 1, in which it is shown that the expectations of our estimator and of the true generalization error of the Gini hypothesis are close. To do so, we first inspect each of these expectations separately, starting with $\mathbb{E}[\hat{\ell}_v]$. The following lemma calculates the expectation of $\hat{\ell}_v$ over those training sets with exactly k appearances of the value v.

Lemma 5 *For k such that $1 < k \leq m$, $\mathbb{E}[\hat{\ell}_v \mid c_v(S) = k] = \frac{k}{m} \cdot 2q_v(1 - q_v)$.*

Proof If $c_v = k$, then $\hat{p}_v = \frac{k}{m}$. Therefore, based on Eq. (12), we have

$$\mathbb{E}[\hat{\ell}_v \mid c_v(S) = k] = \frac{k}{m} \frac{1}{k(k-1)} \mathbb{E}\left[\sum_{i,j \in S(v), i \neq j} \mathbf{1}_{y_i \neq y_j} \mid c_v(S) = k \right] . \tag{14}$$

Let $Z_1, \ldots, Z_k$ be independent binary random variables with $\Pr[Z_i = 1] = q_v$ for all $i \in [k]$. The conditional expectation on the right-hand side of Eq. (14) equals to

$$\mathbb{E}[\sum_{i \neq j} \mathbf{1}_{Z_i \neq Z_j}] \;=\; \sum_{i \neq j} \mathbb{E}[\mathbf{1}_{Z_i \neq Z_j}] \;=\; \sum_{i \neq j} 2\,q_v\,(1 - q_v) \;=\; k(k-1) \cdot 2\,q_v\,(1 - q_v) \;.$$

Combining the above with Eq. (14) concludes the proof. ■

Based on the above lemma, we are now ready to calculate $\mathbb{E}[\hat{\ell}_v]$. We have

$$\mathbb{E}[\hat{\ell}_v] = \sum_S \Pr[S]\, \mathbb{E}[\hat{\ell}_v] = \sum_{k=0}^{m} \sum_{S : c_v(S) = k} \Pr[S] \cdot \mathbb{E}[\hat{\ell}_v \mid c_v(S) = k]. \tag{15}$$

From the definition of $\hat{\ell}$, we have $\mathbb{E}[\hat{\ell}_v \mid c_v(S) = 1] = \frac{1}{2m}$ and $\mathbb{E}[\hat{\ell}_v \mid c_v(S) = 0] = 0$. Combining this with Lemma 5 and Eq. (15), we get

$$\begin{aligned} E[\hat{\ell}_v] &= \Pr[c_v = 1] \cdot \frac{1}{2m} + \sum_{k=2}^{m} \Pr[c_v = k] \cdot \frac{k}{m} \cdot 2q_v(1 - q_v) \\ &= \frac{1}{m} \left(\frac{1}{2} - 2q_v(1 - q_v)\right) \Pr[c_v = 1] + 2q_v(1 - q_v) \sum_{k=0}^{m} \Pr[c_v = k] \cdot \frac{k}{m} \\ &= \frac{1}{m} \left(\frac{1}{2} - 2q_v(1 - q_v)\right) \Pr[c_v = 1] + p_v \cdot 2q_v(1 - q_v) \;, \end{aligned} \tag{16}$$

where the last equality follows from the fact that $\sum_{k=0}^{m} \Pr[c_v = k] \frac{k}{m} = \mathbb{E}[\hat{p}_v] = p_v$. Having calculated the expectation of $\hat{\ell}_v$ we now calculate the expectation of $\ell_v(h_S^{\text{Gini}})$.

Lemma 6 $\mathbb{E}[\ell_v(h_S^{Gini})] = p_v(\frac{1}{2} - 2q_v(1 - q_v)) \Pr[c_v = 0] + p_v \cdot 2q_v(1 - q_v)$.

Proof From the definition of $\ell_v(h_S^{\text{Gini}})$, we have that

$$\begin{aligned}\mathbb{E}[\ell_v(h_S^{\text{Gini}})] &= \mathbb{E}[p_v\,(q_v(1-h_S^{\text{Gini}}(v))+(1-q_v)h_S^{\text{Gini}}(v))] \\ &= p_v\,(q_v(1-\mathbb{E}[h_S^{\text{Gini}}(v)])+(1-q_v)\,\mathbb{E}[h_S^{\text{Gini}}(v)]) \\ &= p_v\,(q_v+(1-2\,q_v)\,\mathbb{E}[h_S^{\text{Gini}}(v)]))\ . \end{aligned} \tag{17}$$

Next, we calculate $\mathbb{E}[h_S^{\text{Gini}}(v)]$ as follows

$$\begin{aligned}\mathbb{E}[h_S^{\text{Gini}}(v)] &= \sum_S \Pr[S] h_S^{\text{Gini}}(v) \\ &= \Pr[c_v(S)=0]\cdot\frac{1}{2} \;+\; \sum_{k=1}^{m}\sum_{i=0}^{k}\Pr[c_v(S)=k \text{ and } c_v^+(S)=i]\,\frac{i}{k} \\ &= \Pr[c_v(S)=0]\cdot\frac{1}{2} \;+\; \sum_{k=1}^{m}\Pr[c_v(S)=k]\sum_{i=0}^{k}\Pr[c_v^+(S)=i \mid c_v(S)=k]\,\frac{i}{k} \\ &= \Pr[c_v(S)=0]\cdot\frac{1}{2}+\sum_{k=1}^{m}\Pr[c_v(S)=k]\cdot q_v \\ &= \Pr[c_v(S)=0]\cdot\frac{1}{2}+\Pr[c_v(S)>0]\cdot q_v \\ &= q_v+\frac{1}{2}(1-2\,q_v)\Pr[c_v(S)=0]\ . \end{aligned} \tag{18}$$

Plugging Eq. (18) into Eq. (17) and rearranging terms we conclude our proof. ■

Equipped with the expectation of $\hat{\ell}_v$ given in Eq. (16) and the expectation of $\ell_v(h_S^{\text{Gini}})$ given in Lemma 6, we are now ready to prove Thm. 1.

Proof [of Thm. 1] Using the definitions of $\ell(h_S^{\text{Gini}})$ and $\hat{\ell}$ we have that

$$\mathbb{E}[\hat{\ell}]-\mathbb{E}[\ell(h_S^{\text{Gini}})] \;=\; \mathbb{E}[\sum_v \hat{\ell}_v]-\mathbb{E}[\sum_v \ell_v(h_S^{\text{Gini}})] \;=\; \sum_v(\mathbb{E}[\hat{\ell}_v]-\mathbb{E}[\ell_v(h_S^{\text{Gini}})])\ . \tag{19}$$

Fix some $v\in V$. From Eq. (16) and Lemma 6 we have

$$\mathbb{E}[\hat{\ell}_v]-\mathbb{E}[\ell_v(h_S^{\text{Gini}})] = (\frac{1}{2}-2q_v(1-q_v))(\frac{1}{m}\Pr[c_v=1]-p_v\Pr[c_v=0])\ . \tag{20}$$

Also, it is easy to see that

$$\begin{aligned}\frac{1}{m}\Pr[c_v=1]-p_v\Pr[c_v=0] &= p_v(1-p_v)^{m-1}-p_v(1-p_v)^m \\ &= p_v^2(1-p_v)^{m-1}=\frac{p_v}{m}\Pr[c_v=1]\ . \end{aligned}$$

Plugging this into Eq. (20) we obtain:

$$\mathbb{E}[\hat{\ell}_v]-\mathbb{E}[\ell_v(h_S^{\text{Gini}})] = (\frac{1}{2}-2q_v(1-q_v))\frac{1}{m}p_v\Pr[c_v=1].$$

For any q_v we have that $0\le 2q_v(1-q_v)\le\frac{1}{2}$, which implies the following inequality:

$$0\le\mathbb{E}[\hat{\ell}_v]-\mathbb{E}[\ell_v(h_S^{\text{Gini}})]\le\frac{1}{2m}\,p_v\Pr[c_v=1]\le\frac{p_v}{2m}.$$

Summing this over v and using Eq. (19) we conclude that

$$0 \;\leq\; \mathbb{E}[\hat{\ell}] - \mathbb{E}[\ell(h_S^{\text{Gini}})] \;\leq\; \sum_v \frac{p_v}{2m} \;=\; \frac{1}{2m} \;.$$

■

4.2 Proof of Thm. 2

We now turn to prove Thm. 2 in which we argue that with high confidence on the choice of S, the value of our estimator is close to the actual generalization error of h_S^{Gini}. To do this, we show that both our estimator and the true generalization error of h_S^{Gini} are concentrated around their mean. The proof of Thm. 2 will then follow from Thm. 1.

We start by showing that our estimator $\hat{\ell}$ is concentrated around its expectation. The concentration of $\hat{\ell}$ follows relatively easily by application of McDiarmid's Theorem (McDiarmid, 1989):

Theorem 7 (McDiarmid) *Let $X_1,\ldots,X_m$ be independent random variables taking values in a set V and let $f : V^m \to \mathbb{R}$ be such that for every $1 \leq i \leq m$*

$$\sup |f(x_1,\ldots,x_m) - f(x_1,\ldots,x_{i-1},x_i',x_{i+1},\ldots,x_m)| \leq c_i$$

where the supremum is taken over all $x_1,\ldots,x_m,x_i' \in V$. Then with probability at least $1-\delta$

$$f(X_1,\ldots,X_m) \leq \mathbb{E}[f(X_1,\ldots,X_m)] + \sqrt{\frac{1}{2}\ln(\frac{1}{\delta})\sum_{i=1}^m c_i}$$

and with probability at least $1-\delta$

$$f(X_1,\ldots,X_m) \geq \mathbb{E}[f(X_1,\ldots,X_m)] - \sqrt{\frac{1}{2}\ln(\frac{1}{\delta})\sum_{i=1}^m c_i} \;.$$

To simplify our notation, we will henceforth use the shorthand $\forall^\delta S \quad \pi[S,\delta]$ to indicate that the predicate $\pi[S,\delta]$ holds with probability of at least $1-\delta$ over the choice of S.

Lemma 8 *Let $\delta \in (0,1)$. Then, $\forall^\delta S \quad \left|\hat{\ell} - \mathbb{E}[\hat{\ell}]\right| \leq 12\sqrt{\frac{\ln(\frac{2}{\delta})}{2m}}$.*

Proof We prove the lemma using McDiardmid's theorem. To do so, we need to show that $\hat{\ell}$ has the bounded differences property; namely, we shall find an upper bound for the effect of any change of a single example in S on $\hat{\ell}$. Changing example (x_i,y_i) in S to (x_i',y_i') is tantamount to first removing (x_i,y_i) and then adding (x_i',y_i'). Since the effect of adding is simply the opposite of the effect of removing, it is sufficient to find an upper bound for the effect a single removal of example can have. Then the effect of a change on the sample would be no larger than twice the effect of the removal.

Let $S^{\setminus i}$ denote the set $S \setminus \{(x_i,y_i)\}$. We therefore need to bound $|\hat{\ell}(S) - \hat{\ell}(S^{\setminus i})|$. Assume, without loss of generality, that $x_i = v$ and $y_i = 0$. Then, using the definition of $\hat{\ell}_v$ we have that

$$|\hat{\ell}(S) - \hat{\ell}(S^{\setminus i})| = |\hat{\ell}_v(S) - \hat{\ell}_v(S^{\setminus i})| \;.$$

Based on the definitions of $\hat{p}_v = c_v/m$ and $\hat{q}_v = c_v^+/c_v$, we can rewrite Eq. (13) as

$$\hat{\ell}_v(S) = \begin{cases} \frac{1}{2m} & c_v = 1 \\ \frac{2c_v^+(c_v - c_v^+)}{m(c_v-1)} & c_v \geq 2. \end{cases}$$

Therefore, if $c_v \geq 3$,

$$\begin{aligned} |\hat{\ell}_v(S) - \hat{\ell}_v(S^{\setminus i})| &= \frac{2c_v^+}{m}\left(\frac{c_v - c_v^+}{c_v - 1} - \frac{c_v - c_v^+ - 1}{c_v - 2}\right) = \frac{2c_v^+(c_v^+ - 1)}{m(c_v-1)(c_v-2)} \\ &\leq \frac{2c_v(c_v-1)}{m(c_v-1)(c_v-2)} = \frac{2c_v}{m(c_v-2)} \leq \frac{6}{m} \ , \end{aligned}$$

while if $c_v = 2$ then

$$|\hat{\ell}_v(S) - \hat{\ell}_v(S^{\setminus i})| = \frac{2c_v^+(2-c_v^+)}{m} - \frac{1}{2m} \leq \frac{2}{m} \ .$$

Lastly, if $c_v = 1$ then $|\hat{\ell}_v(S) - \hat{\ell}_v(S^{\setminus i})| = \frac{1}{2m}$. Therefore for any sample S

$$|\hat{\ell}_v(S) - \hat{\ell}_v(S^{\setminus i})| \leq \frac{6}{m} \ ,$$

and thus the effect of a single change in S is no larger than $\frac{12}{m}$. We can now apply McDiarmid's theorem to get that with probability of at least $1-\delta$:

$$|\hat{\ell} - \mathbb{E}[\hat{\ell}]| \leq \sqrt{\frac{1}{2}\ln\left(\frac{2}{\delta}\right) m(\frac{12}{m})^2} = 12\sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2m}}.$$

■

We now turn to show a concentration bound on the true generalization error $\ell(h_S^{\text{Gini}})$. Here we cannot directly use McDiarmid's Theorem since the bounded differences property does not hold for $\ell(h_S^{\text{Gini}})$. To see this, suppose that $V = \{0,1\}$, $p_0 = p_1 = \frac{1}{2}$, $q_0 = 0.99$ and $q_1 = 1$. Assume in addition that $|S(0)| = 1$; namely, there is only a single example in S for which the feature takes the value 0, an unlikely but possible scenario. In this case, if the single example in $S(0)$ is labeled 1, then $\ell(h_S^{\text{Gini}}) = 0.01$, but if this example is labeled 0, then $\ell(h_S^{\text{Gini}}) = 0.99$. That is, a change of a single example might have a dramatic effect on $\ell(h_S^{\text{Gini}})$. This problem can intuitively be attributed to the fact that S is an atypical sample of the underlying distribution $\{p_v\}$. To circumvent this obstacle, we use the following lemma. Note that a similar result can be derived from the results in Kutin (2002), albeit with much larger constants. The lemma below provides tighter bounds for a more restricted case.

Lemma 9 *Let S be a sample with m examples drawn i.i.d from the distribution $\Pr[X,Y]$. Let δ be a confidence parameter. For two samples S_1 and S_2 with m examples, we say that $d(S_1,S_2) \leq 1$ if there is at most one example that is different between the two samples. Let f be a real function of the sample. If there exists a function of the sample g and real numbers c, b such that the following*

conditions hold:

$$\forall S_1, S_2 \text{ s.t. } d(S_1,S_2) \leq 1 \quad |g(S_1) - g(S_2)| \leq \frac{c}{m} \tag{21}$$

$$\forall^{\delta} S \quad f(S) = g(S) \tag{22}$$

$$|\mathbb{E}[f(S)] - \mathbb{E}[g(S)]| \leq \frac{b}{\sqrt{m}} \ , \tag{23}$$

then

$$\forall^{2\delta} S \quad |f(S) - \mathbb{E}[f(S)]| \leq \frac{c\sqrt{\ln(\frac{2}{\delta})} + b\sqrt{2}}{\sqrt{2m}} \ .$$

Proof From Eq. (21) and McDiarmid's theorem we have

$$\forall^{\delta} S \quad |g(S) - \mathbb{E}[g(S)]| \leq \frac{c\sqrt{\ln(\frac{2}{\delta})}}{\sqrt{2m}}.$$

In addition,

$$|f(S) - \mathbb{E}[f(S)]| \leq |f(S) - g(S)| + |g(S) - \mathbb{E}[g(S)]| + |\mathbb{E}[f(S)] - \mathbb{E}[g(S)]| \ .$$

Therefore, using Eq. (22) and Eq. (23) and applying a union bound, we have

$$\forall^{2\delta} S \, |f(S) - \mathbb{E}[f(S)]| \leq 0 + \frac{c\sqrt{\ln(\frac{2}{\delta})}}{\sqrt{2m}} + \frac{b}{\sqrt{m}} = \frac{c\sqrt{\ln(\frac{2}{\delta})} + b\sqrt{2}}{\sqrt{2m}}.$$

■

To use Lemma 9 we define a new hypothesis h_S^{δ} that depends both on the sample S and on the desired confidence parameter δ. This hypothesis would 'compensate' for atypical samples. We let $f \triangleq \ell(h_S^{\text{Gini}})$ and $g \triangleq \ell(h_S^{\delta})$, and show that the conditions of the lemma hold.

We construct a hypothesis h_S^{δ} such that g satisfies the three requirements given in Eqs. (21-23) based on Lemma 10 below. This lemma states that except for values with small probabilities, we can assure that with high confidence, $c_v(S)$ grows with p_v. This means that as long as p_v is not too small, a change of a single example in $c_v(S)$ does not change $h_S^{\delta}(v)$ too much. On the other hand, if p_v is small then the value v has little effect on the error to begin with. Therefore, regardless of the probability p_v, the error $\ell(h_S^{\delta})$ cannot be changed too much by a single change of example in S. This would allow us to prove a concentration bound on $\ell(h_S^{\delta})$ using McDiardmid's theorem. Let us first introduce a new notation. Given a confidence parameter $\delta > 0$, a probability $p \in [0,1]$, and a sample size m, we define

$$\rho(\delta, p, m) \ \triangleq \ mp - \sqrt{mp \cdot 3\ln(2/\delta)}.$$

Lemma 10 below states that $c_v(S)$ is likely to be at least $\rho(\delta/m, p_v, m)$ for all values with non-negligible probabilities.

Lemma 10 *Let $\delta \in (0,1)$ be a confidence parameter. Then,*

$$\forall^{\delta} S \quad \forall v \in V: \ p_v \geq \frac{6\ln(\frac{2m}{\delta})}{m} \quad \Rightarrow \quad c_v(S) \geq \rho(\delta/m, p_v, m) > 1.$$

Proof The proof is based on lemma 44 from Drukh and Mansour (2005). This lemma states that for all $v \in V$ such that $p_v \geq \frac{3\ln(\frac{2}{\delta})}{m}$ we have that

$$\forall^{\delta} S \quad |p_v - \hat{p}_v| \leq \sqrt{\frac{p_v \cdot 3\ln(\frac{2}{\delta})}{m}}. \tag{24}$$

Based on this lemma, we immediately get that for all v such that $p_v \geq 3\ln(\frac{2}{\delta})/m$,

$$\forall^{\delta} S \quad c_v \geq \rho(\delta, p_v, m).$$

Note, however, that this bound is trivial for $p_v = 3\ln(\frac{2}{\delta})/m$, because in this case $\rho(\delta, p_v, m) = 0$. We therefore use the bound for values in which $p_v \geq 6\ln(\frac{2}{\delta})/m$. For these values it is easy to show that $\rho(\delta, p_v, m) > 1$ for any $\delta \in (0,1)$. Trivially, there are at most m values v for which $p_v \geq \frac{6\ln(2/\delta)}{m}$. Therefore, by substituting δ/m for δ and applying a union bound, the proof is concluded. ■

Based on the bound given in the above lemma, we define h_S^{δ} to be

$$h_S^{\delta}(v) \triangleq \begin{cases} h_S^{\text{Gini}}(v) & p_v < \frac{6\ln(\frac{2m}{\delta})}{m} \text{ or } c_v \geq \rho(\frac{\delta}{m}, p_v, m) \\ \frac{c_v^+ + q_v(\lceil \rho(\frac{\delta}{m}, p_v, m) \rceil - c_v)}{\lceil \rho(\frac{\delta}{m}, p_v, m) \rceil} & \text{otherwise.} \end{cases}$$

That is, $h_S^{\delta}(v)$ is equal to $h_S^{\text{Gini}}(v)$ if either p_v is negligible or if there are enough representatives of v in the sample. If this is not the case, then S is not a typical sample and thus we "force" it to be typical by adding $\lceil \rho(\frac{\delta}{m}, p_v, m) \rceil - c_v$ 'pseudo-examples' to S with the value v and with labels that are distributed according to q_v. Therefore, except for values with negligible probability p_v, the hypothesis $h_S^{\delta}(v)$ is determined by at least $\lceil \rho(\frac{\delta}{m}, p_v, m) \rceil$ 'examples'. As a direct result of this construction we obtain that a single example from S has a small effect on the value of $\ell(h_S^{\delta})$.

We can now show that each of the properties in (21-23) hold. From the definition of h_S^{δ} and Lemma 10 it is clear that Eq. (22) holds. Let us now show that Eq. (23) holds, with b.

Lemma 11 $\left|\mathbb{E}[\ell(h_S^{Gini})] - \mathbb{E}[\ell(h_S^{\delta})]\right| \leq \frac{1}{m}$.

Proof We have

$$\mathbb{E}[\ell(h_S^{\text{Gini}})] - \mathbb{E}[\ell(h_S^{\delta})] = \sum_v \left(\mathbb{E}[\ell_v(h_S^{\text{Gini}}) - \ell_v(h_S^{\delta})] \right). \tag{25}$$

We bound $\mathbb{E}[\ell_v(h_S^{\text{Gini}}) - \ell_v(h_S^{\delta})]$ as follows. First, for values v such that $p_v < 6\ln(\frac{2m}{\delta})/m$, we have that $h_S^{\text{Gini}}(v) = h_S^{\delta}(v)$. Thus $\mathbb{E}[\ell_v(h_S^{\text{Gini}}) - \ell_v(h_S^{\delta})] = 0$. For the rest of the values, $p_v \geq 6\ln(\frac{2m}{\delta})/m$ and thus the definition of $\ell_v(h_S^{\delta})$ implies

$$\begin{aligned} &\mathbb{E}[\ell_v(h_S^{\text{Gini}}) - \ell_v(h_S^{\delta})] = \\ &\quad \Pr[c_v < \rho(\delta/m, p_v, m)] \cdot \mathbb{E}\left[\ell_v(h_S^{\text{Gini}}) - \ell_v(h_S^{\delta}) \mid c_v < \rho(\delta/m, p_v, m) \right] . \end{aligned} \tag{26}$$

Using Eq. (24) again, we obtain that $\Pr[c_v < \rho(\delta/m, p_v, m)] \leq \delta/m$. In addition, since both $\ell_v(h_S^{\text{Gini}})$ and $\ell_v(h_S^{\delta})$ are in $[0, p_v]$ we have that

$$\left| \mathbb{E}\left[\ell_v(h_S^{\text{Gini}}) - \ell_v(h_S^{\delta}) \mid c_v < \rho(\delta/m, p_v, m) \right] \right| \leq p_v.$$

Combining the above two facts with Eq. (26) we get

$$\left|\mathbb{E}[\ell_v(h_S^{\text{Gini}}) - \ell_v(h_S^{\delta})]\right| \leq \frac{p_v \delta}{m} \leq \frac{p_v}{m}.$$

Summing the above over v and using Eq. (25) we conclude that,

$$\left|\mathbb{E}[\ell(h_S^{\text{Gini}}) - \ell(h_S^{\delta})]\right| \leq \sum_v \frac{p_v}{m} = \frac{1}{m}.$$

■

Finally, the following lemma shows that Eq. (21) also holds.

Lemma 12 *For any $\delta > 0$, and for any two samples S_1 and S_2 with m examples such that $d(S_1, S_2) \leq 1$ with d defined as in Lemma 9,*

$$\left|\ell(h_{S_1}^{\delta}) - \ell(h_{S_2}^{\delta})\right| \leq \frac{12 \ln(\frac{2m}{\delta})}{m}.$$

The proof of this lemma is deferred to the appendix.

We have shown that the functions $g \stackrel{\Delta}{=} \ell(h_S^{\delta})$ and $f \stackrel{\Delta}{=} \ell(h_S^{\text{Gini}})$ satisfy the three requirements given in Eqs. (21-23) and therefore Lemma 9 can be used to show that $\ell(h^{\text{Gini}})$ is concentrated.

Lemma 13 $\forall \delta > 0 \quad \forall^{\delta} S \quad \left|\ell(h_S^{Gini}) - \mathbb{E}[\ell(h_S^{Gini})]\right| \leq \frac{12 \ln(\frac{4m}{\delta}) \sqrt{\ln(\frac{4}{\delta})}}{\sqrt{2m}} + \frac{1}{m}$.

Proof In Lemma 9, let $f \stackrel{\Delta}{=} \ell(h_S^{\text{Gini}})$ and let $g \stackrel{\Delta}{=} \ell(h_S^{\delta})$. Let $c \stackrel{\Delta}{=} 12 \ln(\frac{2m}{\delta})$, and let $b \stackrel{\Delta}{=} \frac{1}{\sqrt{m}}$. By Lemma 10, Eq. (22) holds. By Lemma 12, Eq. (21) holds, and by Lemma 11, Eq. (23) holds. Therefore, from Lemma 9 we have

$$\forall \delta > 0 \quad \forall^{2\delta} S \quad |f(S) - \mathbb{E}[f(S)]| \leq \frac{12 \ln(\frac{2m}{\delta}) \sqrt{\ln(\frac{2}{\delta})}}{\sqrt{2m}} + \frac{1}{m} \ .$$

The proof is concluded by substituting $\frac{\delta}{2}$ for δ. ■

Thm. 2 states that with high confidence, the estimator $\hat{\ell}$ is close to the true generalization error of the Gini hypothesis, $\ell(h_S^{\text{Gini}})$. We conclude the analysis of the Gini estimator by proving this theorem.

Proof [of Thm. 2] Substituting $\frac{\delta}{2}$ for δ and applying a union bound, we have that all three properties stated in Lemma 13, Thm. 1 and Lemma 8 hold with probability of at least $1 - \delta$. We therefore conclude that with probability of at least $1 - \delta$,

$$\begin{aligned}\left|\ell(h_S^{\text{Gini}}) - \hat{\ell}\right| &\leq |\ell(h_S^{\text{Gini}}) - \mathbb{E}[\ell(h_S^{\text{Gini}})]| + \left|\mathbb{E}[\ell(h_S^{\text{Gini}})] - \mathbb{E}[\hat{\ell}]\right| + \left|\mathbb{E}[\hat{\ell}] - \hat{\ell}\right| \\ &\leq \frac{2}{m} + \frac{12 \ln\left(\frac{8m}{\delta}\right) \sqrt{\ln\left(\frac{8}{\delta}\right)}}{\sqrt{2m}} + 12\sqrt{\frac{\ln(\frac{4}{\delta})}{2m}} \ = \ O\left(\frac{\ln(\frac{m}{\delta}) \sqrt{\ln(\frac{1}{\delta})}}{\sqrt{m}}\right) \ .\end{aligned}$$

■

4.3 Proof of Thm. 3

Throughout this section we use the notation $S^{(m)}$ to denote a random training set of m examples. Before proving Thm. 3, we provide the following lemma, that shows that the expectation of M_k converges to 0 for any k.

Lemma 14 *For any natural k and a countable V,*

$$\lim_{m\to\infty} \mathbb{E}[M_k(S^{(m)})] = 0$$

Proof Following McAllester and Schapire (2000) we have that for any m

$$\mathbb{E}[M_k(S^{(m)})] = \sum_{v\in V} p_v \Pr[|S_v^{(m)}| = k] \ .$$

Since V is a countable set we can rewrite it as $V \stackrel{\Delta}{=} \{v_1, v_2, v_3, \ldots\}$. Let $\varepsilon > 0$, and let N be a positive integer such that $\sum_{i=1}^{N} p_{v_i} > 1 - \frac{\varepsilon}{2}$. Since $\lim_{m\to\infty} \left(\Pr[|S_v^{(m)}| = k]\right) = 0$ for any natural k, there exists an m' such that for any $m > m'$, $\sum_{i=1}^{N} p_{v_i} \Pr[|S_{v_i}^{(m)}| = k] < \frac{\varepsilon}{2}$. In addition, $\sum_{i=N+1}^{|V|} p_{v_i} < \frac{\varepsilon}{2}$. Hence, for every $m > m'$,

$$\mathbb{E}[M_k(S^{(m)})] = \sum_{i=1}^{N} p_{v_i} \Pr[|S_{v_i}^{(m)}| = k] + \sum_{i=N+1}^{|V|} p_{v_i} \Pr[|S_{v_i}^{(m)}| = k] < \varepsilon.$$

■

Proof [of Thm. 3] To prove Eq. (9), we calculate the expectation of the generalization error $\mathbb{E}[\ell(h_S)]$ of an arbitrary hypothesis mapping $h \in H$ and show that this error is minimized when $h[S] = h_S^{\text{Bayes}}$. Let $f_h : \mathbb{N}\times\mathbb{N} \to [0,1]$ be a function such that $f_h(n_1,n_2) = 1 - f_h(n_1, n_1 - n_2)$ and let h be a hypothesis mapping such that for all $v \in V$, $h[S](v) = f_h(c_v(S), c_v^+(S))$. Then,

$$\begin{aligned}\mathbb{E}[\ell(h[S])] &= \sum_v p_v \mathbb{E}[q_v(1 - f_h(c_v(S), c_v^+(S))) + (1-q_v) f_h(c_v(S), c_v^+(S))] \\ &= \sum_v p_v (q_v + (1-2q_v))\, \mathbb{E}[f_h(c_v(S), c_v^+(S))].\end{aligned}$$

From the above expression it is clear that if $q_v < \frac{1}{2}$ then $\mathbb{E}[\ell(h[S])]$ is minimal when $\mathbb{E}[f_h(c_v(S), c_v^+(S))]$ is minimal, and if $q_v > \frac{1}{2}$ then $\mathbb{E}[\ell(h[S])]$ is minimal when $\mathbb{E}[f_h(c_v(S), c_v^+(S))]$ is maximal. If $q_v = \frac{1}{2}$ the expectation equals $\frac{1}{2}$ regardless of the choice of f_h. We have

$$\begin{aligned}\mathbb{E}[f_h(c_v(S), c_v^+(S))] &= \sum_S \Pr[S]\, f_h(c_v(S), c_v^+(S)) \\ &= \sum_{k=0}^{m} \Pr[c_v(S) = k] \sum_{i=0}^{k} \Pr[c_v^+(S) = i \mid c_v(S) = k]\, f_h(k,i)\end{aligned}$$

Consider the summation on i for a single k from the above sum. If k is odd, then

$$\begin{aligned}
&\sum_{i=0}^{k} \Pr[c_v^+ = i \mid c_v = k] f_h(k,i) \\
&= \sum_{i=0}^{\frac{k-1}{2}} \Pr[c_v^+ = i \mid c_v = k] f_h(k,i) + \sum_{i=\frac{k+1}{2}}^{k} P[c_v^+ = i \mid c_v = k](1 - f_h(k,k-i)) \\
&= \sum_{i=0}^{\frac{k-1}{2}} \Pr[c_v^+ = i \mid c_v = k] f_h(k,i) + \sum_{i=0}^{\frac{k-1}{2}} \Pr[c_v^+ = k-i \mid c_v = k](1 - f_h(k,i)) \\
&= C + \sum_{i=0}^{\frac{k-1}{2}} \left(\Pr[c_v^+ = i \mid c_v = k] - \Pr[c_v^+ = k-i \mid c_v = k] \right) f_h(k,i)
\end{aligned}$$

where C is a constant that does not depend on f_h. In the above expression, note that if $q_v < \frac{1}{2}$ then for each $i \leq \frac{k-1}{2}$, $\Pr[c_v^+ = i \mid c_v = k] - \Pr[c_v^+ = k-i \mid c_v = k]$ is positive, and that if $q_v > \frac{1}{2}$ then this expression is negative. This means that in both cases, to minimize $\mathbb{E}[\ell(h_S)]$, we need to maximize $f_h(k,i)$ for $i \leq \frac{k-1}{2}$. For an even k the analysis is similar, except that we have the special case of $i = \frac{k}{2}$ that does not pair with another summand. However, from the symmetry constraint on f_h it follows that $f_h(k, \frac{k}{2}) = \frac{1}{2}$. Therefore no maximization or minimization is allowed for this value of i. Based on the above analysis, the function f_h that minimizes $\mathbb{E}[\ell(h_S)]$ is:

$$f_h(n_1,n_2) = \begin{cases} 1 & n_2 \leq \frac{n_1-1}{2} \\ 0 & n_2 \geq \frac{n_1+1}{2} \\ \frac{1}{2} & n_2 = \frac{n_1}{2} \end{cases}$$

Setting $h_S(v) = f_h(c_v(S), c_v^+(S))$ we have that $h_S(v) = h_S^{\text{Bayes}}(v)$ for all values v in V.

To prove Eq. (10), we first calculate the difference between $\ell_v(h_\infty^{\text{Bayes}})$ and the expectation of $\ell_v(h_S^{\text{Bayes}})$. Assume without loss of generality that $q_v > \frac{1}{2}$. Then $\ell_v(h_\infty^{\text{Bayes}}) = p_v(1-q_v)$, and

$$\mathbb{E}[\ell_v(h_S^{\text{Bayes}})] = p_v(q_v \Pr[\hat{q}_v < \frac{1}{2}] + (1-q_v)(1 - \Pr[\hat{q}_v < \frac{1}{2}]) + \frac{1}{2}\Pr[\hat{q}_v = \frac{1}{2}]).$$

Subtracting, we have

$$\begin{aligned}
\mathbb{E}[\ell_v(h_S^{\text{Bayes}})] - \ell_v(h_\infty^{\text{Bayes}}) &= p_v(2q_v - 1)(\Pr[\hat{q}_v < \frac{1}{2}] + \frac{1}{2}\Pr[\hat{q}_v = \frac{1}{2}]) \\
&\leq p_v(2q_v-1)\Pr[c_v = 0] \cdot \frac{1}{2} + p_v \sum_{k=1}^{m} \Pr[c_v = k](2q_v - 1)\Pr[\hat{q}_v \leq \frac{1}{2} | c_v = k].
\end{aligned}$$

We use Lemma 17 below to bound $(2q_v - 1)\Pr[\hat{q}_v \leq \frac{1}{2} | c_v = k]$ for $k \geq 3$. For $k = 0, 1, 2$ we maximize this term individually for each k. This leads us to the following bound:

$$\begin{aligned}
&\mathbb{E}[\ell_v(h_S^{\text{Bayes}})] - \ell_v(h_\infty^{\text{Bayes}}) \\
&\leq \frac{1}{2} p_v \Pr[c_v = 0] + \frac{1}{8} p_v \Pr[c_v = 1] + \frac{1}{8} p_v \Pr[c_v = 2] + \sum_{k=3}^{m} \frac{1}{\sqrt{ek}} p_v \Pr[c_v = k].
\end{aligned}$$

Recall that M_k is the probability mass of the values seen k times in the sample. Following McAllester and Schapire (2000) we have that for $k \geq 0$, $\mathbb{E}[M_k] = \sum_v p_v \Pr[c_v = k]$. Hence, summing over all the values v, we have

$$\begin{aligned}\mathbb{E}[\ell(h_S^{\text{Bayes}})] - \ell(h_\infty^{\text{Bayes}}) &= \sum_v (\mathbb{E}[\ell_v(h_S^{\text{Bayes}})] - \ell_v(h_\infty^{\text{Bayes}})) \\ &\leq \frac{1}{2}\mathbb{E}[M_0] + \frac{1}{8}\mathbb{E}[M_1] + \frac{1}{8}\mathbb{E}[M_2] + \sum_{k=3}^{m} \frac{1}{\sqrt{ek}}\mathbb{E}[M_k].\end{aligned}$$

To prove Eq. (11), denote by $S^{(m)}$ a sample of m examples. Let $\varepsilon > 0$ be a scalar. Then there exists an integer t such that $\frac{1}{\sqrt{et}} < \frac{\varepsilon}{2}$. Since $\sum_{k=1}^{m} \mathbb{E}[M_k(S^{(m)})] = 1$, we have

$$\sum_{k=t}^{m} \frac{1}{\sqrt{ek}}\mathbb{E}[M_k(S^{(m)})] < \frac{\varepsilon}{2}. \tag{27}$$

Now, by Lemma 14, for every $k < t$, $\lim_{m\to\infty} \mathbb{E}[M_k(S^{(m)})] = 0$. Hence, there exists an m' such that for every $m > m'$,

$$\frac{1}{2}\mathbb{E}[M_0(S^{(m)})] + \frac{1}{8}\mathbb{E}[M_1(S^{(m)})] + \frac{1}{8}\mathbb{E}[M_2(S^{(m)})] + \sum_{k=3}^{t} \frac{1}{\sqrt{ek}}\mathbb{E}[M_k(S^{(m)})] < \frac{\varepsilon}{2}. \tag{28}$$

Combining Eq. (27) and Eq. (28), we have that for every $m > m'$,

$$\frac{1}{2}\mathbb{E}[M_0] + \frac{1}{8}\mathbb{E}[M_1] + \frac{1}{8}\mathbb{E}[M_2] + \sum_{k=3}^{m} \frac{1}{\sqrt{ek}}\mathbb{E}[M_k] < \varepsilon.$$

Hence the limit of this expression when $m \to \infty$ is 0. ■

4.4 Proof of Thm. 4

To prove Thm. 4, we first introduce some additional notation. Let $\delta \in (0,1)$ be a confidence parameter. Let V_1^δ, V_2^δ, and V_3^δ be three sets that partition V according to the values of the probabilities p_v:

$$\begin{aligned}V_1^\delta &= \{v \mid p_v \leq 6\ln\left(\frac{2m}{\delta}\right) m^{-\frac{2}{3}}\} \\ V_2^\delta &= \{v \mid 6\ln\left(\frac{2m}{\delta}\right) m^{-\frac{2}{3}} < p_v \leq 6\ln\left(\frac{2m}{\delta}\right) m^{-\frac{1}{2}}\} \\ V_3^\delta &= \{v \mid 6\ln\left(\frac{2m}{\delta}\right) m^{-\frac{1}{2}} < p_v\}\end{aligned}$$

We denote the contribution of each set to $\ell(h_S^{\text{Bayes}})$ by $\ell_i^\delta(S) \stackrel{\Delta}{=} \sum_{v\in V_i^\delta} \ell_v(h_S^{\text{Bayes}})$. Additionally, given two samples S and S', let $\kappa(S,S')$ be the predicate that gets the value "true" if for all $v \in V$ we have $c_v(S) = c_v(S')$.

Using the above definitions and the triangle inequality, we can bound $|\ell(h_S^{\text{Bayes}}) - \mathbb{E}[\ell(h_S^{\text{Bayes}})]|$ as follows:

$$\begin{aligned}
|\ell(h_S^{\text{Bayes}}) - \mathbb{E}[\ell(h_S^{\text{Bayes}})]| &= \left|\sum_{i=1}^{3}\left(\ell_i^\delta(S) - \mathbb{E}[\ell_i^\delta]\right)\right| \\
&\leq \left|\,\ell_1^\delta(S) - \mathbb{E}[\ell_1^\delta]\,\right| + \left|\,\ell_2^\delta(S) - \mathbb{E}[\ell_2^\delta(S') \mid \kappa(S,S')]\,\right| + \\
&\quad \left|\,\ell_3^\delta(S) - \mathbb{E}[\ell_3^\delta(S') \mid \kappa(S,S')]\,\right| + \left|\,\mathbb{E}[\ell_2^\delta(S') + \ell_3^\delta(S') \mid \kappa(S,S')] - \mathbb{E}[\ell_2^\delta + \ell_3^\delta]\,\right|.
\end{aligned}$$

To prove Thm. 4 we bound each of the above terms as follows: First, to bound $\left|\,\ell_1^\delta(S) - \mathbb{E}[\ell_1^\delta]\,\right|$ (Lemma 15 below), we use the fact that for each $v \in V_1^\delta$ the probability p_v is small. Thus, a single change of an example in S has a moderate effect on the error and we can use McDiarmid's theorem. To bound $\left|\,\ell_2^\delta(S) - \mathbb{E}[\ell_2^\delta(S') \mid \kappa(S,S')]\,\right|$ (Lemma 16 below) we note that the expectation is taken with respect to those samples S' in which $c_v(S') = c_v(S)$ for all v. Therefore, the variables $\ell_v(h_S^{\text{Bayes}})$ are independent. We show in addition that each of these variables is bounded in $[0, p_v]$ and thus we can apply Hoeffding's bound. Next, to bound $\left|\,\ell_3^\delta(S) - \mathbb{E}[\ell_3^\delta(S') \mid \kappa(S,S')]\,\right|$ (Lemma 19 below), we use the fact that in a typical sample, $c_v(S)$ is large for all $v \in V_3^\delta$. Thus, we bound the difference between $\ell_v(h_S^{\text{Bayes}})$ and $\mathbb{E}[\ell_v(S') \mid \kappa(S,S')]$ for each value in V_3^δ separately. Then, we apply a union bound to show that for all of these values the above difference is small. Finally, we use the same technique to bound $\left|\,\mathbb{E}[\ell_2^\delta(S') + \ell_3^\delta(S') \mid \kappa(S,S')] - \mathbb{E}[\ell_2^\delta + \ell_3^\delta]\,\right|$ (Lemma 20 below). The proof of the first lemma, stated below, is omitted.

Lemma 15 $\forall \delta > 0 \quad \forall^\delta S \quad |\ell_1^\delta(S) - \mathbb{E}[\ell_1^\delta]| \leq \frac{12\ln\left(\frac{2m}{\delta}\right)}{m^{1/6}}\sqrt{\frac{1}{2}\ln\left(\frac{2}{\delta}\right)}$.

Proof We prove the lemma using McDiarmid's theorem. To do so, we examine the effect a removal of a single example (x_i, y_i) from S can have on $\ell_1^\delta(h_S^{\text{Bayes}})$. The largest effect occurs if $x_i \in V_1^\delta$ and the removal of y_i changes the value of $h^{\text{Bayes}}(x_i)$. In this case,

$$|\ell_1^\delta(S) - \ell_1^\delta(S^{\setminus i})| = |\ell_{x_i}(h_S^{\text{Bayes}}) - \ell_{x_i}(h_{S^{\setminus i}}^{\text{Bayes}})| \leq p_v \leq 6\ln\left(\frac{2m}{\delta}\right)m^{-\frac{2}{3}}.$$

Applying McDiarmid's theorem, it follows that $|\ell_1^\delta(S) - \mathbb{E}[\ell_1^\delta]|$ is at most

$$\sqrt{\frac{1}{2}\ln\left(\frac{2}{\delta}\right)m\cdot\left(12\ln\left(\frac{2m}{\delta}\right)m^{-\frac{2}{3}}\right)^2} = \frac{12\ln\left(\frac{2m}{\delta}\right)}{m^{1/6}}\sqrt{\frac{1}{2}\ln\left(\frac{1}{\delta}\right)}\ .$$

■

Lemma 16 $\forall \delta > 0 \quad \forall^\delta S \quad |\ell_2^\delta(S) - \mathbb{E}[\ell_2^\delta(S') \mid \kappa(S,S')]| \leq \frac{\sqrt{3\ln\left(\frac{2m}{\delta}\right)\ln\left(\frac{2}{\delta}\right)}}{m^{1/4}}$.

Proof Since the expectation is taken over samples S' for which $c_v(S') = c_v(S)$ for each $v \in V$, we get that the value of the random variable $\ell_v(h_S^{\text{Bayes}})$ for each v depends only on the assignment of label for each example. Therefore the random variables $\ell_v(h_S^{\text{Bayes}})$ are all independent of each other when conditioned on $\kappa(S,S')$, and $\ell_2^\delta(S) = \sum_{v\in V_2^\delta}\ell_v(h_S^{\text{Bayes}})$ is a sum of independent random variables. The

expectation of this sum is $\mathbb{E}[\ell_2^\delta(S') \mid \kappa(S,S')]$. In addition, it is trivial to show that $\ell_v(h_S^{\text{Bayes}}) \in [0, p_v]$ for all v. Thus, by Hoeffding's inequality,

$$\Pr[|\ell_2^\delta(S) - \mathbb{E}[\ell_2^\delta(S') \mid \kappa(S,S')]| \geq t] \;\leq\; 2e^{-2t^2/\sum_{v\in V_2^\delta} p_v^2}. \tag{29}$$

Using the fact that for v in V_2^δ, $p_v \leq 6\ln\left(\frac{2m}{\delta}\right)/\sqrt{m}$ we obtain that

$$\sum_{v\in V_2^\delta} p_v^2 \;\leq\; \max_{v\in V_2^\delta}\{p_v\} \cdot \sum_{v\in V_2^\delta} p_v \;\leq\; 6\ln\left(\frac{2m}{\delta}\right)/\sqrt{m}\ .$$

Plugging the above into Eq. (29) we get that

$$\Pr[|\ell_2^\delta(S) - \mathbb{E}[\ell_2^\delta(S') \mid \kappa(S,S')]| \geq t] \;\leq\; 2e^{-2t^2\sqrt{m}/(6\ln\left(\frac{2m}{\delta}\right))}\ .$$

Setting the right-hand side to δ and solving for t, we conclude our proof. ■

So far, we have bounded the terms $\left|\, \ell_1^\delta(S) - \mathbb{E}[\ell_1^\delta] \,\right|$ and $\left|\, \ell_2^\delta(S) - \mathbb{E}[\ell_2^\delta(S') \mid \kappa(S,S')] \,\right|$. In both of these cases, we used the fact that p_v is small for all $v \in V_1^\delta \cup V_2^\delta$. We now turn to bound the term $\left|\, \ell_3^\delta(S) - \mathbb{E}[\ell_3^\delta(S') \mid \kappa(S,S')] \,\right|$. In this case, the probabilities p_v are no longer negligible. Therefore, we use a different technique whereby we analyze the probability of $h_S^{\text{Bayes}}(v)$ to be 'wrong', that is to return the less probable label. Since p_v is no longer small, we expect c_v to be relatively large. The following key lemma bounds the probability of $h_S^{\text{Bayes}}(v)$ to be wrong given that c_v is large. The resulting bound depends on the difference between q_v and $1/2$ and becomes vacuous whenever q_v is close to $1/2$. On the other hand, if q_v is close to $1/2$, the price we pay for a wrong prediction is small. In the second part of this lemma, we balance these two terms and end up with a bound that does not depend on q_v.

Lemma 17 *Let $\bar{Z} = (Z_1, \ldots, Z_k)$ be a sequence of i.i.d. binary random variables such that $\Pr[Z_i = 1] = q$ for all i, and assume that $q \geq \frac{1}{2}$. Then,*

$$\Pr[\sum_i Z_i \leq k/2] \;\leq\; e^{-2(q-\frac{1}{2})^2 k} \quad \textit{and} \quad (2q-1)\Pr[\sum_i Z_i \leq k/2] \;\leq\; \frac{1}{\sqrt{ek}}\ .$$

Proof The first inequality is a direct application of Hoeffding's inequality. Multiplying both sides by $2q-1$ we get that the left-hand side of the second inequality is bounded above by $(2q-1)e^{-2(q-\frac{1}{2})^2 k}$. We now let $x = q - \frac{1}{2}$ and use the inequality $2xe^{-2x^2 k} \leq 1/\sqrt{ek}$, which holds for all $x \geq 0$ and $k > 0$. ■

Based on the above lemma, we now bound $\left|\, \ell_3^\delta(S) - \mathbb{E}[\ell_3^\delta(S') \mid \kappa(S,S')] \,\right|$. First, we show that if $c_v(S)$ is large then $\ell_v(S)$ is likely to be close to the expectation of ℓ_v over samples S' in which $c_v(S) = c_v(S')$. This is equivalent to the claim of the following lemma.

Lemma 18 *Under the same assumptions of Lemma 17. Let $f(\bar{Z})$ be the function*

$$f(\bar{Z}) \;=\; \begin{cases} (1-q) & \text{if } \sum_i Z_i > k/2 \\ q & \text{if } \sum_i Z_i < k/2 \\ \frac{1}{2} & \text{if } \sum_i Z_i = k/2 \end{cases}\ .$$

Then, for all $\delta \in (0, e^{-1/2}]$ we have $\forall^\delta \bar{Z} \quad |f(\bar{Z}) - \mathbb{E}[f]| \;\leq\; \sqrt{\frac{2\ln\left(\frac{1}{\delta}\right)}{ek}}$.

Proof To simplify our notation, denote $\alpha = \Pr[\sum_i Z_i > k/2]$, $\beta = \Pr[\sum_i Z_i < k/2]$, and $\gamma = \Pr[\sum_i Z_i = k/2]$. A straightforward calculation shows that

$$|f(\bar{Z}) - \mathbb{E}[f(\bar{Z})]| \;=\; \begin{cases} (2q-1)\,(\beta+\gamma/2) & \text{with probability } \alpha \\ (2q-1)\,(\alpha+\gamma/2) & \text{with probability } \beta \\ (2q-1)\,(\alpha-\beta) & \text{with probability } \gamma \end{cases} .$$

Using the fact that (α,β,γ) is in the probability simplex we immediately obtain that

$$|f(\bar{z}) - \mathbb{E}[f(\bar{Z})]| \le (2q-1) \ .$$

If $2q-1 \le \sqrt{2\ln\left(\frac{1}{\delta}\right)/k}$ then the bound in the lemma clearly holds. Therefore, from now on we assume that $2q-1 > \sqrt{2\ln\left(\frac{1}{\delta}\right)/k}$. In this case, using the first inequality of Lemma 17 we have that $\beta+\gamma \;\le\; e^{-2(q-\frac{1}{2})^2 k} \;\le\; \delta$. Therefore, $1-\delta < \alpha$, and so with probability of at least $1-\delta$ we have that

$$|f(\bar{Z}) - \mathbb{E}[f(\bar{Z})]| \;=\; (2q-1)\,(\beta+\gamma/2) \;\le\; (2q-1)\,(\beta+\gamma) \ .$$

Applying the second inequality of Lemma 17 on the right-hand side of the above inequality we get that $|f(\bar{Z}) - \mathbb{E}[f(\bar{Z})]| \;\le\; \sqrt{1/ek} \;\le\; \sqrt{2\ln(1/\delta)/ek}$, where the last inequality holds since we assume that $\delta \le e^{-1/2}$. ■

Equipped with the above lemma we are now ready to bound $\left|\ \ell_3^\delta(S) - \mathbb{E}[\ell_3^\delta(S') \mid \kappa(S,S')]\ \right|$.

Lemma 19 *If $m \ge 4$ then* $\quad \forall^{(2\delta)} S \quad |\ell_3^\delta(S) - \mathbb{E}[\ell_3^\delta(S') \mid \kappa(S,S')]| \;\le\; 1/m^{\frac{1}{4}}$.

Proof Recall that $\ell_3^\delta(S) = \sum_{v \in V_3^\delta} \ell_v(S)$. $m \ge 4$, hence $\delta/m \le 1/m \le e^{-1/2}$. Choose $v \in V_3^\delta$ and without loss of generality assume that $q_v \ge 1/2$. Thus, from Lemma 18 and the definition of $\ell_v(S)$ we get that with probability of at least $1-\delta/m$ over the choice of the labels in $S(v)$:

$$|\ell_v(S) - \mathbb{E}[\ell_v(S') \mid \kappa(S,S')]| \;\le\; p_v \sqrt{\frac{2\ln\left(\frac{m}{\delta}\right)}{e \cdot c_v(S)}} \ . \tag{30}$$

By the definition of V_3^δ and Lemma 10, $\forall^\delta S,\ \forall v \in V_3^\delta,\ c_v(S) \ge \rho(\delta/m, p_v, m)$. Using the fact that ρ is monotonically increasing with respect to p_v it is possible to show (see Lemma 21 in the appendix)that $\rho(\delta/m, p_v, m) \ge 2\ln\left(\frac{m}{\delta}\right) m^{1/2}$ for all $v \in V_3^\delta$ for $m \ge 4$. Therefore, if indeed $c_v(S) \ge \rho(\delta/m, p_v, m)$ for any $v \in V_3^\delta$, we have that

$$\sqrt{\frac{2\ln\left(\frac{m}{\delta}\right)}{e \cdot c_v(S)}} \le p_v\, m^{-1/4}.$$

Using a union bound to make sure that this condition holds and Eq. (30) holds for all $v \in V_3^\delta$ simultaneously, we obtain that $\forall^{(2\delta)} S \quad \forall v \in V_3^\delta \quad |\ell_v(S) - \mathbb{E}[\ell_v(S') \mid \kappa(S,S')]| \;\le\; p_v\, m^{-1/4}$. Summing over $v \in V_3^\delta$, using the triangle inequality, and using the fact that $\sum_v p_v = 1$ we conclude the proof. ■

Lemma 20 *For $m \geq 8$,*

$$\forall^{\delta} S \quad |\mathbb{E}[\ell_2^{\delta}(S') + \ell_3^{\delta}(S') \mid \kappa(S,S')] - \mathbb{E}[\ell_2^{\delta}(S') + \ell_3^{\delta}(S')]| \leq \frac{1}{m} + \frac{1}{m^{1/6}}.$$

Proof As in the proof of Lemma 19, we use the definitions of V_3^{δ} and V_2^{δ} along with Lemma 10 and Lemma 21 to get that for $m \geq 8$

$$\forall^{\delta} S \quad \forall v \in V_2^{\delta} \cup V_3^{\delta} \quad c_v(S) \geq \rho(\delta/m, p_v, m) \geq 3\ln(m/\delta)m^{1/3} \ . \tag{31}$$

To bound the difference between the conditional expectation and the unconditional expectation, let us first examine both these quantities for individual values v. To simplify our notation, denote $\alpha_1 = \Pr[\hat{q}_v(S') > 1/2 \mid c_v(S') = c_v(S)]$, $\beta_1 = \Pr[\hat{q}_v(S') < 1/2 \mid c_v(S') = c_v(S)]$, and $\gamma_1 = \Pr[\hat{q}_v(S') = 1/2 \mid c_v(S') = c_v(S)]$. Similarly, denote $\alpha_2 = \Pr[\hat{q}_v(S') > 1/2]$, $\beta_2 = \Pr[\hat{q}_v(S') < 1/2]$, and $\gamma_2 = \Pr[\hat{q}_v(S') = 1/2]$. Using the definition of ℓ_v we have that

$$\mathbb{E}[\ell_v(S') \mid c_v(S) = c_v(S')] = p_v \left((1-q_v)\,\alpha_1 + q\,\beta_1 + \frac{1}{2}\gamma_1 \right) \ .$$

Similarly, for the unconditional expectation:

$$\mathbb{E}[\ell_v(S')] = p_v \left((1-q_v)\,\alpha_2 + q\,\beta_2 + \frac{1}{2}\gamma_2 \right) \ . \tag{32}$$

Subtracting the above two equations and rearranging terms it can be shown that

$$\begin{aligned} \Delta &\stackrel{\Delta}{=} |\mathbb{E}[\ell_v(S') \mid c_v(S) = c_v(S')] - \mathbb{E}[\ell_v(S')]| \\ &= p_v\,(q - \frac{1}{2})\,|\,(\beta_1 + \gamma_1) - (\beta_2 + \gamma_2) + (\gamma_1 - \gamma_2)\,| \ . \end{aligned} \tag{33}$$

Let $Z_1, \ldots, Z_{c_v(S)}$ be an i.i.d. sequence of random variables with $\Pr[Z_i = 1] = q_v$. Then we have $\beta_1 + \gamma_1 = \Pr[\sum_i Z_i \leq c_v(S)/2]$. In addition $c_v(S) \geq \lceil \rho(\delta/m, p_v, m) \rceil \stackrel{\Delta}{=} \rho$. Assume without loss of generality that $q_v \geq 1/2$. Thus we have $\Pr[\sum_{i=1}^{\rho} Z_i \leq \rho/2] \geq \Pr[\sum_{i=1}^{c_v(S)} Z_i \leq c_v(S)/2]$. We clearly have that $0 \leq \beta_1 + \gamma_1 \leq \Pr[\sum_{i=1}^{\rho} Z_i \leq \rho/2]$. We now argue that

$$0 \leq \beta_2 + \gamma_2 \leq \frac{\delta}{m} + \Pr[\sum_{i=1}^{\rho} Z_i \leq \rho/2] \ .$$

The left-hand side inequality is trivial. To prove the right-hand side inequality, we note that

$$\begin{aligned} \beta_2 + \gamma_2 = & \sum_{i=1}^{m} \Pr[c_v(S') = i] \Pr\left[\hat{q}_v(S') \leq \frac{1}{2} \mid c_v(S') = i \right] \\ \leq & \Pr[c_v(S') \leq \rho] + \Pr\left[\hat{q}_v(S') \leq \frac{1}{2} \mid c_v(S') = \rho \right] \\ \leq & \frac{\delta}{m} + \Pr[\sum_{i=1}^{\rho} Z_i \leq \rho/2] \ . \end{aligned}$$

Therefore,

$$|(\beta_1+\gamma_1)-(\beta_2+\gamma_2)| \leq \frac{\delta}{m}+\Pr[\sum_{i=1}^{k} Z_i \leq k/2] \ . \tag{34}$$

Similarly, since $0 \leq \gamma_1 \leq \beta_1+\gamma_1$ and $0 \leq \gamma_2 \leq \beta_2+\gamma_2$ we also have that

$$|\gamma_1-\gamma_2| \leq \frac{\delta}{m}+\Pr[\sum_{i=1}^{\rho} Z_i \leq \rho/2] \ . \tag{35}$$

Combining Eq. (34) and Eq. (35) with Eq. (33) we get that

$$\Delta \ \leq \ p_v(2q-1)\left(\frac{\delta}{m}+\Pr[\sum_{i=1}^{\rho} Z_i \leq \rho/2]\right) \ \leq \ p_v\left(\frac{1}{m}+\frac{1}{\sqrt{e\cdot\rho(\frac{\delta}{m},p_v,m)}}\right) \ ,$$

where the last inequality follows from Lemma 17. Finally, by summing over $v \in V_2^\delta \cup V_3^\delta$ and using Eq. (31) we conclude our proof. ■

5. Experiments

In this section we present experimental results that demonstrate the merits of our feature ranking criterion given in Eq. (7). Throughout this section we compare the following four feature ranking criteria:

1. **IG:** The Information Gain criterion (Quinlan, 1993; de Mantaras, 1991; Mitchell, 1997).
2. **IGR:** The Information Gain Ratio criterion (Quinlan, 1993).
3. **Gini:** The original Gini Index (Breiman et al., 1984), which is given in Eq. (2).
4. **Ginger:** Our modified Gini criterion that aims to minimize the generalization error, given in Eq. (7).

We first present experiments with synthetic data that exemplify the generalization properties of the different criteria. Next, we compare the performance of the different criteria on a natural data set from the UCI repository. Finally, we compare the use of the different ranking criteria for the task of growing a decision tree.

5.1 Synthetic Data

Three synthetic data sets were constructed to exemplify the generalization properties of the different ranking criteria in different scenarios. In all of the synthetic data sets the target label was first sampled according to the probability measure $\Pr[Y=1]=\frac{1}{2}$. Synthetic data set I includes only binary features. The goal of data set I is to show that the Ginger criterion behaves similarly to the Gini criterion on binary features. 11 binary features were constructed. For each $i \in \{0,1,\ldots,10\}$ the ith feature was sampled according to the probability measure $\Pr[X_i=Y|Y]=\frac{1+0.1i}{2}$. Thus, feature X_0 is completely uncorrelated with the label, while feature X_{10} perfectly predicts the label.

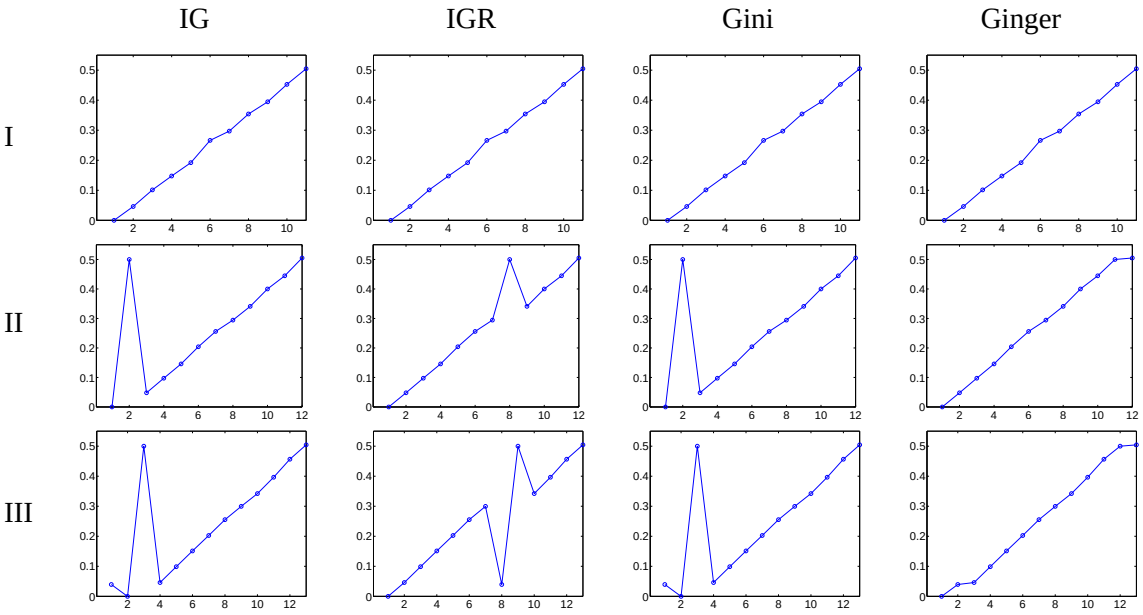


Figure 1: Each of the plots above show the generalization error of each feature (the y axis) against the ranking order of the feature in one of the ranking criteria (the x axis). Each column corresponds to a specific ranking criteria. Each row corresponds to a specific synthetic data set.

A training set of 5000 examples was generated, and the features were ranked using each of the four ranking criteria on the training set. The generalization errors of the 11 classification rules of each feature, defined as in Eq. (5), were measured on a fresh test set of 5000 examples. A plot of the generalization error of each feature against the ranking order of the feature is given for each of the ranking criteria on the top row of Fig. 1. This plot should be monotonically increasing for good feature ranking criteria. As the plots show, all four criteria perform well on this data set.

Data set II is identical to data set I, except that one more feature, indexed X_{11}, was added. X_{11} is simply the index of the example (this simulates an SSN-like feature as described in Sec. 2). Clearly, the generalization error of X_{11} is $\frac{1}{2}$ as no value of the feature that occurred in the training set would occur in a test set. The performance of the four feature ranking criteria on data set II is shown on the second row of Fig. 1. As expected, the Gini criterion and the IG criterion both suffer from overfitting and rank X_{11} very high. The IGR criterion, suggested by Quinlan (1993) attempts to fix the overfitting effect of the IG criterion by dividing IG by the entropy of the feature. As the plots show, this correction indeed causes IGR to rank X_{11} lower than do IG and Gini. However, the correction is not strong enough, as the new feature is still ranked 8th out of 12 features although its generalization error is the worst. Finally, it is clear from the plots that the new Ginger criterion produces a correct ranking of the features in this example.

Data set III is identical to data set II, except that one more feature indexed X_{12} was added. X_{12} was constructed according to the following probability measure:

$$\Pr[X = i \mid Y = 1] = \begin{cases} \frac{1}{2000} & \text{if } i \in \{1,\ldots,2000\} \\ 0 & \text{otherwise} \end{cases} \quad \text{and}$$

$$\Pr[X = i \mid Y = -1] = \begin{cases} \frac{1}{2000} & \text{if } i \in \{2001,\ldots,4000\} \\ 0 & \text{otherwise} \end{cases}$$

X_{12} is thus categorical with many values but it is still highly predictive of the label. The performance of the four feature ranking criteria on data set III is shown on the bottom row of Fig. 1. As the plots show, the rankings of the Gini criterion and of the IG criterion are not adversely affected by the addition of this feature, although they still fail on X_{11}, the SSN-like feature. IGR penalizes X_{12} because it has a large number of values, thus its ranking for this feature is too low. The new Ginger criterion is the only one to rank the features in accordance with their respective generalization error, as is apparent from its monotonically increasing plot.

5.2 Natural Data

To test the ranking criteria on natural data, we used the USCensus1990raw data set from the UCI Repository.[1] This data set contains person records, where each record has 125 features, such as age, salary, marital status etc. Several labeled data sets were constructed from USCensus1990raw by defining a binary target label based on one of the attributes, and using the rest of the attributes as features. For attributes that take more than two values, the binary label was set to 1 if the feature takes its most frequent value and -1 otherwise. Only cases where the probability of the label to be 1 was at least 0.1 and no more than 0.9 were used. This process resulted in 62 binary learning problems.

In Fig. 2, each of the rows corresponds to one learning problem. A plot is shown for each problem and each ranking criterion, depicting the generalization error of each feature against the ranking order of the features. Recall that good ranking criteria should produce monotonically increasing graphs. The plots clearly show that the Ginger criterion produces the most accurate feature ranking. Fig. 3 compares the Ginger criterion to each of the other ranking criteria. In each of the plots, each data point corresponds to one of the 62 learning problems and portrays the difference in generalization error between the feature that was top-ranked by Ginger and the feature that was top-ranked by the other criterion. Positive data points are cases where Ginger outperformed the other criterion. Again, it is apparent that the Ginger criterion outperforms the other criteria.

5.3 Decision Trees

Decision tress are a popular classification tool (see for instance Mitchell, 1997). The process of growing a decision tree is a greedy iterative procedure which is performed as follows: The procedure starts with a tree composed only of a root node. At each iteration, one of the leaves of the tree is turned into an inner node, whose children represent all the possible values of one feature. Choosing

1. The original census data set was used rather than the preprocessed data set. The preprocessed data set obtained from Meek, Thiesson, and Heckerman eliminates categorical attributes that have many values, exactly the type of attributes that this paper addresses. The data set used in our experiments is available through `http://kdd.ics.uci.edu/databases/census1990/USCensus1990raw.data.txt`.

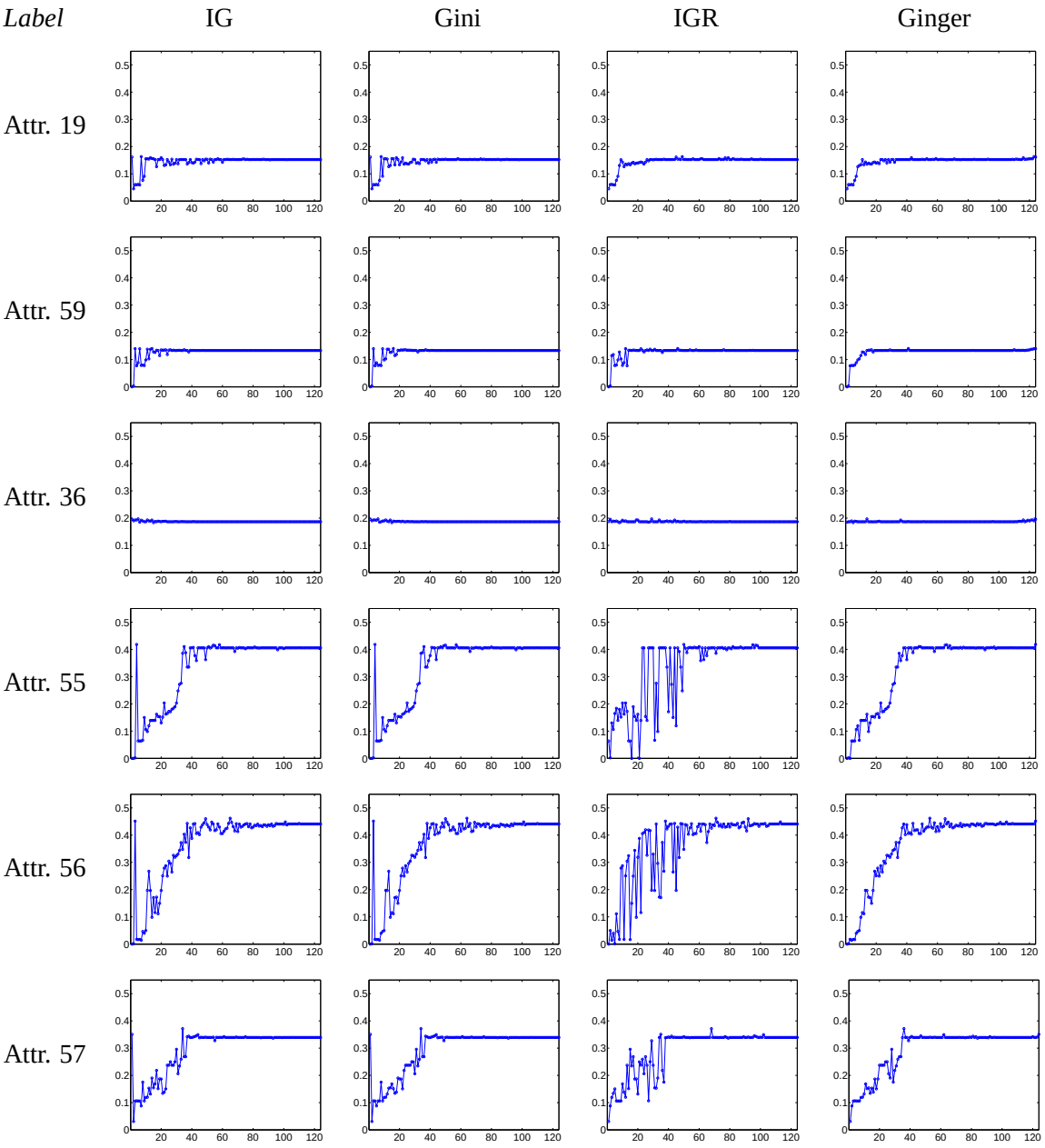


Figure 2: Each of the plots above show the generalization error of the features in a learning problem (the y axis) against the ranking order of the features in one of the ranking criteria (the x axis). Each column corresponds to a specific ranking criterion. Each row corresponds to a specific learning problem, generated from USCensus1990raw by setting the label to be the most common value of one of the attributes.

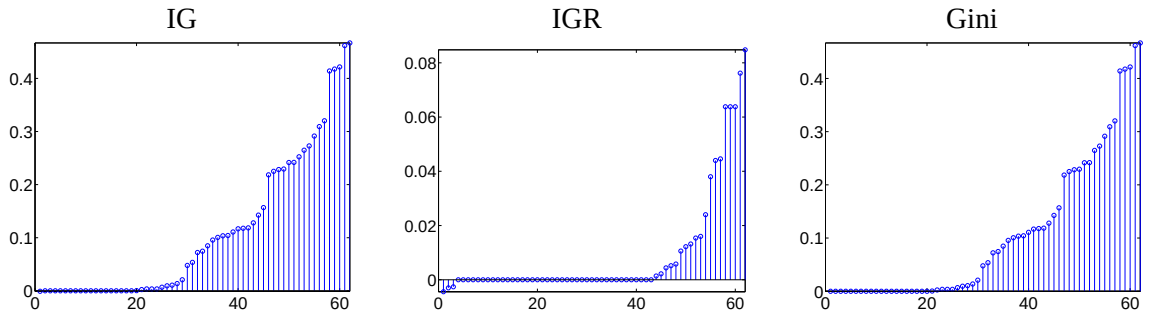


Figure 3: Each plot above portrays the difference in generalization error between the feature that was top-ranked by Ginger and the feature that was top-ranked by one of the other criteria, for each of the 62 learning problems obtained from USCensus1990raw.

which leaf to split and which feature to use for splitting can be based on feature ranking criteria such as the ones discussed in this paper. In our experiments, we compared decision tree learning with each of the four feature ranking criteria: IG, IGR, Gini, and Ginger. The experiments were performed on the 62 learning problems described in Sec. 5.2.

Usually, the iterative process of growing a decision tree continues until no further splits can be made. Then, as a post processing step, the tree is pruned, so as to improve the generalization error of the decision tree. Since this paper focuses on splitting criteria rather then on pruning methods, the experiments do not include tree post-pruning. Instead, the generalization error is measured as a function of the number of splits. Given a ranking criterion, the following procedure is used to choose which leaf to split and which feature to split by: Let m be the number of training examples. A decision tree T with k leaves is equivalent to a mapping $T : \{1,\ldots,m\} \rightarrow \{1,\ldots,k\}$. That is, each example is mapped to one of the leaves of the tree. We can think of the vector $(T(1),\ldots,T(m))$ as the vector of values of a constructed feature. At each iteration of the decision tree learning process, a new tree needs to be generated from the current tree by splitting one of the current tree leaves based on one of the features. Each possible new tree induces a different new constructed feature as described above. To select the leaf to split and the feature to split by, we assess the quality of each new constructed feature based on the ranking criterion in use. The selected leaf and feature are those that correspond to the top-ranked constructed feature.

Fig. 4 shows the training error and generalization error of the Gini, IGR and Ginger splitting criteria as a function of the number of splits, for several learning problems. The IG criterion plot was omitted since its behavior was almost identical to that of the Gini criterion. As can be seen from the plots, the training error of the Gini criterion drops faster, but the resulting tree suffers from severe overfitting. In contrast, the generalization error of the Ginger criterion is much smaller and remains close to the training error, as long as the number of splits is not too large. As expected, after making a large number of splits all criteria exhibit an overfitting effect. Comparing the IGR and the Ginger criteria, we observe that both methods perform rather well, each showing an advantage on some of the learning problems.

Lastly, Fig. 5 compares the performance of the decision tree learning with the Ginger splitting criterion to decision tree learning with the other splitting criteria. In each of the plots, the data points correspond to the 62 learning problems, and portray the difference in the minimal generalization

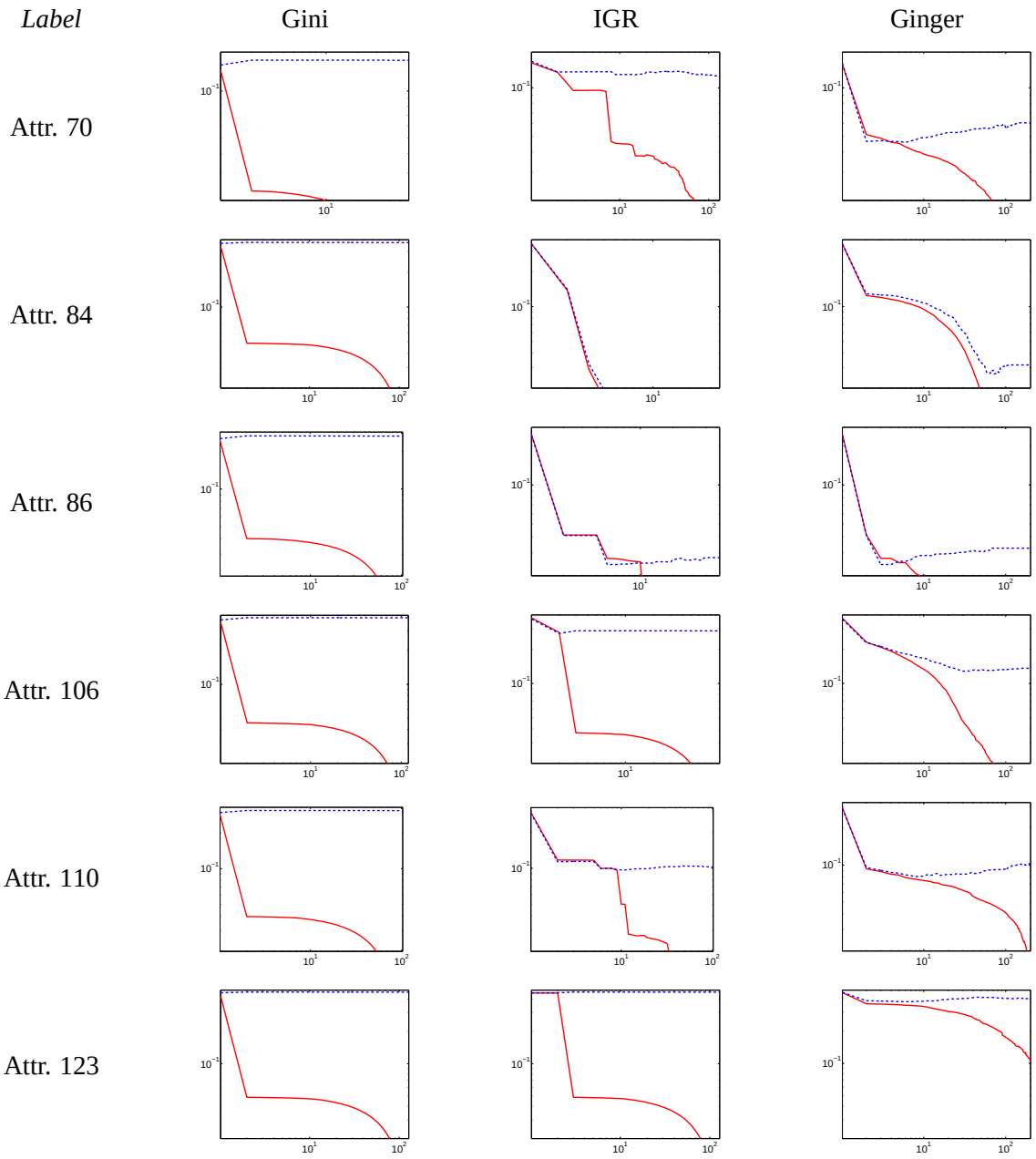


Figure 4: The training error (solid red line) and generalization error (dotted blue line) of decision trees grown according to the Gini, IGR, and Ginger splitting criteria, as a function of the number of splits. Each column corresponds to a specific splitting criterion. Each row corresponds to a specific learning problem, generated from USCensus1990raw by setting the label to be the most common value of one of the attributes.

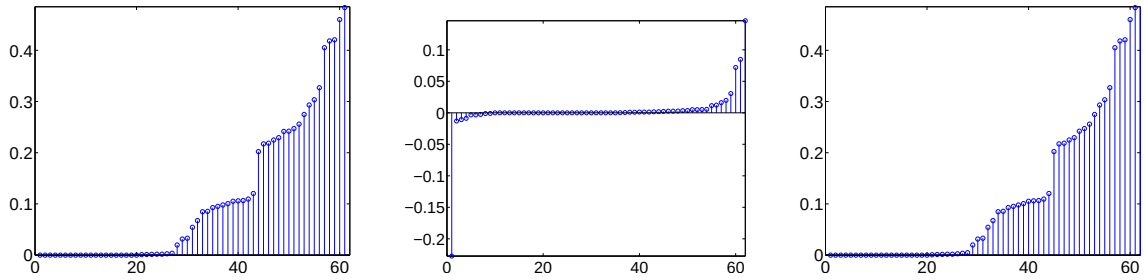


Figure 5: Left: The minimal generalization error of the IG criterion minus the minimal generalization error of the Ginger criterion for each of the labeled data sets. Middle: Same for IGR. Right: Same for Gini.

error achieved by the decision tree grown using Ginger and the one that was achieved using the other criterion. Positive data points are cases where Ginger outperformed the other criterion. The plots show that the Ginger criterion outperforms the IG and Gini criteria, and that in most cases the Ginger criterion outperforms the IGR criterion as well.

6. Discussion

In this paper, a new approach for feature ranking is proposed, based on a direct estimation of the true generalization error of predictors that are deduced from the training set. We focused on two specific predictors, namely h_S^{Gini} and h_S^{Bayes}. An estimator for the generalization error of h_S^{Gini}, termed the Ginger criterion, was proposed and its convergence was analyzed. Experimental evaluation suggests that the Ginger criterion outperforms existing feature ranking methods. We showed that the expected error of h_S^{Bayes} is optimal and proved a concentration bound for this error. Constructing an estimator for h_S^{Bayes} is left for future work.

There are various extensions for this work that we did not pursue. First, it is interesting to analyze the number of categorical features one can rank while avoiding overfitting. The experiments with decision trees suggest that the Ginger criterion has potential to improve the generalization error of decision trees. It may be possible to use the bounds for constructing a stopping criterion for growing the decision tree. Second, our view of a ranking criterion as an estimator for the generalization error of a predictor can be used for constructing new ranking criteria by defining other predictors. Finally, understanding the relationship between this view and information theoretic measures is also an interesting future direction.

Appendix A. Technical Proofs

Lemma 21 *Let c be a positive constant. Then, if $p_v > 6\ln\left(\frac{2}{\delta}\right) m^{-c}$, and $m \geq 2^{\frac{1}{1-c}}$ we have*

$$\forall \delta > 0 \quad \rho(\delta, p_v, m) \geq 3\ln\left(\frac{2}{\delta}\right) m^{1-c}.$$

Proof By the definition of ρ,

$$\rho(\delta, p_v, m) = mp_v - \sqrt{mp_v \cdot 3\ln\left(\frac{2}{\delta}\right)} = \sqrt{mp_v}\left(\sqrt{mp_v} - \sqrt{3\ln\left(\frac{2}{\delta}\right)}\right).$$

Therefore, $\rho(\frac{\delta}{m}, p_v, m)$ is upward monotonic with p_v. Thus if $p_v > 6\ln\left(\frac{2m}{\delta}\right)m^{-c}$,

$$\begin{aligned}
\rho(\delta, p_v, m) &= mp_v - \sqrt{mp_v \cdot 3\ln\left(\frac{2}{\delta}\right)} \\
&\geq 6\ln\left(\frac{2}{\delta}\right)m^{1-c} - \sqrt{6\ln\left(\frac{2}{\delta}\right)m^{1-c} \cdot 3\ln\left(\frac{2}{\delta}\right)} \\
&= 3\ln\left(\frac{2}{\delta}\right)m^{\frac{1-c}{2}}\left(2m^{\frac{1-c}{2}} - \sqrt{2}\right) \\
&= 3\ln\left(\frac{2}{\delta}\right)m^{\frac{1-c}{2}}(m^{\frac{1-c}{2}} + m^{\frac{1-c}{2}} - \sqrt{2}) \\
&\geq 3\ln\left(\frac{2}{\delta}\right)m^{1-c}.
\end{aligned}$$

■

Proof [Lemma 12] Similarly to the proof of Lemma 8, we will bound the effect a single removal of an example from S can have on $\ell(h_S^\delta)$. The maximal effect of a single change in the sample is no larger than twice the maximal effect of a single removal. Assume without loss of generality that the removed example is $x_i = (v, 0)$, and denote the resulting sample by $S^{\setminus i}$. The removal only affects $\ell_v(h_S^\delta)$. Therefore

$$\begin{aligned}
&|\ell(h_S^\delta) - \ell(h_{S^{\setminus i}}^\delta)| = |\ell_v(h_S^\delta) - \ell_v(h_{S^{\setminus i}}^\delta)| \\
&= \left|p_v\left(q_v(1 - h_S^\delta(v)) + (1-q_v)h_S^\delta(v) - p_v q_v(1 - h_{S^{\setminus i}}^\delta(v)) + (1-q_v)h_S^\delta(v)\right)\right| \\
&= \left|p_v(1-2q_v)(h_S^\delta(v) - h_{S^{\setminus i}}^\delta(v))\right| \\
&\leq p_v\left|h_S^\delta(v) - h_{S^{\setminus i}}^\delta(v)\right|.
\end{aligned}$$

For v such that $p_v < \frac{6\ln(\frac{2m}{\delta})}{m}$,

$$|\ell(h_S^\delta) - \ell(h_{S^{\setminus i}}^\delta)| \leq p_v < \frac{6\ln(\frac{2m}{\delta})}{m}. \tag{36}$$

For v such that $p_v \geq \frac{6\ln(\frac{2m}{\delta})}{m}$, we distinguish between three cases by c_v, the number of examples of v in S:

1. $c_v < \rho(\frac{\delta}{m}, p_v, m)$,
2. $\rho(\frac{\delta}{m}, p_v, m) \leq c_v < \rho(\frac{\delta}{m}, p_v, m) + 1$,

3. $\rho(\frac{\delta}{m}, p_v, m) + 1 \leq c_v$.

In case 1,

$$h_S^\delta(v) = \frac{c_v^+ + q_v(\lceil \rho(\frac{\delta}{m}, p_v, m)\rceil - c_v)}{\lceil \rho(\frac{\delta}{m}, p_v, m)\rceil} \text{ and } h_{S^{\setminus i}}^\delta(v) = \frac{c_v^+ + q_v(\lceil \rho(\frac{\delta}{m}, p_v, m)\rceil - (c_v - 1))}{\lceil \rho(\frac{\delta}{m}, p_v, m)\rceil},$$

hence

$$|h_S^\delta(v) - h_{S^{\setminus i}}^\delta(v)| = \frac{q_v}{\lceil \rho(\frac{\delta}{m}, p_v, m)\rceil} .$$

In case 2, $\lceil \rho(\frac{\delta}{m}, p_v, m)\rceil = c_v$, therefore

$$h_S^\delta(v) = h_S^{\text{Gini}}(v) = \frac{c_v^+}{c_v} \text{ and } h_{S^{\setminus i}}^\delta(v) = \frac{c_v^+ + q_v(c_v - (c_v - 1))}{c_v},$$

hence

$$|h_S^\delta(v) - h_{S^{\setminus i}}^\delta(v)| = \frac{q_v}{c_v} = \frac{q_v}{\lceil \rho(\frac{\delta}{m}, p_v, m)\rceil} .$$

In case 3, since $\rho(\frac{\delta}{m}, p_v, m) > 1$ we have $c_v \geq 2$ and

$$h_S^\delta(v) = h_S^{\text{Gini}}(v) = \frac{c_v^+}{c_v} \text{ and } h_{S^{\setminus i}}^\delta(v) = h_{S^{\setminus i}}^\delta(v) = \frac{c_v^+}{c_v - 1}$$

Hence

$$|h_S^\delta(v) - h_{S^{\setminus i}}^\delta(v)| = \frac{c_v^+}{c_v(c_v-1)} \leq \frac{c_v}{c_v(c_v-1)} = \frac{1}{c_v - 1} \leq \frac{1}{\lceil \rho(\frac{\delta}{m}, p_v, m)\rceil} .$$

Therefore, in all cases, for v such that $p_v \geq \frac{6\ln(\frac{2m}{\delta})}{m}$,

$$|\ell(h_S^\delta) - \ell(h_{S^{\setminus i}}^\delta)| \leq p_v \left| h_S^\delta(v) - h_{S^{\setminus i}}^\delta(v) \right| \leq \frac{p_v}{\rho(\frac{\delta}{m}, p_v, m)} = \frac{p_v}{mp_v - \sqrt{mp_v \cdot 3\ln(\frac{2m}{\delta})}}$$

$$= \frac{1}{m} \frac{\sqrt{p_v}}{\sqrt{p_v} - \sqrt{\frac{3\ln(\frac{2m}{\delta})}{m}}} \leq \frac{1}{m} \left(\frac{\sqrt{\frac{6\ln(\frac{2m}{\delta})}{m}}}{\sqrt{\frac{6\ln(\frac{2m}{\delta})}{m}} - \sqrt{\frac{3\ln(\frac{2m}{\delta})}{m}}} \right) = \frac{1}{m} \frac{2}{\sqrt{2}-1} \leq \frac{4}{m} .$$

Combining this with Eq. (36), we have

$$|\ell(h_S^\delta) - \ell(h_{S^{\setminus i}}^\delta)| \leq \max\left\{ \frac{4}{m}, \frac{6\ln(\frac{2m}{\delta})}{m} \right\} = \frac{6\ln(\frac{2m}{\delta})}{m}.$$

Hence, doubling the effect of a single removal, we have that for any two samples S_1 and S_2 such that $d(S_1, S_2) \leq 1$

$$|\ell(h_{S_1}^\delta) - \ell(h_{S_2}^\delta)| \leq \frac{12\ln(\frac{2m}{\delta})}{m} .$$

■

References

A. Antos and I. Kontoyiannis. Convergence properties of functional estimates for discrete distributions. *Random Struct. Algorithms*, 19(3-4):163–193, 2001.

A. Antos, L. Devroye, and L. Gyorfi. Lower bounds for Bayes error estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(7):643–645, 1999.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth & Brooks, 1984.

R. Lopez de Mantaras. A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 6(1):81–92, 1991.

E. Drukh and Y. Mansour. Concentration bounds for unigrams language model. *Journal of Machine Learning Research*, 6:1231–1264, 2005.

I.J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264, 1953.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

M. Kearns and Y. Mansour. On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 459–468, 1996.

S. Kutin. Extensions to mcdiarmid's inequality when differences are bounded with high probability. Technical report, University of Chicago TR-2002-04, 2002.

D.A. McAllester and R.E. Schapire. On the convergence rate of good-turing estimators. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 1–6, 2000.

C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, pages 148–188, 1989.

J. Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3:319–342, 1989.

T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

K. Torkkola. Information theoretic methods. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature Extraction, Foundations and Applications*. Springer, 2006.

L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2004.

Journal of Machine Learning Research 8 (2008) 1115-1146 Submitted 9/07; Revised 2/08; Published 6/08

A Multiple Instance Learning Strategy for Combating Good Word Attacks on Spam Filters

Zach Jorgensen ZDJORGEN@JAGUAR1.USOUTHAL.EDU
Yan Zhou ZHOU@CIS.USOUTHAL.EDU
Meador Inge WMI601@JAGUAR1.USOUTHAL.EDU
School of Computer and Information Sciences
University of South Alabama
Mobile, AL 36688, USA

Editor: Lyle Ungar

Abstract

Statistical spam filters are known to be vulnerable to adversarial attacks. One of the more common adversarial attacks, known as the *good word attack*, thwarts spam filters by appending to spam messages sets of "good" words, which are words that are common in legitimate email but rare in spam. We present a counterattack strategy that attempts to differentiate spam from legitimate email in the input space by transforming each email into a bag of multiple segments, and subsequently applying multiple instance logistic regression on the bags. We treat each segment in the bag as an instance. An email is classified as spam if at least one instance in the corresponding bag is spam, and as legitimate if all the instances in it are legitimate. We show that a classifier using our multiple instance counterattack strategy is more robust to good word attacks than its single instance counterpart and other single instance learners commonly used in the spam filtering domain.

Keywords: spam filtering, multiple instance learning, good word attack, adversarial learning

1. Introduction

It has been nearly thirty years since the first email spam appeared on the Arpanet. Today, to most end users, spam does not seem to be a serious threat due to the apparent effectiveness of current spam filters. Behind the scenes, however, is a seemingly never-ending battle between spammers and spam fighters. With millions of email users, profit-driven spammers have great incentives to spam. With as little as 0.001% response rate, a spammer could potentially profit $25,000 on a $50 product (Carpinter and Hunt, 2006). Over the years, spammers have grown in sophistication with cutting-edge technologies and have become more evasive. The best evidence of their growing effectiveness is a recent estimate of over US $10 billion worldwide spam-related cost (Jennings, 2005). In this paper, we target one of the adversarial techniques spammers often use to circumvent existing spam filters.

Adversarial attacks on spam filters have become an increasing challenge to the anti-spam community. The good word attack (Lowd and Meek, 2005b) is one of the techniques most frequently employed by spammers. This technique involves appending sets of so-called "good words" to spam messages. Good words are words that are common to legitimate emails (also called ham) but rare in spam. Spam messages injected with such words are more likely to appear legitimate and bypass spam filters. So far, relatively little research has been done to investigate how spam filters might be

trained to account for such attacks. This paper presents a possible defense strategy using multiple instance learning that has shown promising results in our experiments.

Multiple instance (MI) learning (Dietterich et al., 1997) differs from single instance supervised learning in that an example is represented by a set, or bag, of instances rather than as just a single instance. The bag is assigned a class label (either *positive* or *negative*) based on the instances it contains; however, the instances within the bag are not necessarily labeled. Classic MI learning assumes that a bag is positive if at least one instance in the bag is positive, and negative if all instances are negative. Therefore, the goal of multiple instance learning is to learn a classification function that accurately maps a given bag to a class. Formally, let $B = \{B_1, \ldots, B_i, \ldots, B_m\}$ be a set of bags where $B_i = \{X_{1i}, X_{2i}, \ldots, X_{ji}\}$ is the i^{th} bag and $X_{1i}, X_{2i}, \ldots, X_{ji}$ are the j instances contained in bag B_i. If B is a training set, then every $B_i \in B$ also has a class label $c_i \in C = \{positive, negative\}$ associated with it. The training process, using B as input, yields a binary classification function $f(B_i) : B \rightarrow C$ that maps a bag to a class label.

Our spam filtering strategy adopts the classical MI assumption, which states that a bag is positive if at least one of its instances is positive, and negative if all instances are negative. We treat each email as a bag of instances. Thus, an email is classified as spam if at least one instance in the corresponding bag is spam, and as legitimate if all the instances in it are legitimate. The idea is that by splitting an email into multiple instances, a multiple instance learner will be able to recognize the spam part of the message even if the message has been injected with good words. Our experimental results show that a multiple instance learner, combined with an appropriate technique for splitting emails into multiple instance bags, is more robust to good word attacks than its single instance counterpart and other single instance learners that are commonly used in the spam filtering domain.

The remainder of this paper is organized as follows. First, we discuss recent research that has motivated our work. Next, we formalize the spam filtering problem as a multiple instance learning problem and explain our proposed counterattack strategy in more detail. Following that, we present our experimental results to demonstrate the effectiveness of our filtering strategy. Finally, we conclude our work and discuss future directions.

2. Related Work

Our work is primarily motivated by recent research on adversarial learning (Dalvi et al., 2004; Lowd and Meek, 2005a; Kolter and Maloof, 2005). Dalvi et al. (2004) consider classification to be a game between classifiers and adversaries in problem domains where adversarial attacks are expected. They model the computation of the adversary's optimal strategy as a constrained optimization problem and approximate its solution based on dynamic programming. Subsequently, an optimal classifier is produced against the optimal adversarial strategy. Their experimental results demonstrate that their game-theoretic approach outperforms traditional classifiers in the spam filtering domain. However, in their adversarial classification framework, they assume both the classifier and the adversary have perfect knowledge of each other, which is unrealistic in practice.

Instead of assuming the adversary has perfect knowledge of the classifier, Lowd and Meek (2005a) formalized the task of adversarial learning as the process of reverse engineering the classifier. In their adversarial classifier reverse engineer (ACRE) framework, the adversary aims to identify difficult spam instances (the ones that are hard to detect by the classifier) through membership queries. The goal is to find a set of negative instances with minimum adversarial cost within a polynomial number of membership queries.

Newsome et al. (2006) emphasize the point that the training data used to build classifiers for spam filtering and the similar problem of internet worm detection is, to a large extent, controlled by an adversary. They describe and demonstrate several attacks on the generators of such classifiers in which the adversary is able to significantly impair the learning of accurate classifiers by manipulating the training data, even while still providing correct labels for the training instances. The attacks involve inserting features, in a specific manner, into one or both classes of the training data and are specifically designed to cause a significant increase in false positives or false negatives for the resulting classifier. They conclude that the generation of classifiers for adversarial environments should take into account the fact that training data is controlled by an adversarial source in order to ensure the production of accurate classifiers.

Barreno et al. (2006) explore possible adversarial attacks on machine learning algorithms from multiple perspectives. They present a taxonomy of different types of attacks on machine learning systems. An attack is *causative* if it targets the training data, and is *exploratory* if it aims to discover information through, for example, offline analysis. An attack is *targeted* if it focuses on a small set of points, and is *indiscriminate* if it targets a general class of points. An *integrity* attack leads to false negatives, and an *availability* attack aims to cause (machine learning) system dysfunction by generating many false negatives and false positives. They also discuss several potential defenses against those attacks, and give a lower bound on the adversary's effort in attacking a naïve learning algorithm.

A practical example of adversarial learning is learning in the presence of the good word attack. Lowd and Meek (2005b) present and evaluate several variations of this type of attack on spam filters. They demonstrate two different ways to carry out the attack: passively and actively. Active good word attacks use feedback obtained by sending test messages to a spam filter in order to determine which words are "good". The active attacks were found to be more effective than the passive attacks; however, active attacks are generally more difficult to perform than passive attacks because they require user-level access to the spam filter, which is not always possible. Passive good word attacks, on the other hand, do not involve any feedback from the spam filter, but rather, guesses are made as to which words are considered good. Three common ways for passively choosing good words are identified. First, *dictionary attacks* involve selecting random words from a large collection of words, such as a dictionary. In testing, this method did not prove to be effective; in fact, it actually increased the chances that the email would be classified as spam. Next, *frequent word attacks* involve the selection of words that occur most often in legitimate messages, such as news articles. This method was more effective than the previous one, but it still required as many as 1,000 good words to be added to the original message. Finally, *frequency ratio attacks* involve the selection of words that occur very often in legitimate messages but not in spam messages. The authors' tests showed that this technique was quite effective, resulting in the average spam message being passed off as legitimate by adding as few as 150 good words to it. Preliminary results were also presented that suggested that frequent retraining on attacked messages may help reduce the effect of good word attacks on spam filters.

Webb et al. (2005) also examined the effectiveness of good word attacks on statistical spam filters. They present a "large-scale evaluation" of the effectiveness of the attack on four spam filters: naïve Bayes, support vector machine (SVM), LogitBoost, and SpamProbe. Their experiments were performed on a large email corpus consisting of around a million spam and ham messages, which they formed by combining several public and private corpora. Such a large and diverse corpus more closely simulates the environment of a server-level spam filter than a client-level filter. The

experimental results show that, on normal email, that is, email that has not been modified with good words, each of the filters is able to attain an accuracy as high as 98%. When testing on "camouflaged messages", however, the accuracies of the filters drop to between 50% and 75%. In their experiments, spam emails were camouflaged by combining them with portions of legitimate messages. They experimented with camouflaged messages containing twice as much spam content as legitimate content, and vice versa. They also proposed and demonstrated a possible solution to the attack. By training on a collection of emails consisting of half normal and half camouflaged messages, and treating all camouflaged messages as spam, they were able to improve the accuracy of the filters when classifying camouflaged messages.

Our counterattack strategy against good word attacks is inspired by work in the field of multiple instance (MI) learning. The concept of MI learning was initially proposed by Dietterich et al. (1997) for predicting drug activities. The challenge of identifying a drug molecule that binds strongly to a target protein is that a drug molecule can have multiple conformations, or shapes. A molecule is positive if at least one of its conformations binds tightly to the target, and negative if none of its conformations bind well to the target. The problem was tackled with an MI model that aims to learn axis-parallel rectangles (APR). Later, learning APR in the multiple instance setting was further studied and proved to be NP-complete by several other researchers in the PAC-learning framework (Auer, 1997; Long and Tan, 1998; Blum and Kalai, 1998).

Several probabilistic models: Diverse Density (DD) (Maron and Lozano-Pérez, 1998) and its variation EM-DD (Zhang and Goldman, 2002), and multiple instance logistic regression (MILR) (Ray and Craven, 2005), employ a maximum likelihood estimation to solve problems in the MI domain. The original DD algorithm searches for the target concept by finding an area in the feature space with maximum diverse density, that is, an area with a high density of positive points and a low density of negative points. The diverse density at a point in the feature space is defined to measure probabilistically how many different positive bags have instances near that point, and how far the negative instances are from that point. EM-DD combines EM with the DD algorithm to reduce the multiple instance learning problem to a single-instance setting. The algorithm uses EM to estimate the instance in each bag which is most likely to be the one responsible for the label of the bag. The MILR algorithm presented by Ray and Craven (2005) is designed to learn linear models in a multiple instance setting. Logistic regression is used to model the posterior probability of the label of each instance in a bag, and the bag level posterior probability is estimated by using softmax to combine the posterior probabilities over the instances of the bag. Similar approaches with different combining functions are presented by Xu and Frank (2004).

Many single-instance learning algorithms have been adapted to solve the multiple instance learning problem. For example, Wang and Zucker (2000) propose the lazy MI learning algorithms, namely Bayesian-*k*NN and citation-*k*NN, which solve the multiple instance learning problem by using the Hausdorff distance to measure the distance between two bags of points in the feature space. Chevaleyre and Zucker (2001) propose the multi-instance decision tree ID3-MI and decision rule learner RIPPER-MI by defining a new multiple instance entropy function and a multiple instance coverage function. Other algorithms that have been adapted to multiple instance learning include the neural network MI-NN (Ramon and Raedt, 2000), DD-SVM (Chen and Wang, 2004), MI-SVM and mi-SVM (Andrews et al., 2003), multi-instance kernels (Gärtner et al., 2002), MI-Ensemble (Zhou and Zhang, 2003), and MI-Boosting (Xu and Frank, 2004).

In this paper, we demonstrate that a counterattack strategy against good word attacks, developed in the framework of multiple instance learning, can be very effective, provided that a single instance

can be properly transformed into a bag of instances. We also explore several possible ways to transform emails into bags of instances. Our experiments also verify earlier observations, discussed in other works (Lowd and Meek, 2005b; Webb et al., 2005), that retraining on emails modified during adversarial attacks may improve the performance of the filters against the attack.

3. Problem Definition

Consider a standard supervised learning problem with a set of training data $D = \{< X_1, Y_1 >, \ldots, < X_m, Y_m >\}$, where X_i is an instance represented as a single feature vector, $Y_i = C(X_i)$ is the target value of X_i, where C is the target function. Normally, the task is to learn C given D. The learning task becomes more difficult when there are adversaries who could alter some instance X_i so that $X_i \rightarrow X_i'$ and cause $Y_i \rightarrow Y_i'$, where $Y_i \neq Y_i'$. Let ΔX_i be the difference between X_i and X_i', that is, $X_i' = X_i + \Delta X_i$. In the case of spam filtering, an adversary can modify spam emails by injecting them with good words. So, ΔX_i represents a set of good words added to a spam message by the spammer. There are two cases that need to be studied separately:

1. the filter is trained on normal emails, that is, emails that have not been injected with good words, and tested on emails which have been injected with good words;

2. both the training and testing sets contain emails injected with good words.

In the first case, the classifier is trained on a clean training set. Predictions made for the altered test instances are highly unreliable. In the second case, the classifier may capture some adversarial patterns as long as the adversaries consistently follow a particular pattern.

In both cases, the problem becomes trivial if we know exactly how the instances are altered; we could recover the original data and solve the problem as if no instances were altered by the adversary. In reality, knowing exactly how the instances are altered is impossible. Instead, we seek to approximately separate X_i and ΔX_i and treat them as separate instances in a bag. We then apply multiple instance learning to learn a hypothesis defined over a set of bags.

4. Multiple Instance Bag Creation

We now formulate the spam filtering problem as a multiple instance binary classification problem in the context of adversarial attacks. Note that the adversary is only interested in altering positive instances, that is, spam, by injecting sets of good words that are commonly encountered in negative instances, that is, legitimate emails, or ham. We propose four different approaches to creating multiple instance bags from emails. We call them split-half (split-H), split-term (split-T), split-projection (split-P), and split-subtraction (split-S). We will now discuss each of these splitting methods, in turn. Later, in Section 8, we investigate and discuss possible weaknesses of some of these splitting methods.

4.1 Split-H

The first and simplest splitting method that we considered, which we call *split-half* (split-H), involves splitting an email down the middle into approximately equal halves. Formally, let $B = \{B_1, \ldots, B_i, \ldots, B_m\}$ be a set of bags (emails), where $B_i = \{X_{i1}, X_{i2}\}$ is the i^{th} bag, and X_{i1}, X_{i2} are

the two instances in the i^{th} bag created from the upper half and the lower half of the email respectively. This splitting approach is reasonable in practice because spammers usually append a section of good words to either the beginning or the end of an email to ensure the legibility of the spam message. As will be discussed in Section 8, this splitting method, because it relies on the positions of words in an email, could potentially be circumvented by the spammer. The next three splitting methods do not rely on the positions of the words and thus do not suffer from that problem.

4.2 Split-T

The second splitting method, *split-term* (split-T), partitions a message into three groups of words (terms) depending on whether the word is an indicator of spam, an indicator of ham, or neutral, that is, $B_i = \{X_{is}, X_{in}, X_{ih}\}$, where X_{is} is the spam-likely instance, X_{in} is the neutral instance, and X_{ih} is the ham-likely instance in bag B_i. The instance to which each word is assigned is based on a weight generated for it during preprocessing. These weights are calculated using word frequencies obtained from the spam and legitimate messages in the training corpus. More specifically, the weight of a term W is given as follows:

$$weight(W) = \frac{p(W \mid D_s)}{p(W \mid D_s) + p(W \mid D_h)},$$

where D_s and D_h are the spam and ham emails in the training set respectively. When splitting an email into instances we used two threshold values, $thresh_s$ and $thresh_\ell$, to determine which instance (spam-likely, ham-likely, or neutral) each word in the email should be assigned to, given its weight. We considered any word with a weight greater than $thresh_s$ to be spammy, any word with a weight less than $thresh_\ell$ to be legitimate, and any word with a weight in between to be neutral. In our experiments, reasonable threshold values were determined by using cross-validation on training emails. Given each training set, $thresh_s$ was selected such that some fraction of the terms chosen during attribute selection (discussed in Section 6.2) would have a weight greater than or equal to it. $thresh_\ell$ was selected so that some other fraction of the terms would have a weight less than or equal to it.

4.3 Split-P

The third splitting method, *split-projection* (split-P), transforms each message into a bag of two instances by projecting the message vector onto the spam and ham prototype vectors. The prototype vectors are computed using all the spam and ham messages in the training set. If we view the spam and ham messages in the training set as two clusters, then the prototypes are essentially the centroid of the two clusters. More specifically, let C_s be the set of emails that are spam and C_ℓ be the set of emails that are legitimate. The prototypes are computed using Rocchio's algorithm (Rocchio Jr., 1971) as follows:

$$P_s = \beta \cdot 1/|C_s| \cdot \sum_{i=1}^{|C_s|} C_{s_i} - \gamma \cdot 1/|C_\ell| \cdot \sum_{i=1}^{|C_\ell|} C_{\ell_i},$$

$$P_\ell = \beta \cdot 1/|C_\ell| \cdot \sum_{i=1}^{|C_\ell|} C_{\ell_i} - \gamma \cdot 1/|C_s| \cdot \sum_{i=1}^{|C_s|} C_{s_i}$$

where C_{s_i} is the i^{th} spam message in C_s and C_{ℓ_i} is the i^{th} ham message in C_ℓ, β is a fixed constant suggested to be 16 and γ is a fixed constant suggested to be 4. Given a message M, two new

instances, M_S and M_ℓ, are formed by projecting M onto P_s and P_ℓ:

$$M_s = \frac{M \cdot P_s}{|P_s|^2} P_s,$$
$$M_\ell = \frac{M \cdot P_\ell}{|P_\ell|^2} P_\ell.$$

The rationale of this splitting approach rests on the assumption that a message is close to the spam prototype in terms of cosine similarity if it is indeed spam, and a ham message is close to the ham prototype.

4.4 Split-S

The last splitting method, *split-subtraction* (split-S), like the former, uses prototype (centroid) vectors. In this method, however, the ham and spam prototypes are calculated by averaging the corresponding attribute values of all of the ham and spam emails, respectively:

$$P_s = 1/|C_s| \cdot \sum_{i=1}^{|C_s|} C_{s_i},$$
$$P_\ell = 1/|C_\ell| \cdot \sum_{i=1}^{|C_\ell|} C_{\ell_i}.$$

where C_s is a set of spam and C_{s_i} is the i^{th} spam message in C_s; C_ℓ is a set of ham, and C_{ℓ_i} is the i^{th} ham message in C_ℓ. A message can then be transformed from a single instance attribute vector M into a bag of two instances by subtracting corresponding attribute values in the single instance vector from the ham prototype and the spam prototype, yielding a legitimate instance $M_\ell = M - P_\ell$ and a spam instance $M_s = M - P_s$, respectively (Zhang and Zhou, 2007).

Now that we have devised several techniques for creating multiple instance bags from email messages, we can transform the standard supervised learning problem of spam filtering into a multiple instance learning problem under the standard MI assumption. In this paper, we adopt the multiple instance logistic regression (MILR) model to train a spam filter that is more robust to adversarial good word attacks than traditional spam filters based on single instance models. We chose to use the MILR classifier over other MI classifiers mainly because its single instance counter-part, logistic regression (LR), which has been shown to be very effective in the spam filtering domain (Yih et al., 2006), appeared to be the best among the single instance learners considered in our experiments. The next section outlines multiple instance logistic regression.

5. Multiple Instance Logistic Regression

Given a set of training bags

$$B = \{< B_1, Y_1 >, \ldots, < B_i, Y_i >, \ldots, < B_m, Y_m >\},$$

let $Pr(Y_i = 1 \mid B_i)$ be the probability that the i^{th} bag is positive, and $Pr(Y_i = 0 \mid B_i)$ be the probability that it is negative. Here Y_i is a dichotomous outcome of the i^{th} bag (for example, spam or legitimate). The bag-level binomial log-likelihood function is:

$$L = \sum_{i=1}^{m} [Y_i \log Pr(Y_i = 1|B_i) + (1 - Y_i) \log Pr(Y_i = 0|B_i)].$$

In a single instance setting where logistic regression is used, given an example X_i, we model the expected value of the dichotomous outcome of X_i with a sigmoidal response function, that is, $Pr(Y_i = 1 \mid X_i) = \exp(p \cdot X_i + b)/(1 + \exp(p \cdot X_i + b))$, then estimate the parameters p and b that maximize the log-likelihood function. In a multiple instance setting, we do not have direct measure of bag-level probabilities in the log-likelihood function. However, since individual instances in the bags can also be considered as binary response data, we estimate the instance-level class probabilities $Pr(Y_{ij} = 1 \mid X_{ij})$ with a sigmoidal response function as follows:

$$Pr(Y_{ij} = 1 \mid X_{ij}) = \frac{\exp(p \cdot X_{ij} + b)}{1 + \exp(p \cdot X_{ij} + b)},$$

where X_{ij} is the j^{th} instance in the i^{th} bag, and p and b are the parameters that need to be estimated. Thus $Pr(Y_i = 0 \mid B_i)$ with instance-level class probabilities can be computed as follows:

$$Pr(Y_{ij} = 0 \mid X_{ij}) = \frac{1}{1 + \exp(p \cdot X_{ij} + b)}.$$

Now we can compute the probability that a bag is negative as:

$$\begin{aligned} Pr(Y_i = 0 \mid B_i) &= \prod_{j=1}^{n} Pr(Y_{ij} = 0 \mid X_{ij}) \\ &= \exp(-\sum_{j=1}^{n} (\log(1 + \exp(p \cdot X_{ij} + b)))) \end{aligned}$$

where n is the number of instances in the i^{th} bag. Note that this probability estimate encodes the multiple instance assumption, that is, a bag is negative if and only if every instance in the bag is negative, and thus the probability estimate

$$Pr(Y_i = 1 \mid B_i) = 1 - Pr(Y_i = 0 \mid B_i)$$

encodes that a bag is positive if at least one instance in the bag is positive. In our case, given a set of emails for training, X_{ij} is a vector of the frequency counts (or other variations such as a *tf-idf* weight) of unique terms in each email. We can apply maximum likelihood estimation (MLE) to maximize the bag-level log-likelihood function, and estimate the parameters p and b that maximize the probability of observing the bags in B.

6. Experimental Setup

We evaluated our multiple instance learning counterattack strategy on emails from the 2006 TREC Public Spam Corpus (Cormack and Lynam, 2006). Good word attacks were simulated by generating a list of good words from the corpus and injecting them into spam messages in the training and/or test data sets. We compared our counterattack strategy, using the multiple instance logistic regression model and the four splitting methods introduced above, to its single instance learning

counterpart—logistic regression (LR)—and to the support vector machine (SVM) and the multinomial naïve Bayes (MNB) classifiers. Additionally, we tested a relatively new compression-based spam filter (Bratko and Filipič, 2005) against the good word attack. The information in the next two subsections regarding corpus preprocessing and feature selection and weighting does not apply to the compression-based filter; it will be discussed separately in Section 7.3.

6.1 Experimental Data

Our experimental data consists of 36,674 spam and legitimate email messages from the 2006 TREC spam corpus. We preprocessed the entire corpus by stripping HTML and non-textual parts and applying stemming and stop-list to all terms. The **to**, **from**, **cc**, **subject**, and **received** headers were retained, while the rest of the headers were stripped. Messages that had an empty body after preprocessing were discarded. Tokenization was done by splitting on nonalphanumeric characters. We did not take any measures to counter obfuscated words in the spam messages, as that is out of the scope of this paper. Given that there are a large number of possible ways to disguise a word, most content-based spam filters will not be able to deobfuscate the text of a message efficiently (Carpinter and Hunt, 2006). Recently, an efficient complementary filter (Lee and Ng, 2005) has been demonstrated to be able to effectively deobfuscate text with high accuracy. In practice, this type of technique could be used during preprocessing.

For our experiments we sorted the emails in the corpus chronologically by receiving date and evenly divided them into 11 subsets $\{D_1, \ldots, D_{11}\}$. In other words, the messages in subset n come chronologically before the messages in subset $n+1$. Experiments were run in an on-line fashion, that is, training on subset n and testing on subset $n+1$. Each subset contains approximately 3300 messages. The percentage of spam messages in each subset varies as in the operational setting (see Figure 1). We used the Multiple Instance Learning Tool Kit (MILK) (Xu, 2003) implementation of MILR and the Weka 3.4.7 (Witten and Frank, 2000) implementations of LR, SVM and multinomial naïve Bayes, in our experiments. For the compression-based filter, we used the spam filter described in Bratko and Filipič (2005), which uses the prediction by partial matching algorithm with escape method D (PPMD) and is available as part of the PSMSLib C++ library (Bratko, 2008).

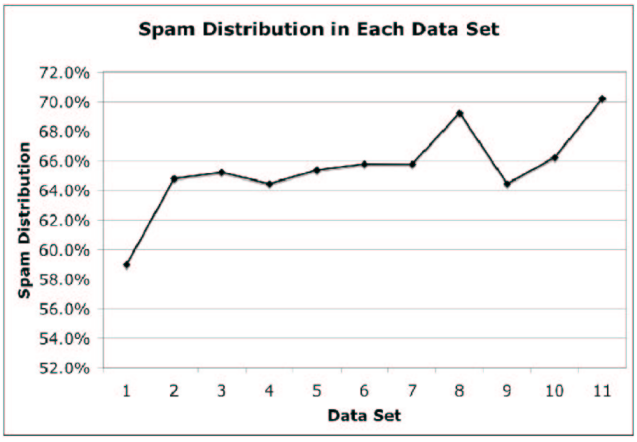


Figure 1: Percentage of emails in each data set that are spam.

6.2 Feature Selection and Weighting

We reduced the feature space used to describe the emails in our experiments to the top 500 features ranked using information gain. Feature selection is necessary for reasons of efficiency and for avoiding the curse of dimensionality. It is also common practice in the spam filtering domain. In our experiments, retaining 500 features appeared to be the best compromise among the classifiers in terms of improved efficiency and impaired performance. Figure 2 shows how the performance of the classifiers varies as the number of retained features increases.

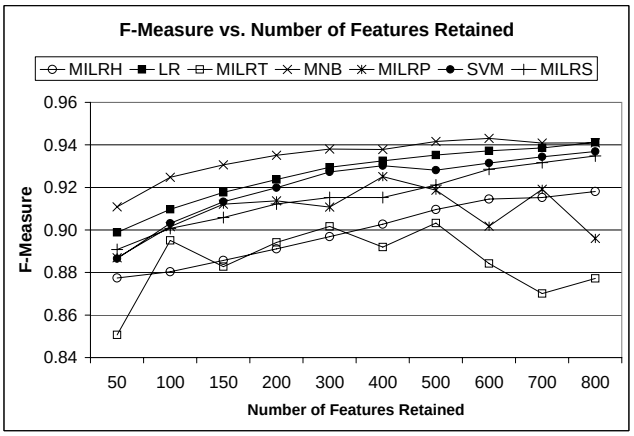


Figure 2: Effect of number of retained features on f-measure.

Attribute values for each email were calculated using the common *tf-idf* (term frequency inverse document frequency) weighting scheme. Under this weighting scheme, attributes are assigned a weight that corresponds to their importance to the email message in the corpus that contains them. The tf-idf weight for a given term in a given email is calculated as follows. Let f be the number of occurrences of the given term in the given email, the *term frequency*. We normalize f by dividing it by the maximum value of f for the given term over all emails in the corpus. Let tf be the normalized value of f. The *inverse document frequency*, idf, is $\log_2(\frac{a}{b})$ where a is the total number of emails in the corpus and b is the number of emails in the corpus that contain the given term. Then the *weight* for the given term is $w = tf \times idf$. Note that tf-idf weighting is widely used in information retrieval and text mining, and has been shown to be able to greatly improve the performance of multinomial naïve Bayes in several text categorization tasks (Kibriya et al., 2004).

6.3 Good Word List Creation

The good word list used in our simulated good word attacks was generated in two different ways: 1) the *global* good word list was generated using all 36,674 messages in the 2006 Trec corpus, 2) and the *local* good word list was generated using messages in the current training set. When generating the global good word list, we ranked every unique word in the corpus according to the ratio of its frequency in the legitimate messages over its frequency in the spam messages. We then selected the top 1,000 words from the ranking to use as our good word list. Generating the good word list in this manner has an important implication. Since the list was generated from the entire corpus rather than from the subset of messages used to train the classifiers, and since we represent emails using a feature vector of 500 features, some of the words in the list will not have an effect

on the classification of messages that they are injected into. Such a list is more representative of the kind of list a spammer would be able to produce in practice, since the spammer would have no way of knowing the exact features used by the target filter. We noticed that in our experiments, only about 10% of the injected good words were actually retained in the feature vector, yet they had a significant impact on the classification. Nevertheless, we also tested the extreme case in which we assumed the adversary has perfect knowledge of the training set and the selected features. We created a local good word list from messages in each training set and kept only the words that are in the selected feature vector.

6.4 Threshold Values for Split-Term

The two threshold values, $thresh_s$ and $thresh_\ell$, must be determined for the splitting method split-term. As mentioned earlier, for each training set, $thresh_s$ was selected such that some fraction of the terms chosen would have a weight greater than or equal to it. $thresh_\ell$ was selected so that some other fraction of the terms would have a weight less than or equal to it. For each of the ten training sets we selected, by using 5-fold cross validation, the best threshold values that divide the terms into three categories—spam, ham, and neutral.

The selected thresholds were used for testing on the test set. Table 1 lists the percentages of the terms divided by the thresholds selected for each training set.

Subset	% of terms with weights $\geq thresh_s$	% of terms with weights $\leq thresh_\ell$
1	20%	50%
2	20%	50%
3	30%	50%
4	10%	50%
5	20%	50%
6	20%	50%
7	10%	50%
8	10%	50%
9	30%	50%
10	30%	50%

Table 1: Percentages of the terms divided by the MILRT threshold values selected for each training set.

7. Experimental Results

We now present the results of two experiments in which we evaluate the effectiveness of our proposed multiple instance counterattack strategy. In the first experiment, we train all of the classifiers on normal email (that is, email that has not been injected with good words) and then test them on email that has been injected with good words. In the second experiment we train on both normal and attacked emails to observe how doing so affects classification of both normal and attacked emails. The compression-based filter and its susceptibility to the good word attack are examined separately in Section 7.3.

7.1 Experiment 1: Attacking the Test Set

In this experiment, we tested the ability of the MILR algorithm, using the four splitting methods introduced above, to classify email injected with good words. We also tested the single instance logistic regression (LR), support vector machine (SVM) and multinomial naïve Bayes (MNB) classifiers for comparison. The classifiers were each trained and tested on the eleven chronologically sorted data sets in an on-line fashion. That is, all of the classifiers were trained on the same unaltered data set D_n, and then tested on the data set D_{n+1}, for $n = 1...10$. Fifteen variations of each test set were created to test the susceptibility of the classifiers to good word attacks of varying strengths. The first version of each test set was left unmodified, that is, no good words were injected. Half of the spam messages (selected at random) in each of the remaining 14 variations of each test set were injected with some quantity of random good words from our global good word list, beginning with 10 words. With each successive version of the test set, the quantity of good words injected into half of the spam messages was increased: first in increments of 10 words, up to 50, and then in increments of 50 words up to 500. The injected words were randomly selected, without replacement, from our global good word list on a message by message basis. We chose to inject good words into only half of the messages in each test set because, in practice, spam messages injected with good words account for only a subset of the spam emails encountered by a given filter. The precision of each classifier was fixed at 0.9 and the corresponding recall on each version of the test set for all 10 test sets was averaged and recorded for each classifier. In our results, we use "MILRH", "MILRT", "MILRP" and "MILRS" where split-H, split-T, split-P and split-S were used with MILR, respectively.

Figure 3 shows how the average recall of each classifier is affected as the good word attack increases in strength (that is, the quantity of good words injected into the spam emails increases). Figures 4-7 and Table 2 show the ROC curves and corresponding AUC values, respectively, for each classifier as the good words are injected. Each ROC graph contains six curves, each corresponding to a specific quantity of good words. We chose not to include curves for all quantities of good words in order to keep the graphs readable. To make comparison easier, Figure 8 shows two ROC graphs containing the ROC curves of all the classifiers when 0 words and 500 words are added to the test set respectively. ROC graphs show the tradeoffs between true positives and false positives and are commonly used to visualize the performance of classifiers (Fawcett, 2006). The total area under a ROC curve (AUC) is also commonly used as a metric to compare classifiers. The AUC of a spam classifier can be interpreted as the probability that the classifier will rate a randomly chosen spam email as more spammy than a randomly chosen legitimate email. In our results, each ROC curve shown is an average of the curves resulting from the ten subsets. The curves were averaged using the vertical averaging algorithm given by Fawcett (2006).

From the results we can see that, with the exception of MILRT, the good word attack significantly affected the ability of each classifier to identify spam emails. MILRT was the most resilient of all the classifiers to the attack, dropping by only 3.7% (from 0.963 to 0.927) in average recall after 500 good words had been added to the spam messages. MILRH and MILRP stood up better to the attack than the single instance classifiers and the MILRS classifier, but the attack still had a very noticeable effect on their ability to classify spam, reducing the average recall of MILRH by 30.8% (from 0.972 to 0.673) and the average recall of MILRP by 35.3% (from 0.938 to 0.607). Of the single instance classifiers, LR was the most resilient; however, the attack still had a very significant effect on its ability to classify spam, reducing its average recall by 42.5% (from 0.986 to 0.567). The

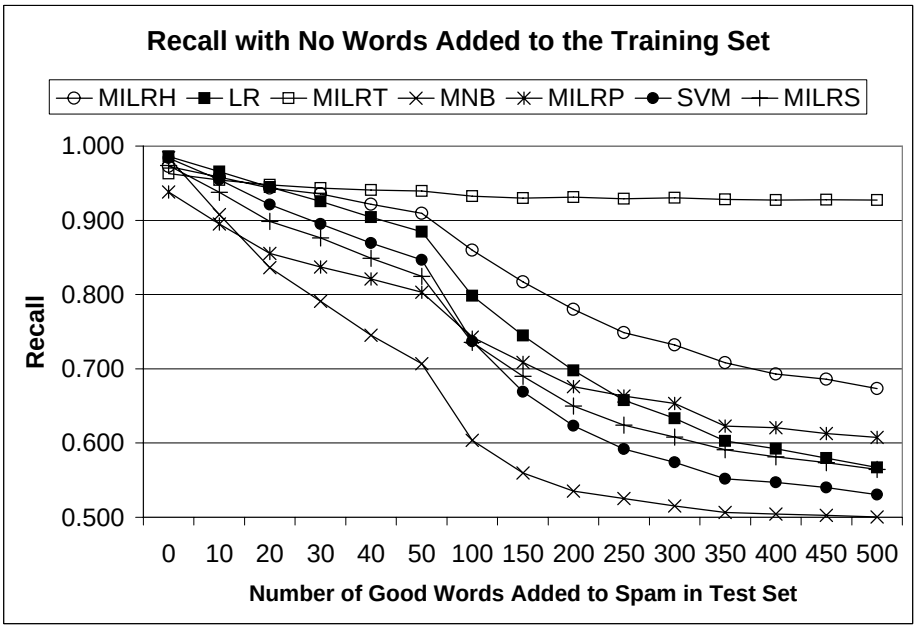
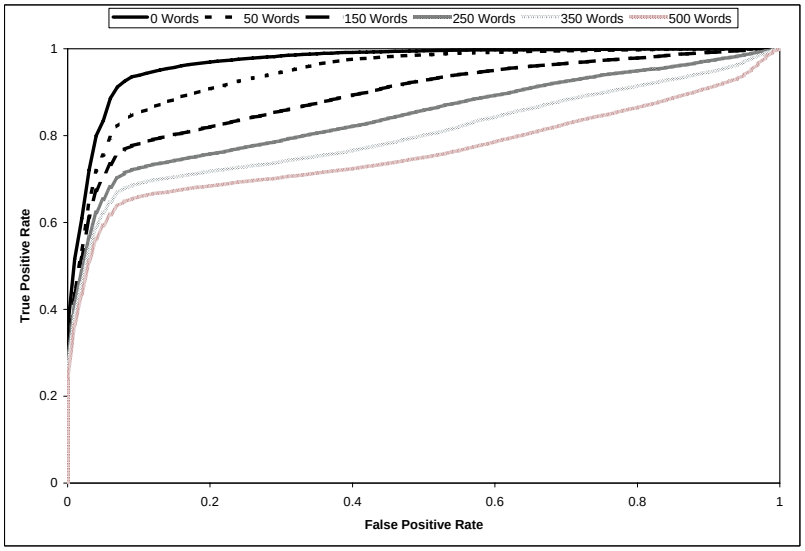


Figure 3: The change in average recall, corresponding to a fixed precision of 0.9, as the quantity of good words injected into half of the spam messages in the test set increases; no good words were injected into the training set.

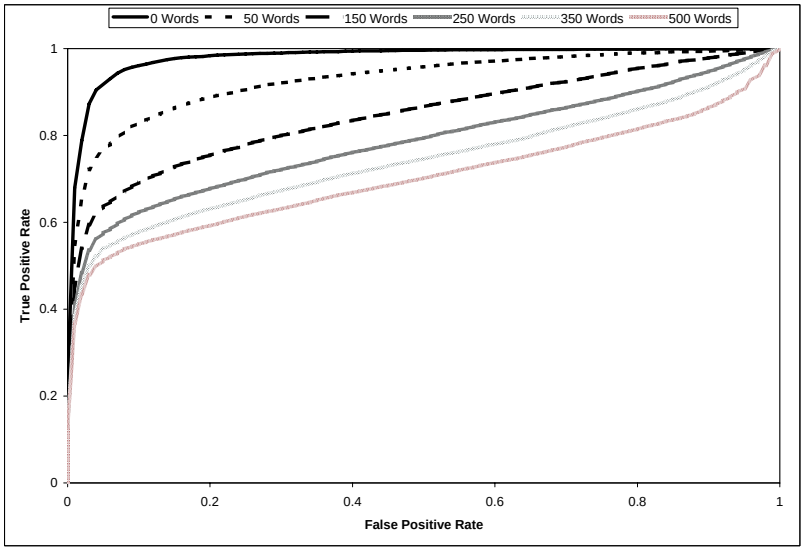
Words	MILRT	MILRH	MILRS	MILRP	LR	MNB	SVM
0	0.946	0.966	0.962	0.957	**0.981**	0.976	0.979
10	0.945	0.962	0.944	0.934	**0.968**	0.935	0.961
20	0.943	0.956	0.928	0.914	**0.958**	0.898	0.946
30	0.941	**0.953**	0.917	0.901	0.948	0.871	0.933
40	0.941	**0.949**	0.903	0.888	0.939	0.842	0.921
50	0.940	**0.943**	0.889	0.875	0.928	0.816	0.910
100	**0.937**	0.919	0.838	0.831	0.883	0.730	0.855
150	**0.935**	0.893	0.804	0.800	0.845	0.684	0.810
200	**0.935**	0.868	0.775	0.774	0.813	0.656	0.774
250	**0.934**	0.844	0.749	0.756	0.785	0.642	0.745
300	**0.934**	0.825	0.730	0.741	0.764	0.631	0.725
350	**0.933**	0.803	0.712	0.726	0.742	0.622	0.702
400	**0.933**	0.786	0.699	0.714	0.727	0.617	0.689
450	**0.933**	0.775	0.688	0.710	0.712	0.614	0.675
500	**0.933**	0.762	0.681	0.702	0.702	0.611	0.664

Table 2: Area Under the ROC Curve as the Quantity of Injected Good Words is Increased.

average recall of MNB and SVM dropped by 49.2% (from 0.984 to 0.500) and 46% (from 0.984 to

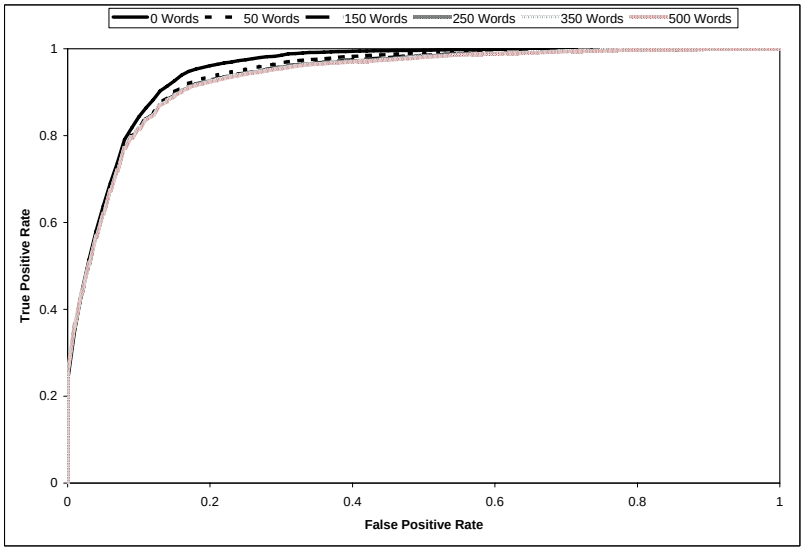


(a) MILRH

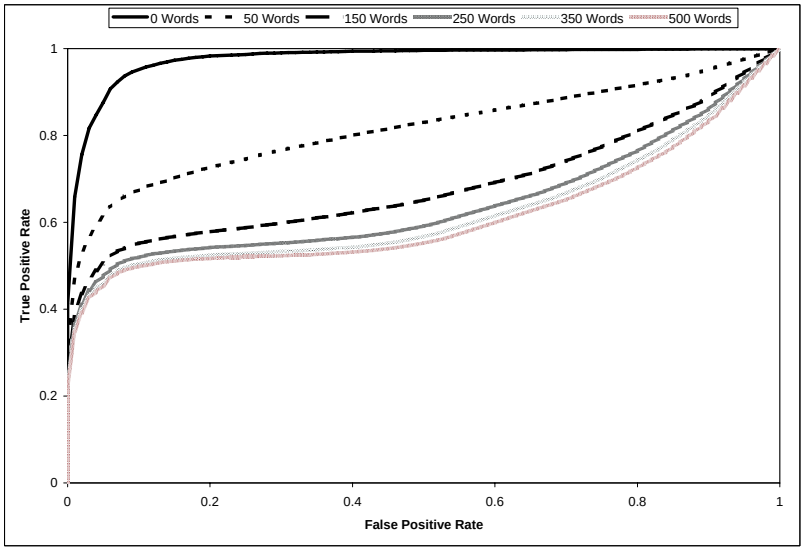


(b) LR

Figure 4: Average ROC curves of (a) MILRH and (b) LR when specific quantities of good words are injected into half of the messages in the test set.

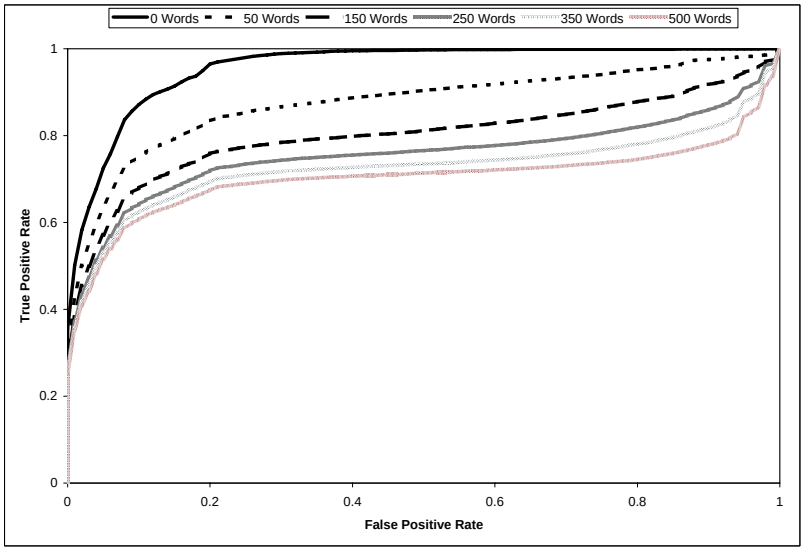


(a) MILRT

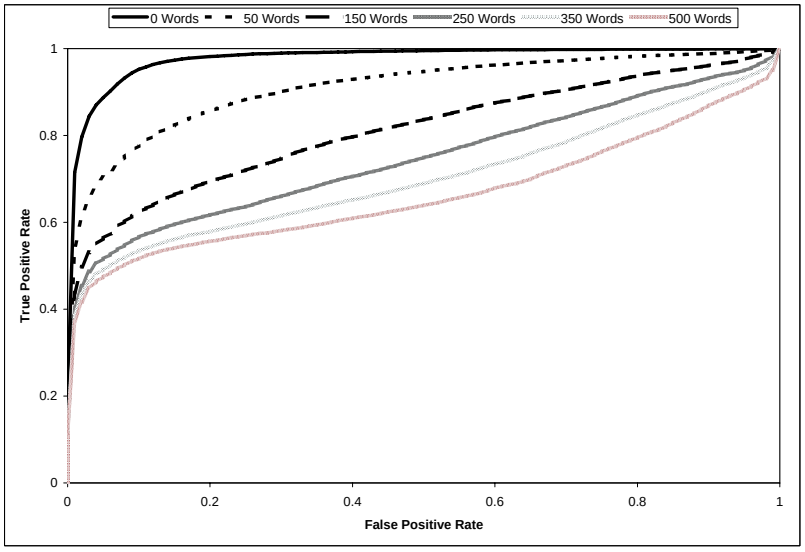


(b) MNB

Figure 5: Average ROC curves of (a) MILRT and (b) MNB when specific quantities of good words are injected into half of the messages in the test set.

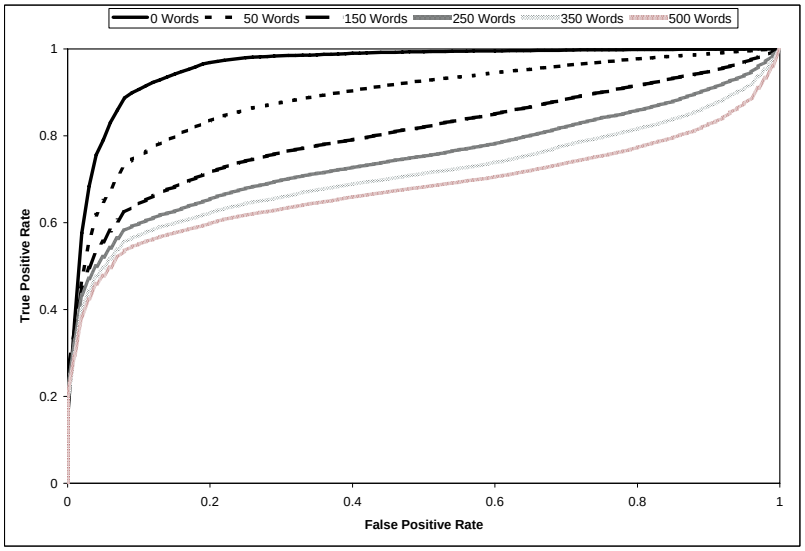


(a) MILRP



(b) SVM

Figure 6: Average ROC curves of (a) MILRP and (b) SVM when specific quantities of good words are injected into half of the messages in the test set.



MILRS

Figure 7: Average ROC curves of MILRS when specific quantities of good words are injected into half of the messages in the test set.

0.531) respectively. MILRS turned out to be nearly as vulnerable to the attack as the single instance classifiers, dropping by 42% (from 0.974 to 0.565) in average recall.

One thing that is clear from these results is that the effectiveness of our multiple instance counterattack strategy is very much dependent on the specific technique used to split emails into multiple instance bags. The success of the split-term method is due to the fact that the classifier is able to consider both spammy and legitimate terms independently, since they are placed into separate instances in the bag created from an email. Under the multiple instance assumption, if at least one instance in a bag is spammy, the entire bag is labeled as spammy. When good words are injected into a spam message they end up in the legitimate instance of the bag and have no effect on the spammy instance; thus the bag still contains a spammy instance and is classified correctly as spam. We verified this by running the experiment again on the following classifier configurations: MILR with no splitting (single instance bags), MILRT with the neutral and hammy instances discarded from each bag, and LR with spammy terms only (all legitimate terms were excluded from the feature vector). We found that using MILR without any of the splitting methods (all bags contained a single instance), caused it to behave almost identically to the way LR behaved in experiment 1. We also found that discarding the neutral and hammy instances from the MILRT bags resulted in a classifier that was unaffected by the good word attack, but was only able to attain a maximum recall of 0.757 and a maximum precision of 0.906. Training LR on spammy terms only produced very similar results; it was unaffected by the good word attack, but only attained a maximum recall of 0.723 and a maximum precision of 0.931.

To test the extreme case, in which an adversary has perfect knowledge of the training set and the selected features, we repeated the experiment using a local good word list for each training set. The words in each of the local good word lists were generated from the respective training set and were

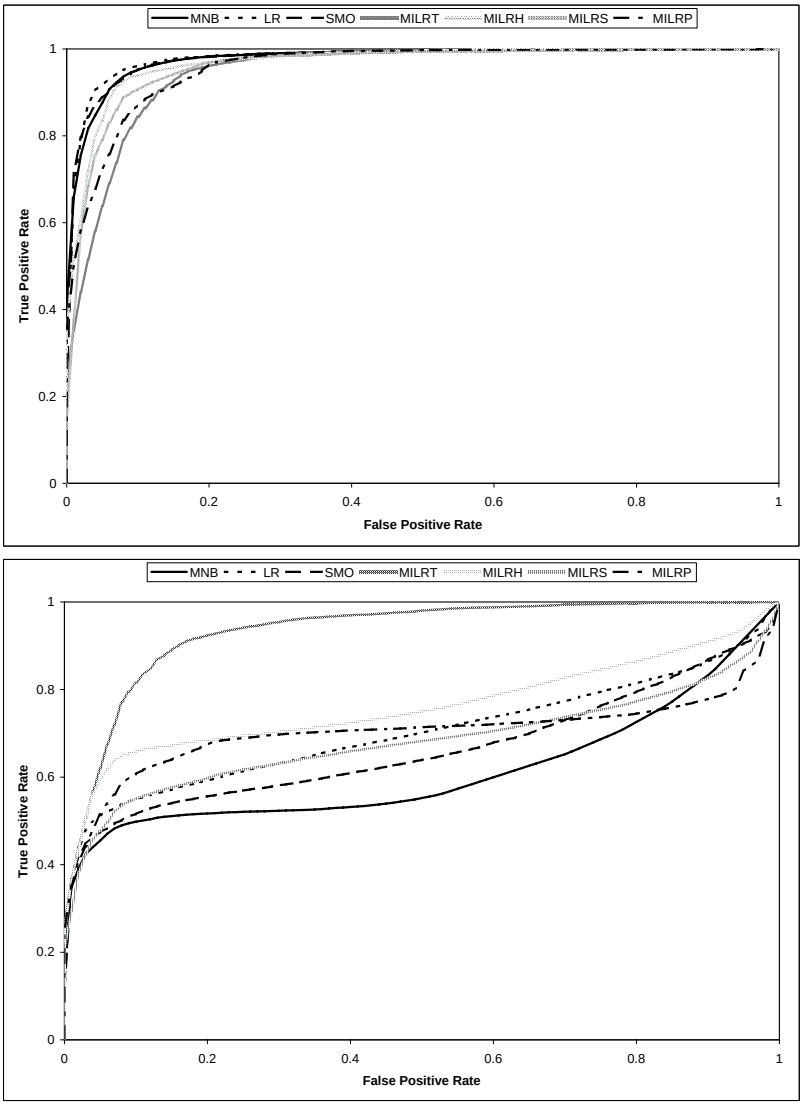


Figure 8: ROC curves for all classifiers when 0 words (top) and 500 words (bottom) have been injected into the test set.

limited to only those good words that were in the selected feature vector for the training set. For each corresponding test set, the entire contents of the local good word list were added to all of the spam messages in the set. Figure 9 shows the result of this attack on each of the classifiers in terms of precision and recall. MILR, with every splitting method, was more resilient to the attack than any of the single instance classifiers. MILRT again was most resilient to the attack. However, the effect of the attack was severe enough for all of the classifiers to consider them defeated for practical purposes. Although it is not realistic to assume that the adversary could obtain perfect knowledge of the target filter in practice, these results serve to illustrate the amount of damage a good word attack could potentially inflict on these classifiers in the extreme cases.

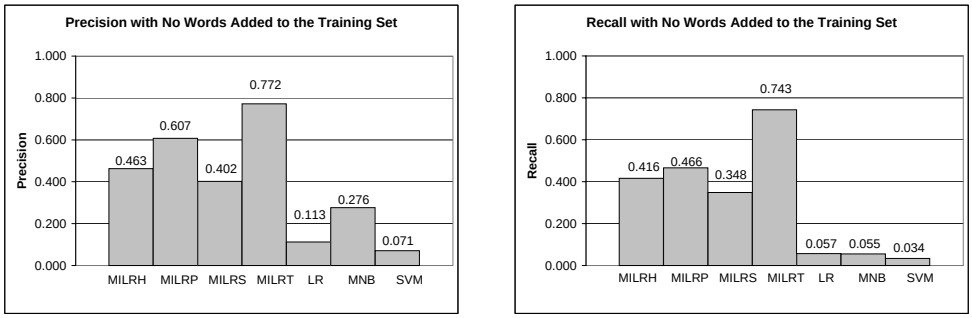


Figure 9: The average precision (left) and recall (right) of each classifier when injecting the entire local good word list into all of the spam messages in the test set.

7.2 Experiment 2: Training on Attacked Spam Messages

In the second experiment, our goal was to observe the effect that training on messages injected with good words has on the susceptibility of the classifiers to attacks on the test set. As in the previous experiment, we tested each of the classifiers on the eleven chronologically sorted data sets in an on-line fashion. This time, however, in addition to creating 15 versions of the test set injected with increasing quantities of good words, we also created 5 versions of the training set. We injected 10 good words into half of the spam messages (selected at random) in the first version of the training set and then increased the number of injected good words by 10 for each subsequent version, up to 50 good words for the fifth version. We also tried injecting larger numbers of good words, but after exceeding 50 words, the additional effect was minimal; therefore, those results are not shown here. For each version of the training set we tested the classifiers on the 15 versions of the corresponding test set. As before, good words were selected from our global good word list randomly and without replacement on a message by message basis. For all ten tests, the precision of each classifier was fixed at 0.9 and the corresponding recall values on each version of the test set were averaged and recorded, separately for each of the 5 versions of the training set. Figures 10– 14 show the change in average recall as the number of good words injected into the training set increased from 10 to 50. Figure 15 shows two graphs containing the ROC curves of all the classifiers when 0 good words and 500 good words are added to the test set respectively and 10 good words are added to the training set. Figure 16 shows the same curves after 50 good words have been added to the training set.

Injecting just 10 good words into half of the spam messages in the training set appeared to lessen the effect of the good word attack for almost all of the classifiers. In particular, the average recall of LR with 500 good words injected into half of the spam messages in the test set was 32.1% higher after 10 good words had been injected into the training set compared to when no good words had been injected into the training set (comparing Figures 3 and 10). Likewise, the average recall values of MNB, SVM, MILRH, MILRP and MILRS were 32.6% higher, 29.4% higher, 10.1% higher, 26.9% higher and 25.8% higher respectively. The average recall for MILRT was actually 5.5% lower even though it was still the best among all the classifiers.

Increasing the number of good words injected into the training set from 10 to 20 (see Figure 11) continued to lessen the effect of the attack for all of the classifiers. After 30 good words had been injected into the training set, the presence of good words in the test messages actually began to increase the likelihood that such messages would be correctly classified as spam. These results

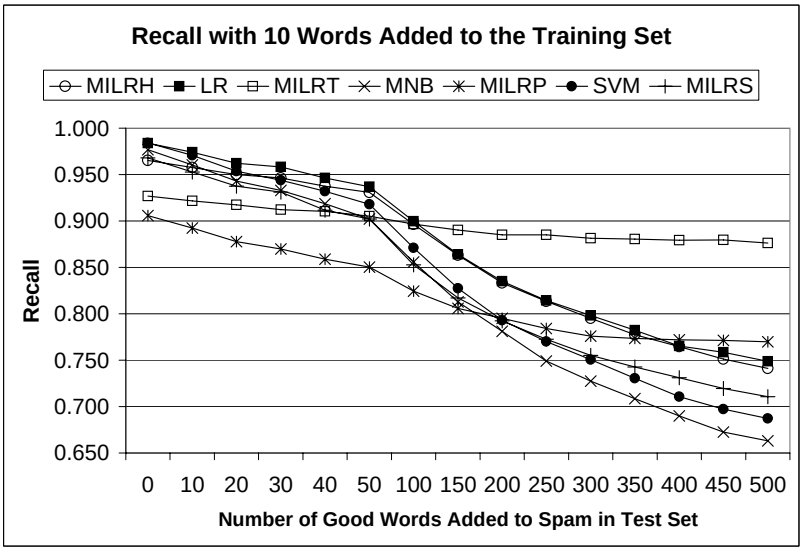


Figure 10: The change in average recall, corresponding to a fixed precision of 0.9, as the number of good words injected into half of the spam messages in the test set increases; 10 good words were also injected into half of the spam messages in the training set.

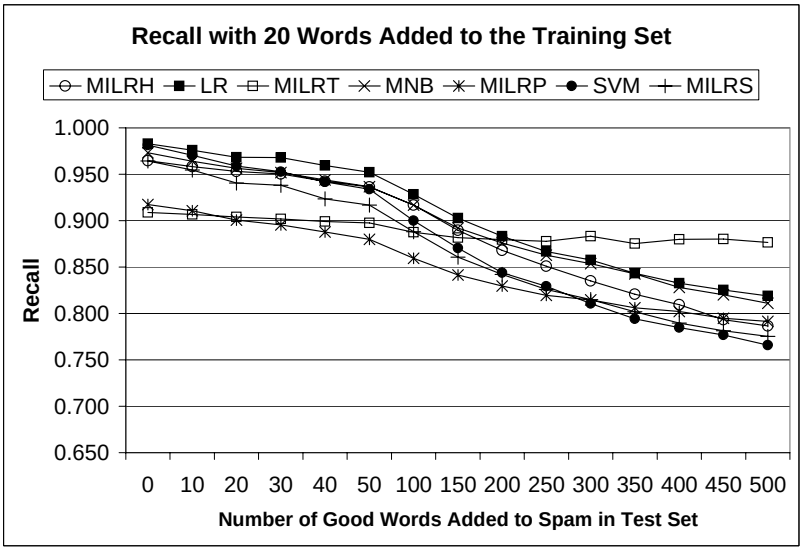


Figure 11: The change in average recall, corresponding to a fixed precision of 0.9, as the number of good words injected into half of the spam messages in the test set increases; 20 good words were also injected into half of the spam messages in the training set.

confirm the observations of several other researchers (Lowd and Meek, 2005b; Webb et al., 2005), that retraining on normal and attacked emails may help to counter the effects of the good word

attack. However, it is important to realize that this would only work in cases where the attacked messages being classified contained the same good words as the attacked messages that the spam filter was trained on. One of the major advantages of our proposed multiple instance strategy is that the spam filter need not be trained on attacked messages in order to be effective against attacks and further, that frequent retraining on attacked messages is not necessary for the strategy to maintain its effectiveness.

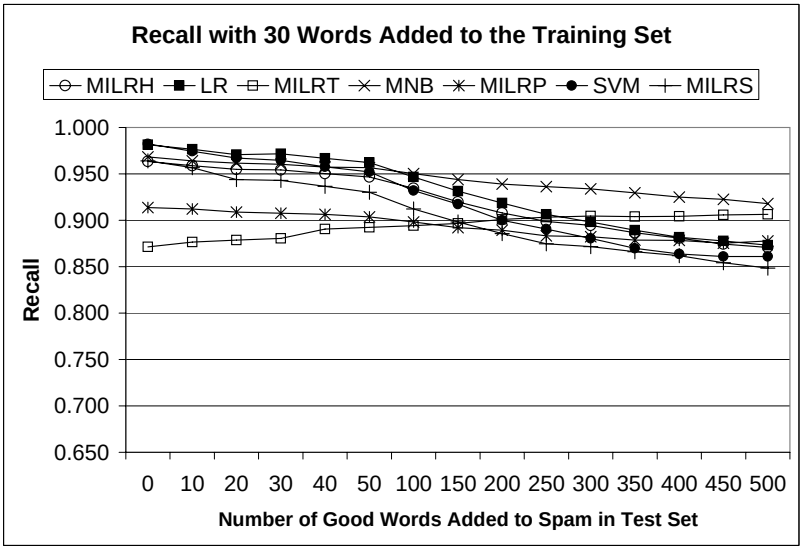


Figure 12: The change in average recall, corresponding to a fixed precision of 0.9, as the number of good words injected into half of the spam messages in the test set increases; 30 good words were also injected into half of the spam messages in the training set.

To test the extreme case, in which an adversary has perfect knowledge of the training set and the selected features, we repeated the experiment using local good word lists for each training set. The words in each of the local good word lists were generated from the respective training set and were limited to only those good words that were in the selected feature vector for the training set. The entire contents of the local good word list were added to all of the spam messages in the corresponding training and test sets. Figure 17 shows the result of this attack on each of the classifiers in terms of precision and recall. Notice that for every classifier, with the exception of multinomial naive Bayes, the effects of the attack on the training set were completely countered by adding the same words to the spam messages in the training set; however, we should again point out that these results are possible only because the good words added to the spam messages in the training and test sets were the same. In practice, there is no such guarantee.

7.3 Attacking the Compression-Based Filter

Relatively new to the spam filtering scene is the idea of using statistical data compression algorithms for spam filtering. Bratko and Filipič (2005) proposed and investigated the use of character-level data compression models for spam filtering. The general idea is to construct two compression models, one from a collection of spam emails and one from a collection of legitimate emails, and

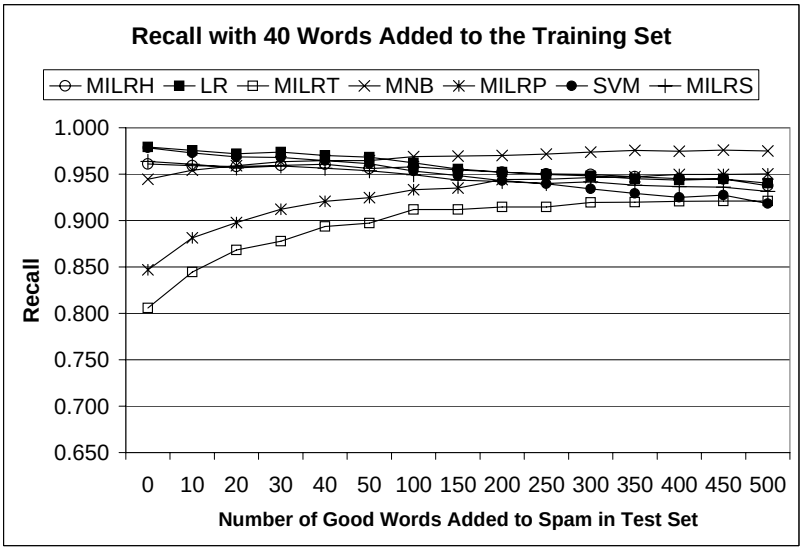


Figure 13: The change in average recall, corresponding to a fixed precision of 0.9, as the number of good words injected into half of the spam messages in the test set increases; 40 good words were also injected into half of the the spam messages in the training set.

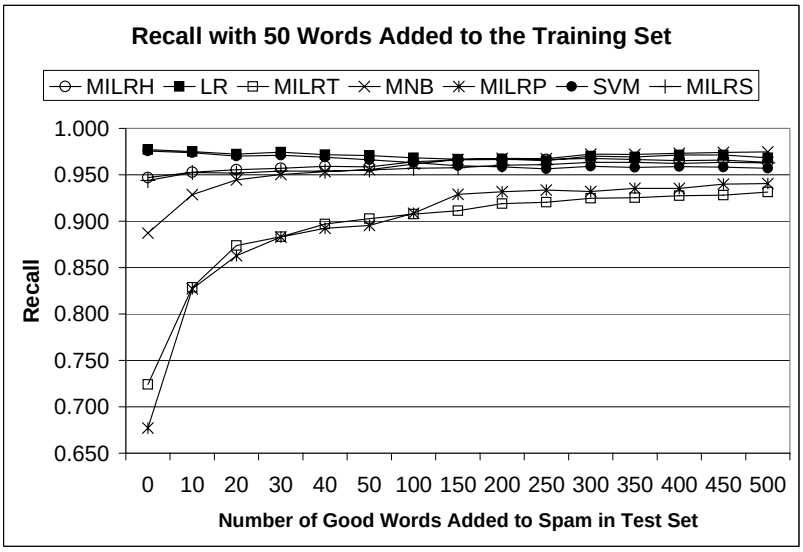


Figure 14: The change in average recall, corresponding to a fixed precision of 0.9, as the number of good words injected into half of the spam messages in the test set increases; 50 good words were also injected into half of the the spam messages in the training set.

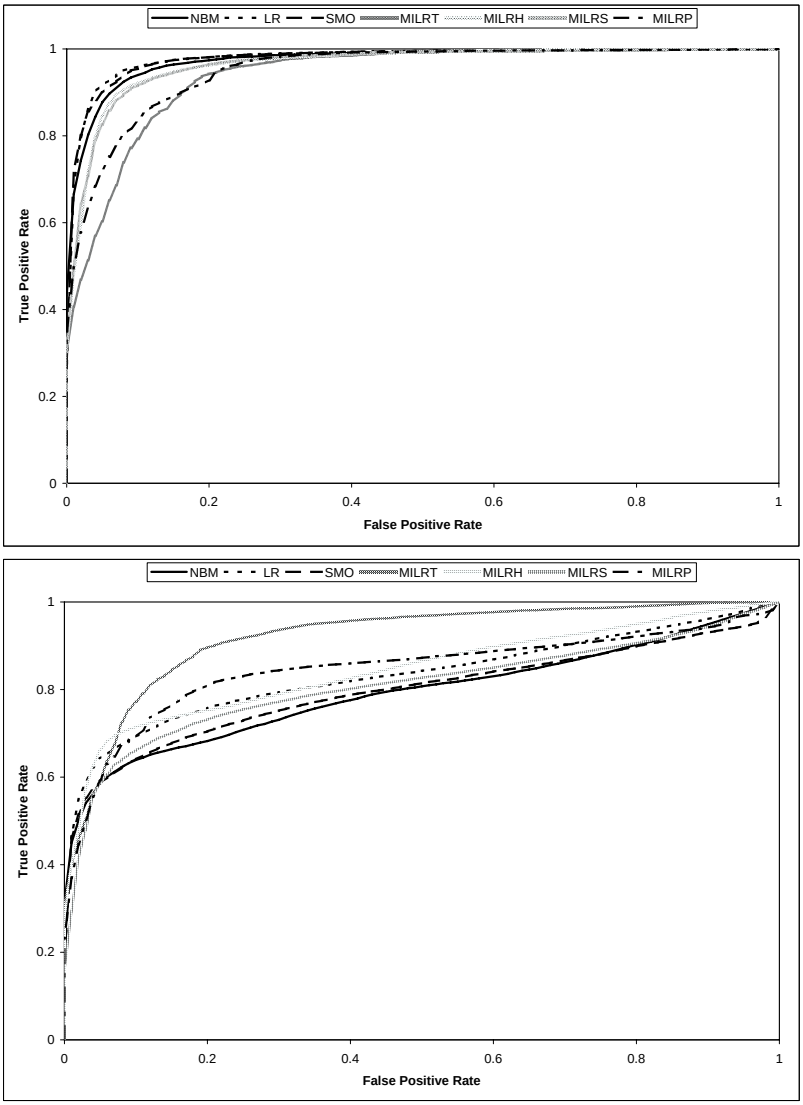


Figure 15: ROC curves for all classifiers when 0 good words (top) and 500 good words (bottom) have been injected into the test set and 10 good words have been injected into the training set.

then to classify a message according to which of the resulting models compresses the message more efficiently. Using statistical data compression for spam filtering has a number of advantages over machine learning algorithms that use word-level models. First, the compression algorithms work on the character level rather than the word level. For this reason, preprocessing and feature selection, both of which are highly prone to error, are unnecessary. Instead the algorithm is able to make full use of all message features. Another benefit of the character-level nature of the compression algorithms is that they are more robust to obfuscation. Spammers often disguise spammy words by deliberately misspelling them or by inserting punctuation between characters. This can cause

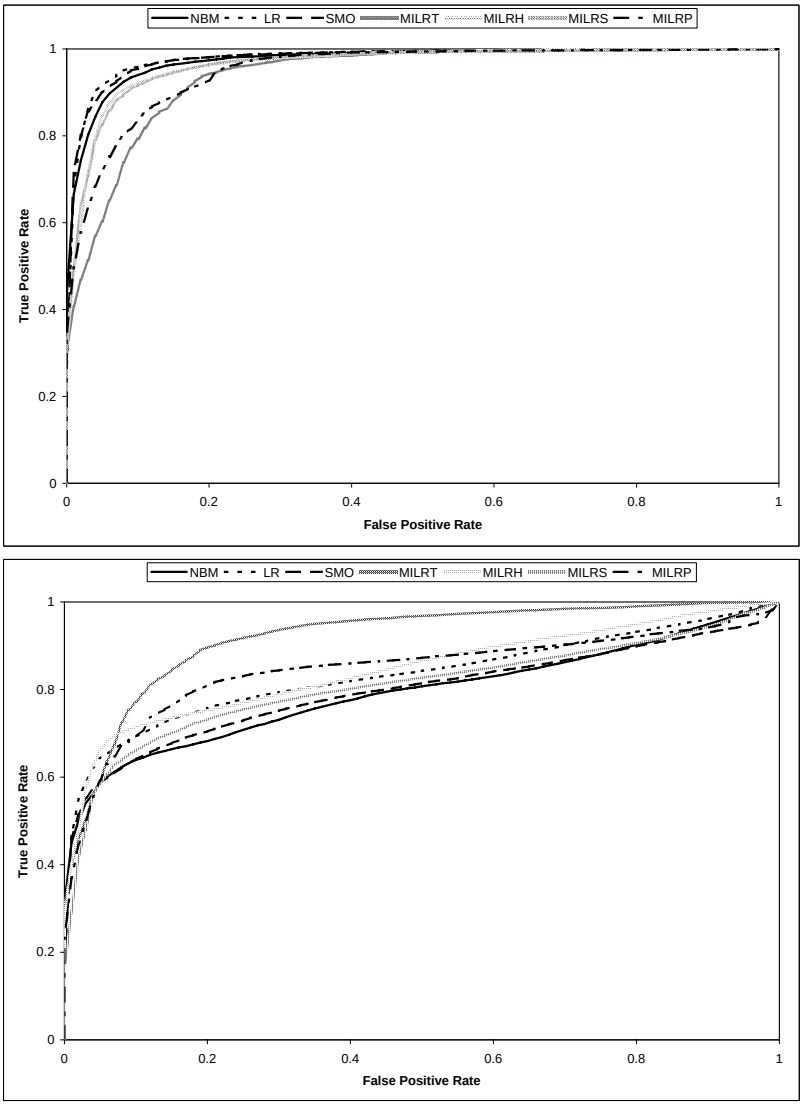


Figure 16: ROC curves for all classifiers when 0 good words (top) and 500 good words (bottom) have been injected into the test set and 50 good words have been injected into the training set.

a word-level spam filter to misclassify emails containing such words unless extra care and effort are expended to detect and deal with these obfuscations. Bratko and Filipič (2005) implemented their compression-based spam filter using the prediction by partial matching algorithm with escape method D (PPMD). Their experiments for the Trec 2005 spam track showed promising results. They also demonstrated that their filter was indeed quite robust to obfuscation. To our knowledge, however, the effects of the good word attack on such filters have not yet been investigated.

We repeated the two experiments from Sections 7.1 and 7.2 on the compression-based spam filter discussed by Bratko and Filipič (2005). Since preprocessing of the input corpus is unnecessary

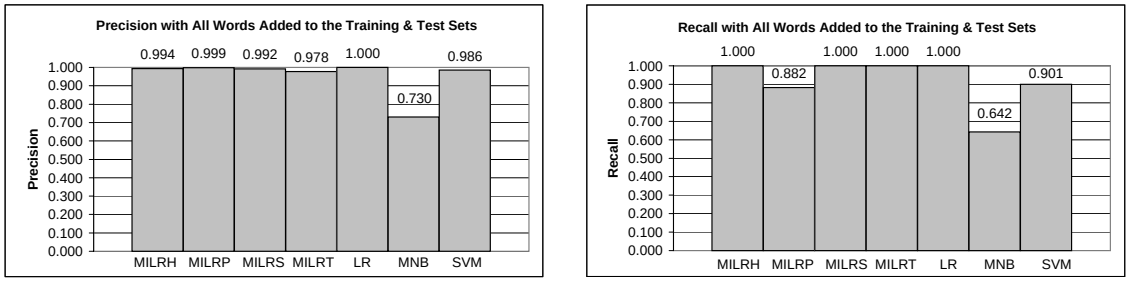


Figure 17: The average precision (left) and recall (right) of each classifier when injecting the entire contents of each local good word list into all of the spam messages in the corresponding training and test sets.

for this type of filter, we ran the experiments using the raw version of the corpus. However, we also ran the two experiments using the preprocessed corpus in order to observe how well the filter stood up against the attack using the same data available to the other filters. Figure 18 shows the results of the first experiment with the PPMD filter on the raw (PPMD1) and preprocessed (PPMD2) corpora. The attack had no effect on the precision of the filter, regardless of which version of the corpus was used; it remained consistently at 0.999 and is not shown on the chart. On the other hand, the recall suffered as a result of the attack, on both corpora. The decrease in recall on the preprocessed corpus was comparable to that of the single instance algorithms. When the raw corpus was used, however, the effect was much less severe. The compression-based filter implementation discussed by Bratko and Filipič (2005), which we used in these experiments, was set by default to truncate all messages to 2500 bytes, presumably for efficiency reasons. However, since our simulated attack appends good words to the bottom of the spam messages, it is possible that truncating the messages could result in some or all of the added good words being removed from spam messages that are longer than 2500 bytes. Therefore, we ran the experiment twice more, truncating at 5000 bytes and 10000 bytes. The result was a drop in average recall (when 500 good words are injected) to 0.702 when truncating at 5000 bytes and a drop to 0.627 when truncating at 10000 bytes, for PPMD1 (raw corpus) (see Figure 19). For PPMD2 (preprocessed corpus) the average recall dropped to 0.503 when truncating at 5000 bytes and dropped to 0.482 when truncating at 10000 bytes (see Figure 20). There was no additional drop in precision when truncating at 5000 or 10000 bytes.

Figure 21 shows the results of the second experiment on the PPMD filter, in terms of average recall, when 10 good words have been injected into the training set. Again, there was virtually no change in average precision so it is not shown on the charts. Apparently injecting 10 good words into the training set was enough to counter any number of good words in the test set. As figure 21 shows, the results of adding up to 50 good words to the training set are nearly identical.

Although it is difficult to directly compare the compression-based filter to the other filters discussed in this paper, due to differences in their modeling and preprocessing requirements, it is safe to say that the compression-based filter is susceptible to good word attacks. However, this type of filter also has a definite advantage over the other algorithms in that it is able to take advantage of message features that the other algorithms cannot easily use, making it more difficult to attack.

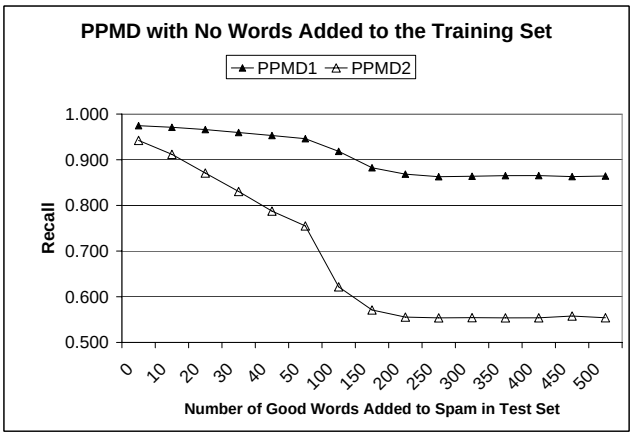


Figure 18: Effect of the good word attack on the PPMD algorithm as the number of good words added into half of the spam messages in the test set increases; no good words were added to the spam in the training set. Messages truncated at 2.5kB.

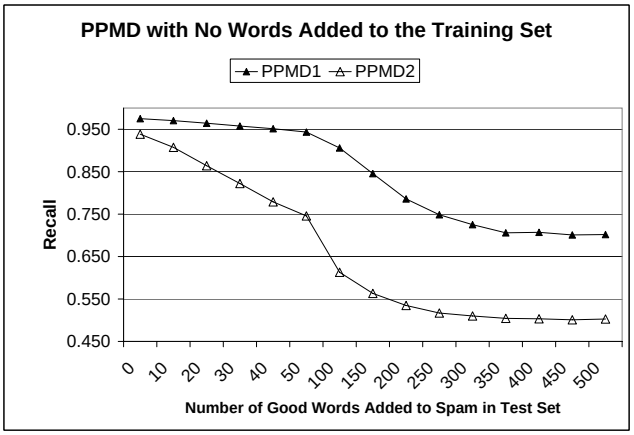


Figure 19: Effect of the good word attack on the PPMD algorithm as the number of good words added into half of the spam messages in the test set increases; no good words were added to the spam in the training set. Messages truncated at 5kB.

8. Potential Attacks on the Splitting Methods

In this section we investigate possible attacks on the splitting methods we have proposed. We recognized two possible ways for a spammer to attack a spam filter equipped with splitting methods like split-H. Both of the attacks work because split-H relies on how the words are physically positioned in an email to split it into multiple instances. One way to attack it is to create a visual pattern with good words so that the original spam message is still legible after the attack, but the spam is fragmented in such a way that "spammy" words are well separated. If this is done correctly, when

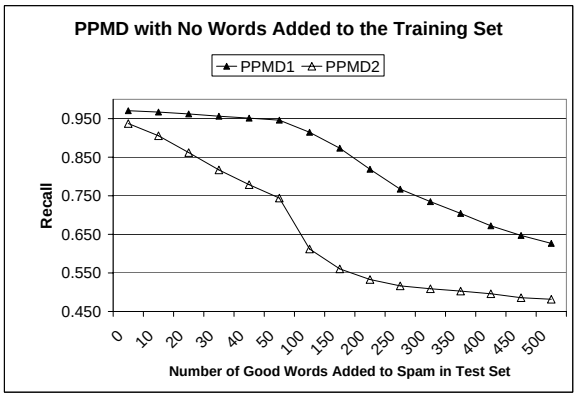


Figure 20: Effect of the good word attack on the PPMD algorithm when half of the spam messages in the test set were altered. Messages truncated at 10kB.

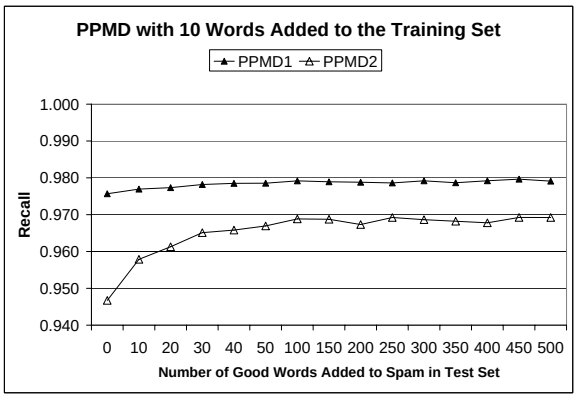


Figure 21: Effect of the good word attack on the PPMD algorithm when half of the spam in the training/test sets were altered; 10 words were added to the training spam.

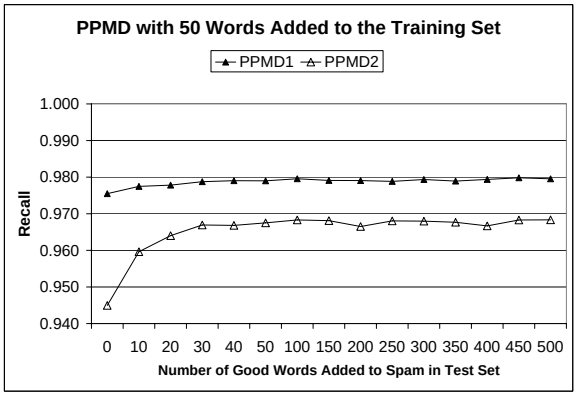


Figure 22: Effect of the good word attack on the PPMD algorithm when half of the spam in the training/test sets were altered; 50 words were added to the training spam.

the attacked message is split, good words should outweigh spammy words in both instances. The example in Table 3 illustrates the idea.

From: foo@internet.org				
To: foo-foo@email.org				
Subject: meeting agenda				
good words	...	**low**	...	good words
good words	...	**mortgage**	...	good words
good words	...	**rate**	...	good words

Table 3: Attacking split-H by fragmenting spam with injected good words.

We tested this attack by running MILRH (MILR with split-H) on the 11 data sets, with the test set at each iteration attacked with 500 good words in the following manner. 50% of the spam messages in each test set were selected at random to be attacked by inserting 3 random good words before and after every 6 words in the message. No more and no less than 500 words were inserted into any message, regardless of the length of the message. That is, in the case of short messages, after 3 good words were inserted before and after every 6 words of the message, words were added to the end of the message until a total of 500 words had been added. For long messages, once 500 words were added, the process was stopped. The good words were selected, without replacement, from the same global good word list used in the other experiments.

Figure 23 compares the effects of adding 500 good words to the messages in the manner just described to the effects of adding 500 good words by appending them to the end of the messages (as in experiment 1), in terms of the recall averaged over the ten tests (corresponding to a fixed precision of 0.9). As the figure shows, attacking the messages in the manner described here drastically decreases the effectiveness of split-H, reducing the average recall of MILRH by 24.8% to 0.506 (compared to that of MILRH in experiment 1 with 500 good words added to the test set, which was 0.673). This attack had the same effect on the other splitting methods as did the attack in experiment 1 (Section 7.1) since the physical position of the words in the attacked messages has no influence on how they are split with those methods; thus, those results are not shown here.

A second way to defeat the split-H method is to append a very large block of good words to the spam messages, so that after the split, good words would still outweigh spammy words in both instances in the bag. In fact, we believe this is exactly what happened in experiment 1. Observe in Figure 3 that the average recall of MILRH did not really begin to drop significantly until after 50 good words had been injected into the spam messages in the test set. As even more good words were injected into the spam messages, the average recall continued to drop as the longer messages began to accumulate enough good words to outweigh the spammy words in both instances. In practice, depending on the length of the spam message, coming up with a large enough block of good words might prove difficult.

9. Conclusions and Future Work

A multiple instance learning counterattack strategy for combating adversarial good word attacks on statistical spam filters has been proposed. In the proposed strategy, emails are treated as multiple

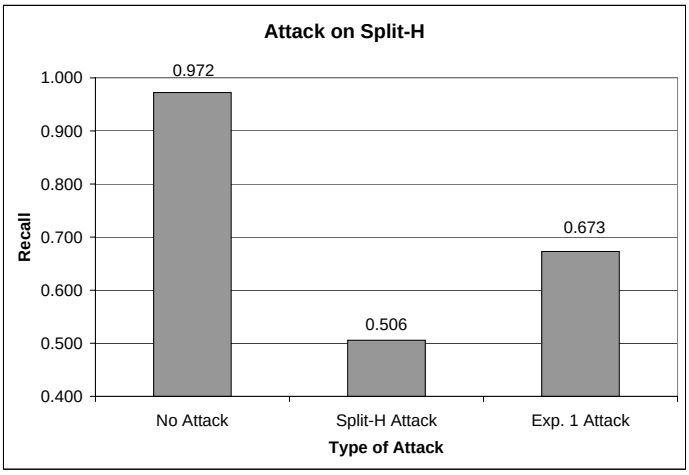


Figure 23: A comparison of the effects of the split-H attack and the experiment 1 style attack, in terms of average recall, with precision fixed at 0.9.

instance bags and a logistic model at the instance level is learned indirectly by maximizing the bag-level binomial log-likelihood function. The proposed counterattack strategy has been demonstrated on good word attacks of varying strength and has been shown to be effective. Additionally, we have confirmed earlier reports that re-training on attacked as well as normal emails may strengthen a spam filter against good word attacks. One of the advantages of our proposed strategy, as demonstrated by our experiments, is that it is effective even when trained on normal email and that frequent re-training on attacked messages is not necessary to maintain that effectiveness. We presented several possible methods for creating multiple instance bags from emails. As was observed from our experimental results, the splitting method used ultimately determines how well the strategy performs. The splitting methods we presented here work fairly well, especially the split-term method, but there are possibly other, perhaps better, methods that could be used. We plan to investigate other possible splitting methods in the future.

Since it is an arms race between spammers and filter designers, we also plan to make our MI strategy adaptive as new spam techniques are devised, and on-line as the concept of spam drifts over time. In addition, we plan to investigate the possibility of extending the proposed multiple instance learning strategy to handle similar adversarial attacks in other domains.

References

S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *NIPS 15*, pages 561–568. MIT Press, 2003.

P. Auer. On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In *Proceedings of the 14th International Conference on Machine Learning*, pages 21–29, San Francisco, CA, 1997. Morgan Kaufmann.

M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-272-0. doi: http://doi.acm.org/10.1145/1128817.1128824.

A. Blum and A. Kalai. A note on learning from multiple-instance examples. *Machine Learning*, 30 (1):23–30, 1998.

A. Bratko. Probabilistic sequence modeling shared library. `http://ai.ijs.si/andrej/psmslib.html`, 2008.

A. Bratko and B. Filipič. Spam filtering using compression models. Technical Report IJS-DP-9227, Department of Intelligent Systems, Jožef Stefan Institute, Ljubljana, Slovenia, 2005.

J. Carpinter and R. Hunt. Tightening the net: A review of current and next generation spam filtering tools. *Computers and Security*, 25(8):566–578, 2006.

Y. Chen and J.Z. Wang. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5:913–939, 2004.

Y. Chevaleyre and J.D. Zucker. Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. application to the mutagenesis problem. In *Proceedings of the 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 204–214, 2001.

G. V. Cormack and T. R. Lynam. Spam track guidelines — TREC 2005-2007. `http://plg.uwaterloo.ca/ gvcormac/treccorpus06/`, 2006.

N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108. ACM Press, 2004.

T.G. Dietterich, R.H. Lathrop, and T. Lozano-Pérez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence Journal*, 89(1-2):31–71, 1997.

T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

T. Gärtner, P. Flach, A. Kowalczyk, and A. Smola. Multi-instance kernels. In *Proceedings of the 19th International Conference on Machine Learning*, pages 179–186, San Francisco, CA, 2002. Morgan Kaufmann.

R. Jennings. The global economic impact of spam. Technical report, Ferris Research, 2005.

A.M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes. Multinomila naive bayes for text categorization revisited. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, pages 488–499. Springer, 2004.

J.Z. Kolter and M.A. Maloof. Using additive expert ensembles to cope with concept drift. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 449–456, New York, NY, 2005. ACM Press.

H. Lee and A. Ng. Spam deobfuscation using a hidden Markov model. In *Proceedings of the Second Conference on Email and Anti-Spam*, 2005.

P. Long and L. Tan. PAC learning axis-aligned rectangles with respect to product distribution from multiple-instance examples. *Machine Learning*, 30(1):7–21, 1998.

D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the 2005 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641–647. ACM Press, 2005a.

D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *Proceedings of the 2nd Conference on Email and Anti-Spam*, 2005b.

O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. *Advances in Neural Information Processing Systems*, 10:570–576, 1998.

J. Newsome, B. Karp, and D. Song. Paragraph: Thwarting signature learning by training maliciously. In *Recent Advances in Intrusion Detection: 9th International Symposium (RAID)*, pages 81–105, 2006.

J. Ramon and L.D. Raedt. Multi instance neural networks. In *Proceedings of ICML-2000 workshop on Attribute-Value and Relational Learning*, 2000.

S. Ray and M. Craven. Supervised versus multiple instance learning: An empirical comparison. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 697–704, New York, NY, 2005. ACM Press.

J. Rocchio Jr. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 68–73. Prentice Hall, 1971.

J. Wang and J.D. Zucker. Solving the multiple-instance learning problem: A lazy learning approach. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1119–1125, San Francisco, CA, 2000. Morgan Kaufmann.

S. Webb, S. Chitti, and C. Pu. An experimental evaluation of spam filter performance and robustness against attack. In *The 1st International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 19–21, 2005.

I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann, San Francisco, CA, USA, 2000.

X. Xu. Statistical learning in multiple instance problems. Master's thesis, University of Waikato, 2003.

X. Xu and E. Frank. Logistic regression and boosting for labeled bags of instances. In *Proceedings of the Pacific-Asian Conference on Knowledge discovery and data mining*. Springer-Verlag, 2004.

W. Yih, J. Goodman, and G. Hulten. Learning at low false positive rates. In *Proceedings of the Third Conference on Email and Anti-Spam*, 2006.

M.-L. Zhang and Z.-H. Zhou. Multi-label learning by instance differentiation. In *The 22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, pages 669–674, Vancouver, Canada, 2007.

Q. Zhang and S. Goldman. EM-DD: An improved multiple-instance learning technique. In *Proceedings of the 2001 Neural Information Processing Systems (NIPS) Conference*, pages 1073–1080, Cambridge, MA, 2002. MIT Press.

Z.H. Zhou and M.L. Zhang. Ensembles of multi-instance learners. In *ECML-03, 15th European Conference on Machine Learning*, pages 492–502, 2003.

Journal of Machine Learning Research 9 (2008) 1147-1178 Submitted 3/07; Revised 8/07; Published 6/08

Cross-Validation Optimization for Large Scale Structured Classification Kernel Methods

Matthias W. Seeger SEEGER@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstr. 38
Tübingen, Germany

Editor: Zoubin Ghahramani

Abstract

We propose a highly efficient framework for penalized likelihood kernel methods applied to multi-class models with a large, structured set of classes. As opposed to many previous approaches which try to decompose the fitting problem into many smaller ones, we focus on a Newton optimization of the complete model, making use of model structure and linear conjugate gradients in order to approximate Newton search directions. Crucially, our learning method is based entirely on matrix-vector multiplication primitives with the kernel matrices and their derivatives, allowing straightforward specialization to new kernels, and focusing code optimization efforts to these primitives only.

Kernel parameters are learned automatically, by maximizing the cross-validation log likelihood in a gradient-based way, and predictive probabilities are estimated. We demonstrate our approach on large scale text classification tasks with hierarchical structure on thousands of classes, achieving state-of-the-art results in an order of magnitude less time than previous work.

Parts of this work appeared in the conference paper Seeger (2007).

Keywords: multi-way classification, kernel logistic regression, hierarchical classification, cross validation optimization, Newton-Raphson optimization

1. Introduction

In recent years, machine learning researchers started to address problems with kernel machines which require models with a large number of dependent variables, and whose fitting demand training samples with very many cases. For example, for multi-way classification models with a hierarchically structured label space (Cai and Hofmann, 2004), modern applications call for predictions on thousands of classes, and very large data sets become available. However, if n and C denote data set size and number of classes respectively, nonparametric kernel methods like *support vector machines* (SVMs) or *Gaussian processes* (GPs) typically scale super-linearly in nC, if dependencies between the latent class functions are represented properly.

Furthermore, most large scale kernel methods proposed so far refrain from solving the problem of learning hyperparameters (kernel or loss function parameters), also known as "learning the kernels". The user has to run cross-validation schemes essentially "by hand", which is not suitable for learning more than a few hyperparameters. However, many models for modern applications come with a large number of hyperparameters (for example to represent dependencies through "mixing" as in independent components analysis), and adjusting them through optimization must make use of gradients.

We propose a general framework for learning in probabilistic kernel classification models. While the models treated here are not novel, a major feature of our approach is the high computational efficiency with which the primary fitting (for fixed hyperparameters) is done. For example, our framework applied to hierarchical classification with hundreds of classes and thousands of data points requires a few minutes for fitting. The central idea is to step back from what seems to be the dominating approach in machine learning at the moment, namely to solve a large convex optimization problem by iteratively solving very many small ones. A popular approach for these small steps is to minimize the criterion w.r.t. a few variables only, keeping the other ones fixed, and many variations of this theme have been proposed. In this paper, we focus on the opposite approach of trying to find directions which lead to fast descent, no matter how many of the variables are involved. This is essentially Newton's method, and one aspect of our work is to find approximate Newton directions very efficiently, making use of model structure and linear conjugate gradients in order to reduce the computation to standard linear algebra primitives on large contiguous chunks of memory. Interestingly, such global approaches are generally favoured in the optimization community for problems (such as kernel methods fitting) which cannot be decomposed naturally into parts. While other gradient-based optimizers such as scaled conjugate gradients could be used as well, they require more fine-tuning (for example, preconditioning) to the specific problem they are applied to, while Newton's method is closer to a "black box" technique and can be transferred to novel situations without many changes.

For multi-way classification, our primary fitting method scales linearly in C, and depends on n mainly via a fixed number of *matrix-vector multiplications* (MVM) with $n \times n$ kernel matrices. In many situations, these MVM primitives can be computed very efficiently, often without having to store the kernel matrices themselves.

We also show how to choose hyperparameters *automatically* by maximizing the cross-validation log likelihood, making use of our primary fitting technology as inner loop in order to compute the CV criterion and its gradient. It is important to note that our hyperparameter learning method works by gradient-based optimization, where the dominating part of the gradient computation does not scale with the number of hyperparameters at all.[1] The gradient computation also requires a number of MVMs with derivatives of kernel matrices, which can be reduced to kernel MVMs for many frequently used kernels (see Section 7.3). Therefore, our approach can in principle be used to learn a large number of hyperparameters without user interaction.

We apply our framework to hierarchical classification with many classes. The hierarchy is represented through an ANOVA setup. While the C latent class functions are fully dependent *a priori*, the scaling of our method stays close to what unstructured (flat) classification with C classes would require. We test our framework on the same tasks treated by Cai and Hofmann (2004), achieving comparable results in at least an order of magnitude less time.

Our proposal to use approximate Newton methods is not novel as such. The Newton method, or a variant of it called Fisher scoring, is the standard approach for fitting generalized linear models in statistics (Green and Silverman, 1994; McCullach and Nelder, 1983), at least if parametric models are fitted to moderately sized samples. Our primary fitting method for flat multi-way classification (see Section 2) appeared in Williams and Barber (1998). However, we demonstrate the usefulness of this principle on a much larger scale, showing how model structure can (and has to) be exploited

1. Such scaling behaviour is fairly standard in Gaussian process marginal likelihood maximization techniques (Williams and Barber, 1998), but has only recently been brought to attention in the SVM community (Keerthi et al., 2007).

in this context. Furthermore, we demonstrate how the secondary task of hyperparameter learning can be reduced to the same underlying primitives.

The structure of the paper is as follows. Our model and method of parameter fitting is given in Section 2. An extension to hierarchical classification is provided in Section 3, and in Section 4 we give our automatic hyperparameter learning procedure. Essential computational details are discussed in Section 5. Experimental results on a very large hierarchical text classification and several standard machine learning problems are given in Section 6. We close with a discussion in Section 7, relating our global direction approach to popular block coordinate descent techniques in Section 7.2, and pointing out future work in Section 7.4.

Optimized C++ software for our framework is available as part of the `LHOTSE` toolbox for adaptive statistical models, which is freely available for non-commercial purposes.[2] The implementation contains the linear kernel case used in Section 6.1 (see Appendix D.3), as well as a generic representation described in Appendix D.1, with which the experiments in Section 6.2, Section 6.3 have been done. It is fairly simple to include new kernels or (approximate) kernel MVM implementations.

2. Penalized Multiple Logistic Regression

In this section, we introduce our framework on a multi-way classification model with C classes, where structure between classes is not modelled. We refer to this setup as *flat classification*, in that the label set is flat (unstructured).

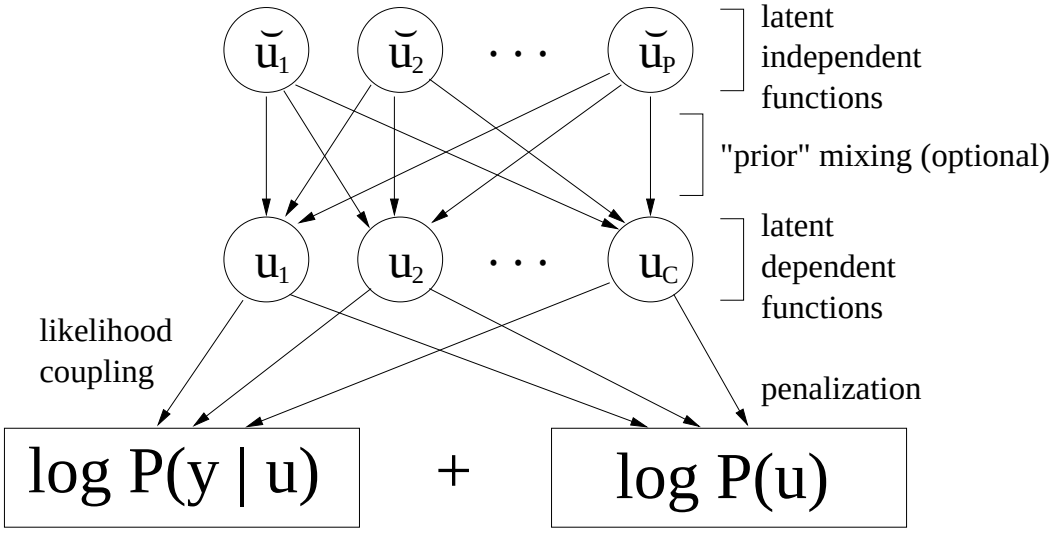


Figure 1: Structure of penalized likelihood optimization.

In general, our framework is applicable to models of the form depicted in Figure 1. A set of latent (unobserved) functions $u_c(\cdot)$ is fitted to observed data by penalized likelihood maximization. For many models, the penalisation term (also called regulariser) corresponds to the logarithm of a prior density over the $u_c(\cdot)$. This primary fitting step corresponds to a convex optimization problem over finitely many variables. Structure in such models is represented either as couplings in the log

2. Available at `www.kyb.tuebingen.mpg.de/bs/people/seeger/lhotse/`.

likelihood function, or in the penalisation (or log prior) term. The latter can be realized through the linear mixing of *a priori* independent functions $\breve{u}_p(\cdot)$, in other words the penaliser over the latter decouples w.r.t. p (our main example of such mixing is hierarchical classification, developed in Section 3).

We now apply this general framework to flat classification, where $y \in \{1,\ldots,C\}$ is to be predicted from $x \in \mathcal{X}$, given some i.i.d. data $D = \{(x_i,y_i) \,|\, i = 1,\ldots,n\}$. Our notation convention for vectors and matrices is detailed in Appendix A, where we also collect all major notational definitions in a table. We code y_i as $y_i \in \{0,1\}^C$, $1^T y_i = 1$ (zero-one coding).[3] We employ the *multiple logistic regression model*, consisting of C latent class functions $u_c(\cdot)$ feeding into the multiple logistic (or softmax) likelihood $P(y_{ic} = 1|x_i, u_i(\cdot)) = e^{u_c(x_i)}/(\sum_{c'} e^{u_{c'}(x_i)})$.

We write $u_c(\cdot) = f_c(\cdot) + b_c$ for intercept (or bias) parameters $b_c \in \mathbb{R}$ and functions $f_c(\cdot)$ living in a reproducing kernel Hilbert space (RKHS) with kernel $K^{(c)} = K^{(c)}(\cdot,\cdot)$ (Schölkopf and Smola, 2002), and consider the *penalized negative log likelihood*

$$\Phi = -\sum_{i=1}^{n} \log P(y_i|u_i) + (1/2)\sum_{c=1}^{C} \| f_c(\cdot)\|_c^2 + (1/2)\sigma^{-2}\|b\|^2, \quad u_i = (u_c(x_i))_c \in \mathbb{R}^C,$$

which we minimize for primary fitting. Here, $\|\cdot\|_c$ is the RKHS norm for kernel $K^{(c)}$. The idea is that deviations in f_c from desired functional properties encoded in $K^{(c)}$ are penalized by a large $\| f_c(\cdot)\|_c^2$. For example, for the Gaussian kernel (7), non-smooth f_c are penalized, and for the linear kernel (Appendix D.3), $\| f_c(\cdot)\|_c^2$ is the squared norm of the weight vector. Details on penalized likelihood kernel methods and RKHS penalisation can be found in Green and Silverman (1994) and Schölkopf and Smola (2002).

The model can also be understood in a Bayesian context, where the penalisation terms come from zero mean Gaussian process priors on the functions $f_c(\cdot)$, and b has a zero mean Gaussian prior with variance σ^2. From this viewpoint, we do a maximum a-posteriori (MAP) approximation here, without however taking covariances into account properly (which would be much more expensive to do). Details on Gaussian processes for machine learning can be found in Seeger (2004) and Rasmussen and Williams (2006).

Since the likelihood depends on the $f_c(\cdot)$ only through the values $f_c(x_i)$ at the data points, every minimizer of Φ must be a kernel expansion: $f_c(\cdot) = \sum_i \alpha_{ic} K^{(c)}(\cdot, x_i)$. This fact is known as representer theorem (Green and Silverman, 1994; Wahba, 1990). Plugging this in, the regulariser becomes $(1/2)\alpha^T K\alpha + (1/2)\sigma^{-2}\|b\|^2$, where $K^{(c)} = (K^{(c)}(x_i,x_j))_{i,j} \in \mathbb{R}^{n,n}$, and $K = \mathrm{diag}(K^{(c)})_c$ is block-diagonal. The kernels $K^{(c)}$ can in general be different, although sharing kernels among classes can lead to computational savings, in that some of the blocks $K^{(c)}$ are identical. Our implementation of block sharing is described in Appendix D.1.

We show in Section 5.1.1 that the b_c may be eliminated as $b = \sigma^2(I \otimes 1^T)\alpha$. Thus, if $\tilde{K} = K + \sigma^2(I \otimes 1)(I \otimes 1^T)$, then our criterion Φ becomes

$$\Phi = \Phi_{lh} + \frac{1}{2}\alpha^T \tilde{K}\alpha, \quad \Phi_{lh} = -y^T u + 1^T l, \quad l_i = \log 1^T \exp(u_i), \quad u = \tilde{K}\alpha. \tag{1}$$

Φ is strictly convex in α, being a sum of linear, quadratic, and *logsumexp* terms of the form $\log 1^T \exp(u_i)$ (Boyd and Vandenberghe, 2002), so it has a unique minimum point $\hat{\alpha}$. The corre-

3. We switch between the formats y_i, y_i. Note that y_{ic} denotes a component in $y_i = (y_{ic})_c$.

sponding kernel expansions are

$$\hat{u}_c(\cdot) = \sum_i \hat{\alpha}_{ic}(K^{(c)}(\cdot, x_i) + \sigma^2).$$

Estimates of the conditional probability on test points x_* are obtained by plugging $\hat{u}_c(x_*)$ into the likelihood. These estimates are asymptotically consistent, although better finite sample estimates could probably be obtained by a more Bayesian treatment.

We note that this setup is related to the multi-class SVM (Crammer and Singer, 2001), where $-\log P(y_i|u_i)$ is replaced by the margin loss $-u_{y_i}(x_i) + \max_c\{u_c(x_i) + 1 - \delta_{c,y_i}\}$. Here, $\delta_{a,b} = \mathrm{I}_{\{a=b\}}$. The negative log multiple logistic likelihood has similar properties, but is smooth as a function of u, and the primary fitting of α does not require constrained convex optimization. Furthermore, universal consistency for estimates of $P(y_*|x_*)$ can be established for the multiple logistic loss, but fails to hold for the SVM variant (Bartlett and Tewari, 2004).

We will minimize Φ using the *Newton-Raphson* (NR) algorithm. The computation of Newton search directions requires solving a system with the Hessian and the gradient of Φ, which we will do approximately using the *linear conjugate gradients* (LCG) algorithm. This can be done without fully computing, storing, or inverting the Hessian, all of which would not be possible for large nC. In fact, the task is reduced to computing $k_1(k_2+2)$ MVMs with K, where k_1 is the number of NR iterations, k_2 the number of LCG steps for computing each Newton direction. Since NR is a second-order convergent method, k_1 is generally small. k_2 determines the quality of each Newton direction, and again, fairly small values seem sufficient (see Section 6.1). Details are provided in Section 5.1.

Finally, some readers may wonder why we favour the NR algorithm here, which in practice can be fairly complicated to implement, while we could do a simpler gradient-based optimization of Φ w.r.t. α, for example by scaled (non-linear) conjugate gradients (SCG). The problem is that on tasks of the size we want to address, non-invariant methods such as SCG tend to fail completely if not properly preconditioned, and we experienced exactly that in preliminary experiments. In contrast to that, NR is invariant to the choice of optimization variables, so does not have to be preconditioned. It is by far the preferred method in the optimization literature (Bertsekas, 1999; Boyd and Vandenberghe, 2002), and many ideas for preconditioning or Quasi-Newton try to approximate the NR directions. We think that a proper SCG implementation can be at least as efficient as NR, but needs fine-tuning to the specific problem, which in the case of hierarchical classification (discussed next) is already quite difficult. More details on this point are given in Section 5.4 and also Section 7.2.

3. Hierarchical Classification

So far we dealt with flat classification, the classes being independent *a priori*, with block-diagonal kernel matrix K. However, if the label set has a known structure,[4] we can benefit from representing it in the model. Here we focus on hierarchical classification, the label set $\{1,\ldots,C\}$ being the leaf nodes of a tree. Classes with lower common ancestor should be more closely related. In this section, we propose a model for this setup and show how it can be dealt with in our framework with minor modifications and reasonable extra cost.

In flat classification, the latent class functions $u_c(\cdot)$ are modelled as *a priori* independent, in that the penaliser (or the log prior in the GP view) is a sum of individual terms for each c, without

4. Learning an unknown label set structure may be achieved by expectation maximization techniques, but this is subject to future work.

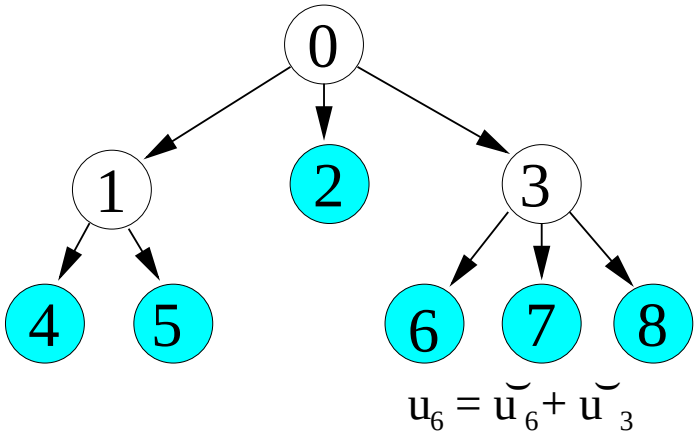


Figure 2: Example of a tree-structured target space, where labels correspond to leaf nodes (shaded).

interaction terms. Analysis of variance (ANOVA) models go beyond such independent designs, they have previously been applied to text classification by Cai and Hofmann (2004), see also Shahbaba and Neal (2007). Let $\{0,\ldots,P\}$ be the nodes of the tree, 0 being the root, and the numbers are assigned breadth first ($1,2,\ldots$ are the root's children). The tree is determined by P and n_p, $p = 0,\ldots,P$, the number of children of node p. Let L be the set of leaf nodes, $|L| = C$. Assign a *pair* of latent functions $u_p, \breve{u}_p$ to each node, except the root. The $\breve{u}_p$ are assumed *a priori* independent, as in flat classification. u_p is the sum of $\breve{u}_{p'}$, where p' is running over the nodes (including p) on the path from the root to p. An example is given in Figure 2. The class functions to be fed into the classification likelihood are the $u_{L(c)}$ of the leafs. This setup represents similarities according to the hierarchy. For example, if leafs $L(c), L(c')$ have the common parent p, then $u_{L(c)} = u_p + \breve{u}_{L(c)}$, $u_{L(c')} = u_p + \breve{u}_{L(c')}$, so the class functions *share* the effect u_p. Since regularisation forces all independent effects $\breve{u}_{p'}$ to be smooth, the classes c, c' are urged to behave similarly *a priori*.

Let $u = (u_p(x_i))_{i,p}$, $\breve{u} = (\breve{u}_p(x_i))_{i,p} \in \mathbb{R}^{nP}$. The vectors are linearly related as $u = (\Phi \otimes I)\breve{u}$, $\Phi \in \{0,1\}^{P,P}$, a special case of the mixing of Figure 1. Importantly, Φ has a simple structure which allows MVM with Φ or Φ^T to be computed easily in $O(P)$, without having to compute or store Φ explicitly. Let $cs_p = \sum_{p'<p} n_{p'}$, and define $\Phi_p \in \mathbb{R}^{d,d}$, $d = cs_p + n_p$, to be the upper left block of Φ, so that $\Phi = \Phi_P$. If p is a leaf node, then $\Phi_p = \Phi_{p-1}$. Otherwise, Φ_p is obtained from Φ_{p-1} by attaching rows $(\delta_p^T \Phi_{p-1}, \delta_j^T)$, $j = 1,\ldots,n_p$, where $\delta_p^T \Phi_{p-1}$ is the p-th row of Φ_{p-1}. This is because $u_{cs_p+j} = u_p + \breve{u}_{cs_p+j}$ for the functions of the children of p. Formally,

$$\Phi_p = \begin{pmatrix} \Phi_{p-1} & 0 \\ 1\delta_p^T \Phi_{p-1} & I \end{pmatrix},$$

where the lower right $I \in \mathbb{R}^{n_p,n_p}$. Note that Φ is lower triangular with $\operatorname{diag}\Phi = I$. This recursive definition directly implies simple methods for computing $v \mapsto \Phi v$ and $v \mapsto \Phi^T v$.

Under the hierarchical model, the class functions $u_{L(c)}$ are strongly dependent *a priori*. Representing this prior coupling in our framework amounts to simply plugging in the implied kernel matrix

$$K = (\Phi_{L,\cdot} \otimes I)\breve{K}(\Phi_{L,\cdot}^T \otimes I), \tag{2}$$

into the flat classification model of Section 2. Here, the inner $\check{K}$ is block-diagonal, while in the flat model, K itself had this property. In the hierarchical case, K is not sparse and certainly not block-diagonal, but we are still able to compute kernel MVMs efficiently: pre- and post-multiplying by Φ is very cheap, and $\check{K}$ is block-diagonal just as in the flat case.

In fact, the step from flat to hierarchical classification requires minor modifications of existing code only. If code for representing a block-diagonal K is available, we can use it to represent the inner $\check{K}$, just replacing C by P. This simplicity carries through to the hyperparameter learning method (see Section 4). The cost of a kernel MVM is increased[5] by a factor $P/C < 2$, which in most hierarchies in practice is close to 1.

However, it would be wrong to claim that hierarchical classification in general comes as cheap as flat classification. In fact, primary fitting becomes more costly, precisely because there is more coupling between the variables. In the flat case, the Hessian of Φ (1) is close to block-diagonal. The LCG algorithm to compute Newton directions converges quickly, because it nearly decomposes into C independent ones, and fewer NR steps are required. In the hierarchical case, this "near-decomposition" does not hold, and both LCG and NR need more iterations to attain the same accuracy, although each LCG step comes at about the same cost as in the flat case.

In numerical mathematics, much work has been done to approximately decouple linear systems by preconditioning. In some of these strategies, knowledge about the structure of the system matrix (in our case: the hierarchy) can be used to drive preconditioning. An important point for future research is to find a good preconditioning strategy for the system (5). However, in all our experiments so far the fitting of the hierarchical model took less than twice the time required for the flat model on the same task.

4. Hyperparameter Learning

Our framework comes with an automatic method for setting free hyperparameters h, by gradient-based maximization of the cross-validation (CV) log likelihood. Our primary fitting method of Section 2 is used here as principal subroutine. Such a setup is commonplace in Bayesian statistics, where (marginal) inference is typically employed as subroutine in parameter learning.

Recall that primary fitting works by minimizing Φ (1) w.r.t. α. Let $\{I_k\}$ be a partition of the data set range $\{1,\ldots,n\}$, with $J_k = \{1,\ldots,n\} \setminus I_k$, and let

$$\Phi_{J_k} = u_{[J_k]}^T((1/2)\alpha_{[J_k]} - y_{J_k}) + 1^T l_{[J_k]}$$

be the negative log likelihood of the subset J_k of the data. Here, $u_{[J_k]} = \tilde{K}_{J_k}\alpha_{[J_k]}$. The $\alpha_{[J_k]}$ are independent variables, *not* part of a common[6] α. The cross-validation criterion is

$$\Psi = \sum_k \Psi_{I_k}, \quad \Psi_{I_k} = -y_{I_k}^T u_{[I_k]} + 1^T l_{[I_k]}, \quad u_{[I_k]} = \tilde{K}_{I_k,J_k}\alpha_{[J_k]}, \tag{3}$$

where $\alpha_{[J_k]}$ is the minimizer of Φ_{J_k}. Since for each k, we fit and evaluate the likelihood on disjoint parts of y, Ψ is an unbiased estimator of the true negative expected log likelihood.

In order to adjust h, we pick a fixed partition at random, then do gradient-based minimization of Ψ w.r.t. h. To this end, we maintain the set $\{\alpha_{[J_k]}\}$ of primary variables, and iterate between re-fitting

5. Nodes with a single child only can be pruned from the hierarchy. Note that our formalism does not require all leaf nodes to have the same depth.

6. Which is why they are not referred to as α_{J_k}.

those for each fold k, and computing Ψ and $\nabla_h \Psi$. The gradient can be determined analytically, using a computation which is equivalent to the Newton direction computations for $\alpha_{[J_k]}$, meaning that the same code can be used. Details are given in Section 5.2. Note that Ψ is not a convex objective.

As for computational complexity, suppose there are q folds. The update of the $\alpha_{[J_k]}$ requires q primary fitting applications, but since they are initialized with the previous values $\alpha_{[J_k]}$, they do converge very rapidly, especially during later iterations. Computing Ψ based on the $\alpha_{[J_k]}$ comes basically for free. The gradient computation decomposes into two parts: accumulation, and kernel derivative MVMs. The accumulation part requires solving q systems of size $((q-1)/q)nC$, thus qk_3 kernel MVMs on the $\tilde{K}_{J_k}$ if linear conjugate gradients (LCG) is used, k_3 being the number of LCG steps. We also need two buffer matrices E, F of qnC elements each. Note that the accumulation step is *independent* of the number of hyperparameters. The second part consists of q kernel derivative MVMs for each independent component of h. This second part is much simpler than the accumulation one, consisting entirely of large matrix operations, which can be run very efficiently using specialized numerical linear algebra code. The method for computing Ψ and $\nabla_h \Psi$ can be plugged into a custom gradient-based optimizer, such as Quasi-Newton or conjugate gradients, in order to learn h.

As shown in Section 5.3, the extension of hyperparameter learning to the hierarchical case of Section 3 is done by wrapping the accumulation part with Φ MVMs, the coding and additional memory effort being minimal.

We finally note from our findings in practice (see Section 6.3) that on large tasks, our automatic method can require some fine-tuning. This is due to the delicate dependencies between the different approximations used. The accuracy of Ψ and $\nabla_h \Psi$ depends on how accurate the inner NR optimizations for $\alpha_{[J_k]}$ turn out, and the latter depend on how many iterations of LCG are done in order to compute search directions. Fortunately, Φ_{J_k} and its gradient w.r.t. $u_{[J_k]}$ can be computed exactly in order to assess inner optimization convergence, so we do at least know when things go wrong. In our implementation, we deem an evaluation of Ψ and $\nabla_h \Psi$ usable if the average of $\|\nabla_{u_{[J_k]}} \Phi_{J_k}\|$ over folds is below a threshold, which depends on the problem and on time constraints. A failed evaluation leads to a right bracket there for the outer optimization line search, in that step sizes beyond the failed one are not accessed. We can now tune the basic running time parameters k_1, k_2 so that Ψ evaluations do not fail too often. In this context, it is important to regard the $\{\alpha_{[J_k]}\}$ as an *inner state* alongside the hyperparameter vector h. Although inner optimizations are convex, for large problems and reasonable k_1, k_2, successive minima are attained only when we start from the previous best inner state. This is true especially during later stages, where for certain problems (see Section 6.3) h attends "extreme" values and the inner optimizations become quite hard.[7] Therefore, the inner state used to initialize a given Ψ evaluation is the final one for the last recent *successful* evaluation.[8] Inner states attained during failed evaluations are discarded.

7. Although inner optimizations are convex, speed of convergence of NR depends strongly on the value of h. For "extreme" values, the Newton direction computation by LCG is harder, and search directions can become large in early NR iterations. The latter may be because we work in u rather than α space, but only the former is really feasible.
8. Within outer line searches, we use $\{\alpha_{[J_k]}\}$ from the last recent successful evaluation *to the left* (along the search direction).

5. Computational Details

In this section, we provide details for the material above. The techniques given here do characterize our framework, they are novel in this combination, and some of them may be useful in other contexts as well. More specific details of our implementation can be found in Appendix D.

5.1 Details for Flat Classification

In this section, we provide details for the primary fitting optimization in the case of flat multi-way classification, introduced in Section 2. Note that this fitting method appeared in Williams and Barber (1998) in the context of approximate Gaussian process inference, although some fairly essential ideas here are novel to our knowledge (symmetrisation of Newton system, pair optimization line search, numerical stability considerations).

Recall that we want to minimize the strictly convex criterion Φ (1) w.r.t. α, using the Newton-Raphson (NR) method. Modern variants of this algorithm iterate line searches along the *Newton directions* $-H^{-1}g$, where g, H are gradient and Hessian of Φ at the current α. We will start with the Newton direction computation in Section 5.1.1, commenting on the line searches afterwards in Section 5.1.2 (it turns out that it basically comes for free). An overview of the fitting algorithm is given in Section 5.1.3.

5.1.1 COMPUTING THE NEWTON DIRECTION

Recall Φ and related variables from (1). Let $\pi_{ic} = P(y_{ic} = 1|u_i)$, $\pi = \exp(u - 1 \otimes l)$, and recall that Φ_{lh} is the likelihood part in Φ. Now,

$$g := \nabla\Phi_{lh} = \pi - y,\ W := \nabla\nabla\Phi_{lh} = D - DP_{cls}D,\ P_{cls} = (1 \otimes I)(1^T \otimes I).$$

Here, $D = \operatorname{diag}\pi$, and gradient and Hessian are taken w.r.t. u (*not* w.r.t. α). Our convention for nC vectors and matrices and the use of $\otimes$ is explained in Appendix A. The form of W can be understood by noting that W is block-diagonal in a *different* ordering, which uses c (classes) as inner and i (data points) as outer index, then switching to our standard ordering.

It is easy to compute gradient and Hessian of Φ w.r.t. α, b. A full (classical) Newton step is given by the system

$$\begin{aligned}
&(I + WK)\alpha' + W(I \otimes 1)b' = Wu - g, \\
&(I \otimes 1^T)WK\alpha' + (I \otimes 1^T)W(I \otimes 1)b' + \sigma^{-2}b' = (I \otimes 1^T)(Wu - g),
\end{aligned}$$

and the Newton search direction is obtained as the difference $\alpha' - \alpha, b' - b$. Subtracting $(I \otimes 1^T)$ times the first from the second, we obtain $b' = \sigma^2(I \otimes 1^T)\alpha'$, and plugging this into the first equation, we have

$$\left(I + W\left(K + \sigma^2 P_{data}\right)\right)\alpha' = Wu - g, \quad P_{data} = (I \otimes 1)(I \otimes 1^T). \tag{4}$$

Note that $P_{data}a = (\sum_{i'} a_{i'})_i$, which does the same as P_{cls}, but on index i rather than c. We denote

$$\tilde{K} = K + \sigma^2 P_{data},$$

noting that this corresponds to $\tilde{K}^{(c)} = K^{(c)} + \sigma^2 11^T$. The correct way of incorporating intercept parameters is to add the constant σ^2 to the kernels, then to obtain $b_c = \sigma^2 \sum_i \alpha_{ic}$. This is the meaning

of "eliminating b" in Section 2. While we could optimize σ^2 as a hyperparameter, we consider it fixed and given for simplicity.[9] In the sequel, we consider b being eliminated from the model by replacing $K \to \tilde{K}$ everywhere. We have $u = \tilde{K}\alpha$.

We can solve the system (4) exactly if we can tolerate a scaling of $O(n^3C)$ and $O(n^2C)$ memory. Note that this scaling is linear rather than cubic in C. The exact solution is derived in Appendix C. It is efficient for moderate n, and generally useful for code debugging, and is supported by our implementation. In the remainder of this section, we focus on approximate computations.

Although we could solve the system using a bi-conjugate gradients solver, we can do much better by transforming it into symmetric positive definite form. First, note that W is positive semidefinite, but singular. This can be seen by noting that the parameterization of our likelihood in terms of u_i is overcomplete, in that $u_i + \kappa 1$ gives the same likelihood values for all κ. We could fix one of the u_i components, which would however lead to subtle dependencies between the remaining $C-1$ functions $u_c(\cdot)$. In order to justify our *a priori* independent treatment of these functions, we have to retain the overcomplete likelihood. The nullspace $\ker W$ is given by $\{(d)_c \,|\, d \in \mathbb{R}^n\}$ and has dimension n. This can be seen by noting that $Wa = 0$ iff $a = (\bar{a})_c$, $\bar{a} = \sum_{c'} a^{(c')}$. W has rank $n(C-1)$. We have $a \in \mathrm{ran}\, W$ iff $\sum_c a^{(c)} = (1^T \otimes I)a = 0$ (recall that $\ker W$ and $\mathrm{ran}\, W$ are orthogonal, and their direct sum is $\mathbb{R}^{nC}$). From (4) we see that $\alpha' + g$ lies in $\mathrm{ran}\, W$. Note that $\sum_c g^{(c)} = \sum_c (\pi^{(c)} - y^{(c)}) = 1 - 1 = 0$, therefore $g \in \mathrm{ran}\, W$, thus $\alpha' \in \mathrm{ran}\, W$. We see that the dual coefficients must fulfill the constraint $\alpha \in \mathrm{ran}\, W$. Note that $\mathrm{ran}\, W$ is in fact independent of D. Whatever starting value is used for α, it should be projected onto $\mathrm{ran}\, W$, which is done by subtracting $C^{-1}P_{cls}\alpha$. The NR updates then make sure that the constraint remains fulfilled.

Next, we need a decomposition $W = VV^T$ of W. Such a V exists (because W is positive semidefinite). In fact,

$$W = ADA^T, \quad A = I - DP_{cls}.$$

This follows easily from $(1^T \otimes I)D(1 \otimes I) = \sum_{c'} D^{(c')} = I$. Thus, $W = VV^T$ with $V = AD^{1/2}$. The matrix A has fixed points $\mathrm{ran}\, W$, namely if $a \in \mathrm{ran}\, W$, then $(1^T \otimes I)a = 0$, so that $Aa = a$.

Since there exists some $\tilde{v}$ (not unique) s.t. $\alpha' = W\tilde{v}$, we can rewrite the system (4) as

$$V\left(I + V^T\tilde{K}V\right)V^T\tilde{v} = V\left(V^Tu - \tilde{g}\right),$$

where $\tilde{g}$ is s.t. $g = V\tilde{g}$ (such a vector exists because $g \in \mathrm{ran}\, W$). This suggests the following procedure for finding α':

$$\left(I + V^T\tilde{K}V\right)\beta = V^Tu - \tilde{g}, \quad \alpha' = V\beta. \tag{5}$$

To see the validity of this approach, simply multiply both sides of (5) by V from the left, which shows that $V\beta$ solves the original system. Since the latter has a unique solution (strict convexity!), we must have $V\beta = \alpha'$. Finally, we note that $\tilde{g} = D^{-1/2}g$ does the job, because $VD^{-1/2}g = Ag = g$. The latter follows because $g \in \mathrm{ran}\, W$.

Thus, in exact arithmetic, the Newton direction computation is implemented in a three-stage procedure. First, compute $\tilde{g} = D^{-1/2}g$. Second, solve the system (5) for β. This is a symmetric positive definite system with the typically well-conditioned matrix $I + V^T\tilde{K}V$, and can be solved efficiently using the linear conjugate gradients (LCG) algorithm (Saad, 1996). The cost of each step is dominated by the MVM $v \mapsto Kv$, which scales linearly in C, due to the block-diagonal structure of K. Third, set $\alpha' = V\beta$. The Newton direction is obtained as $\alpha' - \alpha$.

9. In our experience so far, a good value of σ^2 is fairly robust across different tasks for the same problem, but may differ strongly between different problems. It can be chosen based on some initial experiments.

We can start the LCG run from a good guess as follows. Let α be the current dual vector which solved the last recent system. We would like to initialize β s.t. $\alpha = V\beta = AD^{1/2}\beta$. If we assume that $D^{1/2}\beta \in \mathrm{ran}\, W$, then $\alpha = D^{1/2}\beta$. Therefore, a good initialization is $\beta = D^{-1/2}\alpha$. Alternatively, we may also retain β from the last recent system.

Issues of numerical stability are addressed in Appendix B. Furthermore, the LCG algorithm is hardly ever run without some sort of preconditioning. Our present implementation uses diagonal preconditioning, as described in Appendix B. We have already noted in Section 3 that a non-diagonal preconditioning strategy could be valuable, but this is subject to future work.

5.1.2 THE LINE SEARCH

The classical NR algorithm proceeds doing full steps $\alpha \to \alpha'$, but modern variants typically employ a line search along the Newton direction $\alpha' - \alpha$. In the non-convex case, this ensures global convergence, and even for our convex objective Φ, a line search saves time and leads to numerically more stable behaviour. Interestingly, the special structure of our problem leads to the fact that line searches essentially come for free, certainly compared with the effort of obtaining Newton directions. We refer to this simple idea as *pair optimization*, the reader may be reminded of similar tricks in primal-dual schemes for SVM.

Let $s = \alpha' - \alpha$ be the NR direction, computed as shown above, and set α_0 to α. The line search minimizes (or sufficiently decreases) Φ on the line segment $\alpha_0 + \lambda s$, $\lambda \in (0,1]$, starting with $\lambda = 1$ (which is the classical Newton step). The idea is to treat Φ as a function of the pair (u, α), where $u = \tilde{K}\alpha$. The corresponding line segment is $u = u_0 + \lambda\tilde{s}$, $\tilde{s} = \tilde{K}s$, requiring a single kernel MVM for computing $\tilde{s}$. Let $j = \mathrm{argmax}\,|\tilde{s}_j|$. For an evaluation of Φ at u, we reconstruct $\lambda = (u_j - u_{0,j})/\tilde{s}_j$ and $\alpha = \alpha_0 + \lambda s$, then

$$\Phi = u^T\left((1/2)\alpha - y\right) + 1^T l, \quad \nabla\Phi = \pi - y + \alpha,$$

so that an evaluation comes at the cost $O(nC)$ and does not require additional kernel MVM applications. We now do the line minimization of Φ in the variable u. The driving feature of pair optimization is that we can go back and forth between α and u without significant cost, once the search direction is known w.r.t. both variables.

5.1.3 OVERVIEW OF THE OPTIMIZATION ALGORITHM

In Algorithm 1, we give a schematic overview of the primary fitting algorithm, written in terms of a MVM primitive $v \mapsto Kv$. For simplicity, we do not include the measures discussed in Appendix B to increase numerical stability.

5.2 Details for Hyperparameter Learning

In this section, we provide details for the CV hyperparameter learning scheme, introduced in Section 4. The gradient of the CV criterion Ψ (3) is computed as follows. Ψ is a sum of terms Ψ_{I_k}, one for each fold. We focus on a single term and write $I = I_k$, $J = J_k$. $\alpha_{[J]}$ is determined by the stationary equation $\alpha_{[J]} + g_{[J]} = 0$ (all terms of subscript $[J]$ are as in Section 5.1.1, but for the subset J of the data, and w.r.t. $\alpha_{[J]}$). Taking derivatives gives

$$d\alpha_{[J]} = -W_{[J]}\left((dK_J)\alpha_{[J]} + \tilde{K}_J(d\alpha_{[J]})\right),$$

Algorithm 1 Newton-Raphson optimization to find posterior mode $\hat{\alpha}$.

Require: Starting values for α, b. Targets y.
  $\alpha = \alpha - C^{-1}(\sum_{c'} \alpha^{(c')})_c$, so that $\alpha \in \operatorname{ran} W$. $u = \tilde{K}\alpha$.
  repeat
    Compute $l, \log(\pi)$ from u. Compute Φ.
    if relative improvement in Φ small enough **then**
      Terminate outer loop.
    else if maximum number of iterations done **then**
      Terminate outer loop.
    end if
    Initialize $\beta = D^{-1/2}\alpha$. Compute r.h.s. $r = V^T u - \tilde{g}$, $\tilde{g} = D^{-1/2} g$.
    Compute preconditioner $\operatorname{diag}(I + V^T \tilde{K} V)$.
    Run preconditioned CG algorithm in order to solve the system (5) approximately. The CG code is configured by a primitive to compute $v \mapsto (I + V^T \tilde{K} V)v$, which in turn calls the primitive for $v \mapsto Kv$.
    Compute $\alpha' = AD^{1/2}\beta'$.
    Do line search along $s = \alpha' - \alpha$. This is done in u, along $\tilde{s} = \tilde{K}s$.
    Assign line minimizer to α, u.
  until forever

since $dg_{[J]} = W_{[J]} du_{[J]}$. We obtain a system for $d\alpha_{[J]}$ which is symmetrised as in Section 5.1.1:

$$\left(I + V_{[J]}^T \tilde{K}_J V_{[J]}\right)\beta = -V_{[J]}^T (dK_J)\alpha_{[J]}, \quad d\alpha_{[J]} = V_{[J]}\beta.$$

Also,

$$d\Psi_I = \left(\pi_{[I]} - y_I\right)^T \left((dK_{I,J})\alpha_{[J]} + \tilde{K}_{I,J}(d\alpha_{[J]})\right).$$

With

$$f = I_{\cdot,I}(\pi_{[I]} - y_I) - I_{\cdot,J} V_{[J]} \left(I + V_{[J]}^T \tilde{K}_J V_{[J]}\right)^{-1} V_{[J]}^T \tilde{K}_{J,I}(\pi_{[I]} - y_I),$$

we have that $d\Psi_I = (I_{\cdot,J}\alpha_{[J]})^T (dK) f$.

If we collect these vectors as columns of $E, F \in \mathbb{R}^{nC,q}$, q the number of folds, we have that

$$d\Psi = \operatorname{tr} E^T (dK) F \tag{6}$$

for the complete criterion. The computation of E, F was called "accumulation" in Section 4. It involves a loop over folds, in which $\alpha_{[J_k]}$ is determined by NR optimization, starting from its previous value, then f (column of F) is computed by solving one more system of the same form as is required to compute Newton directions. Importantly, this accumulation phase is independent of the number of hyperparameters. The gradient computation then requires to compute (6) for each component, using kernel derivative MVMs. First of all, $\partial K / \partial h_p$ is block-diagonal just as K, and for many standard kernels, it is a simple expression, involving K itself (see Section 7.3), so one may be able to share computations between the different gradient components. Importantly, the computation of (6) is easily broken down into large numerical linear algebra primitives, for which very efficient code may be used (see Section 7.2). This is a significant advantage in the presence of

many hyperparameters. For moderately many hyperparameters, the accumulation clearly dominates the CV criterion and gradient computation.

The dominating part of the accumulation is the re-optimisation of the $\alpha_{[J]}$, which are done by calling the optimized code for primary fitting (Section 5.1) as subroutine. Here, a feature of our implementation becomes important. Instead of representing each K_{J_k} separately, we represent the full K only for all subset kernel MVMs. The representation depends on the covariance function, and in general on how kernel MVMs are actually done. A generic representation is described in Appendix D.1. In order to work on the data subset J_k, we *shuffle* the representation such that in the permuted kernel matrix, K_{J_k} forms the upper left corner. This means that linear algebra primitives with K_{J_k} can be run without mapping matrix coordinates through an index, which would be many times slower. Details on "covariance shuffling" are given in Appendix D.2.

As mentioned in Section 5.1.1 and detailed in Appendix C, we can also compute Newton directions exactly in $O(Cn^3)$ in the flat classification case. This exact treatment can be extended to the computation of Ψ and its gradient, as is shown in Appendix C. Exact computations lead to more robust behaviour, and may actually run faster for small to moderate n. Exact computations are also useful for debugging purposes.

5.3 Details for Hierarchical Classification

In this section, we provide details for hierarchical classification method, introduced in Section 3. Recall that $u = (\Phi \otimes I)\check{u}$ for an indicator matrix Φ of simple structure, and that MVM with Φ or Φ^T can be computed easily in $O(P)$, without having to store Φ. Since the $\check{u}_p(\cdot)$ are given independent priors (or regularisers), the corresponding kernel matrix $\check{K}$ is block-diagonal. The induced covariance matrix K over u_L is given by (2), and hierarchical classification differs from the flat variant only in that this non-block-diagonal matrix is used.

The MVM primitive $v \mapsto Kv$ is computed in three steps. MVM with $(\Phi_{L,\cdot} \otimes I)$ and $(\Phi_{L,\cdot}^T \otimes I)$ works by computing $S \mapsto S\Phi$, $S \mapsto S\Phi^T$ for $S \in \mathbb{R}^{n,P}$. In between, MVM with $\check{K}$ has to be done in the same way as for flat classification, only that $\check{K}$ has P rather than C diagonal blocks.

The diagonal preconditioning of LCG (see Appendix B) requires the computation of $\operatorname{diag} K \in \mathbb{R}^{nC}$. We have

$$K_{ic,ic} = (\delta_p^T\Phi \otimes \delta_i^T)\check{K}(\Phi^T\delta_p \otimes \delta_i) = \delta_p^T\Phi(\operatorname{diag}(\check{K}_i^{(p')})_{p'})\Phi^T\delta_p, \quad p = L(c),$$

where $\delta_p^T\Phi$ is the p-th row of Φ. From the recursive structure of Φ we know that if $n_p > 0$, then $\delta_{cs_p+j}^T\Phi = \delta_p^T\Phi + \delta_{cs_p+j}^T$, $j = 1,\ldots,n_p$, so if

$$d_{i(cs_p+j)} = d_{ip} + \check{K}_i^{(cs_p+j)}, \ j = 1,\ldots,n_p,$$

then $\operatorname{diag} K = d_L$.

Hyperparameter learning (see Section 5.2) is easily extended to the hierarchical case, recalling (2) and the fact that Φ does not depend on hyperparameters. Define $\tilde{E} = (\Phi_{L,\cdot}^T \otimes I)E \in \mathbb{R}^{nP,q}$, $\tilde{F}$ accordingly, with E, F given in Section 5.2. The gradient components (6) translate to $\operatorname{tr}\tilde{E}^T(d\check{K})\tilde{F}$, where $\check{K}$ is block-diagonal as before. In our implementation, we reserve buffer space for $\tilde{E}, \tilde{F}$, yet build E, F there during accumulation. We then transform them to $\tilde{E}, \tilde{F}$ using in-place computations.

The step from flat to hierarchical classification requires only minor modifications of existing code. Wrappers for MVM and the other primitives essentially pre- and post-multiply their input

with Φ and Φ^T respectively, calling the existing "flat" primitives for $\check{K}$ in between (block-diagonal with P rather than C blocks).

5.4 Why Newton Raphson?

Why do we propose to use the second-order NR method for minimizing Φ, instead of using a simpler gradient-based technique such as scaled conjugate gradients (SCG)? We already motivated our choice at the end of Section 2, but give more details concerning this important point here.

The convex problems we are interested in here live in very high-dimensional spaces and come with complicated couplings between the components which cannot be characterized simply. Certainly, there is no simple decomposition into parts. It is well known in the optimization literature (Bertsekas, 1999) that simple gradient-based techniques such as SCG require well-chosen preconditioning in order to work effectively in such cases.

For example, we could optimize Φ (1) w.r.t. α directly, the gradient requires a single MVM with K rather than solving a system. However, this problem is very ill-conditioned, the Hessian being $\tilde{K}W\tilde{K}+\tilde{K}$ (large kernel matrices are typically very ill-conditioned, and here we deal with K^2), and SCG runs exceedingly slowly to the point of being essentially useless (as we determined in experiments). It can be saved (to our knowledge) only by preconditioning, which in our case requires to solve a system again. Another idea is to optimize Φ w.r.t. u by SCG, which works better. The Hessian is $W+\tilde{K}^{-1}$, whose condition number is similar to K. In preliminary direct comparisons, the NR method still works more efficiently, meaning that SCG would require additional preconditioning specific to the problem at hand, which would likely be different for flat and hierarchical classification. From our experience, and also from the predominance of NR in the optimization literature, we opted for this method which comes with self-tuning capabilities, making it easier to transfer the framework to novel problems.

6. Experiments

In this section, we provide experimental results for our method on a range of flat and hierarchical classification tasks.

6.1 Hierarchical Classification: Patent Texts

We use the WIPO-alpha collection,[10] many thanks to L. Cai, T. Hofmann for providing us with the count data and dictionary. We did Porter stemming, stop word removal, and removal of empty categories. The attributes are bag-of-words over the dictionary. All input vectors x_i were scaled to unit norm. Many thanks to Peter Gehler for helping us with the preprocessing.

These tasks have previously been studied by Cai and Hofmann (2004), where patents (title and claim text) are to be classified w.r.t. the standard taxonomy *IPC*, a tree with 4 levels and 5229 nodes. Sections A, B,..., H form the first level. As in Cai and Hofmann (2004), we concentrate on the 8 subtasks rooted at the sections, ranging in size from D ($n = 1140, C = 160, P = 187$) to B ($n = 9794, C = 1172, P = 1319$). We use linear kernels (see Appendix D.3) with variance parameters v_c.

All experiments are averaged over three training/test splits, different methods using the same ones. The CV criterion Ψ is used with a different (randomly drawn) 5-partition per section and

10. Available at `www.wipo.int/tools/en/dbindex.html`, or google for "Data Collections hosted by WIPO".

split, the same across all methods. Our method outputs a predictive distribution $p_j \in \mathbb{R}^C$ for each test case x_j. The standard prediction $y(x_j) = \text{argmax}_c\, p_{jc}$ maximizes expected accuracy, classes are ranked as $r_j(c) \leq r_j(c')$ iff $p_{jc} \geq p_{jc'}$, where $r_j(c) \in \{1,\ldots,C\}$ is the rank of class c for case x_j. Let y_j denote the true label for x_j. The test scores we use here are the same as in Cai and Hofmann (2004): *accuracy* (acc) $m^{-1}\sum_j \mathrm{I}_{\{y(x_j)=y_j\}}$, *precision* (prec) $m^{-1}\sum_j r_j(y_j)^{-1}$, *parent accuracy* (pacc) $m^{-1}\sum_j \mathrm{I}_{\{\text{par}(y(x_j))=\text{par}(y_j)\}}$, par$(c)$ being the parent of leaf node $L(c)$ (recall that $L(c)$ corresponds to class c). Here, m is the test set size. Let $\Delta(c,c')$ be half the length of the shortest path between leafs $L(c), L(c')$. The *taxo-loss* (taxo) is $m^{-1}\sum_j \Delta(y(x_j), y_j)$. These scores are motivated in Cai and Hofmann (2004). For taxo-loss and parent accuracy, we better choose $y(x_j)$ to minimize expected loss,[11] which is different in general than the standard prediction (the latter maximizes expected accuracy and precision).

We compare methods F1, F2, H1, H2 (F: flat, not using IPC; H: hierarchical). F1: all v_c shared (1); H1: v_c shared across each level of the tree (3). F2, H2: v_c shared across each subtree rooted at root's children (A: 15, B: 34, C: 17, D: 7, E: 7, F: 17, G: 12, H: 5). The numbers in parentheses are the total number of hyperparameters. Recall that there are three parameters determining the running time (see Section 2, Section 4). For hyperparameter learning: $k_1 = 8, k_2 = 4, k_3 = 15$ (F1, F2); $k_1 = 10, k_2 = 4, k_3 = 25$ (H1, H2).[12] For the final fitting (after hyperpars have been learned): $k_1 = 25, k_2 = 12$ (F1, F2); $k_1 = 30, k_2 = 17$ (H1, H2). The optimization is started from $v_c = 5$ for all methods. We set $\sigma^2 = 0.01$ throughout. Results are given in Table 1.

The hierarchical model outperforms the flat one consistently, especially w.r.t. taxo-loss and parent accuracy. Also, minimizing expected loss is consistently better than using the standard rule for the latter, although the differences are not significant. H1 and H2 do not perform differently: choosing many different v_c in the linear kernel seems no advantage here (but see Section 6.2). The results are quite similar to the ones of Cai and Hofmann (2004), obtained with a support vector machine variant. However, for our method, the recommendation in Cai and Hofmann (2004) to use $v_c = 1$ (not further motivated there) leads to significantly worse results in all scores. The v_c chosen by our method are generally larger. Note that their code has not been made publicly available, so a direct comparison with "all other things equal" could not be done.

In Table 2, we present running times[13] for the final fitting and for a single fold during hyperparameter optimization (5 of these are required for Ψ, $\nabla_h \Psi$). In comparison, a final fitting time of 2200*s* on the D section is quoted in Cai and Hofmann (2004), using a SVM variant, while we require 119*s* (more than six times faster).[14] It is precisely this high efficiency of primary fitting, which allows us to use it as inner loop for automatic hyperparameter learning (Cai and Hofmann, 2004, do not adjust hyperparameters to the data). Possible reasons for the performance difference are given in Section 7.2.

11. For parent accuracy, let $p(j)$ be the node with maximal mass (under p_j) of its children which are leafs, then $y(x_j)$ must be a child of $p(j)$.
12. Except for section C, where $k_1 = 14, k_2 = 6, k_3 = 35$.
13. Processor time on 64bit 2.33GHz AMD machines.
14. Cai and Hofmann average over three training/test splits. The timing figure 2200*s* in their paper is for three splits (thanks to one of the reviewers to point this out).

	acc (%)				prec (%)				taxo			
	F1	H1	F2	H2	F1	H1	F2	H2	F1	H1	F2	H2
A	40.6	**41.9**	40.5	**41.9**	51.6	**53.4**	51.4	**53.4**	1.27	**1.19**	1.29	**1.19**
B	32.0	**32.9**	31.7	32.7	41.8	**43.8**	41.6	43.7	1.52	**1.44**	1.55	**1.44**
C	33.7	**34.7**	34.1	34.5	45.2	**46.6**	45.4	46.4	1.34	**1.26**	1.35	1.27
D	40.0	40.6	39.7	**40.8**	52.4	54.1	52.2	**54.3**	1.19	**1.11**	1.18	**1.11**
E	33.0	**34.2**	32.8	34.1	45.1	**47.1**	45.0	**47.1**	1.39	**1.31**	1.38	**1.31**
F	31.4	32.4	31.4	**32.5**	42.8	44.9	42.8	**45.0**	1.43	**1.34**	1.43	**1.34**
G	40.1	**40.7**	40.2	**40.7**	51.2	**52.5**	51.3	**52.5**	1.32	**1.26**	1.32	**1.26**
H	39.3	39.6	39.4	**39.7**	52.4	53.3	52.5	**53.4**	1.17	1.15	1.17	**1.14**

	taxo[0-1]				pacc (%)				pacc[0-1] (%)			
	F1	H1	F2	H2	F1	H1	F2	H2	F1	H1	F2	H2
A	1.28	1.19	1.29	**1.18**	58.9	**61.6**	58.2	61.5	57.2	61.3	56.9	**61.4**
B	1.54	**1.44**	1.56	**1.44**	53.6	56.4	52.7	**56.6**	51.9	**55.9**	51.4	**55.9**
C	1.33	**1.26**	1.32	**1.26**	58.9	**62.6**	58.5	62.0	58.6	**61.8**	58.9	61.6
D	1.20	**1.12**	1.22	**1.12**	64.6	67.0	64.4	**67.1**	63.5	**67.1**	62.6	67.0
E	1.43	**1.33**	1.44	1.34	56.0	59.1	56.2	**59.2**	54.0	**58.2**	53.5	57.9
F	1.43	**1.34**	1.44	**1.34**	56.8	59.7	56.8	**59.8**	54.9	58.7	54.6	**58.9**
G	1.32	**1.26**	1.32	**1.26**	58.0	**59.7**	57.6	59.6	56.8	**59.2**	56.6	58.9
H	1.19	1.16	1.19	**1.15**	61.6	**62.5**	61.8	**62.5**	59.9	61.6	60.0	**61.8**

Table 1: Results on patent text classification tasks A-H. Methods F1, F2 flat, H1, H2 hierarchical. taxo[0-1], pacc[0-1] for $\mathrm{argmax}_c\, p_{jc}$ standard prediction rule, rather than minimization of expected loss.

	Final NR (s)		CV Fold (s)			Final NR (s)		CV Fold (s)	
	F1	H1	F1	H1		F1	H1	F1	H1
A	2030	3873	573	598	E	131.5	203.4	32.2	49.6
B	3751	8657	873	1720	F	1202	2871	426	568
C	4237	7422	719	1326	G	1342	2947	232	579
D	56.3	118.5	9.32	20.2	H	971.7	1052	146	230

Table 2: Running times for tasks A-H. Method F1 flat, H1 hierarchical. Final NR: Final fitting with Newton-Raphson. CV Fold: Re-optimization of $\alpha_{[J]}$ and gradient accumulation for single fold J.

6.2 Flat Classification: Remote Sensing

We use the *satimage* remote sensing task from the *statlog* repository.[15] This task has been used in the extensive SVM multi-class study of Hsu and Lin (2002), where it is among the data sets on which the different methods show the most variance. It has $n = 4435$ training, $m = 2000$ test cases, and $C = 6$ classes. Covariates x have 36 attributes with values in $\{0,\ldots,255\}$. No preprocessing was done.

We use the isotropic *Gaussian* (RBF) covariance function

$$K^{(c)}(x,x') = v_c \exp\left(-\frac{w_c}{2}\|x-x'\|^2\right), \quad v_c, w_c > 0. \tag{7}$$

We compare the methods *mc-sep* (ours with separate kernels for each class; 12 hyperparameters), *mc-tied* (ours with a single shared kernel; 2 hyperparameters), *mc-semi* (ours with single kernel $M^{(1)}$, but different v_c; 7 hyperparameters), *1rest* (one-against-rest; 12 hyperparameters). For *1rest*, C binary classifiers are fitted on the tasks of separating class c from all others. They are combined afterwards by the rule $x_* \mapsto \mathrm{argmax}_c \hat{P}_c(+1|x_*)$, where $\hat{P}_c(+1|x_*)$ is the predictive probability estimate of the c-classifier.[16] Note that *1rest* is arguably the most efficient method, in that its binary classifiers can be fitted separately and in parallel. Even if run sequentially, *1rest* typically requires less memory by a factor of C than a joint multi-class method, although this is not true if the kernel matrices are dominating the memory requirements and they are shared between classes in a multi-class method (as in *mc-tied* and *mc-semi* here).

We use our 5-fold CV criterion Ψ for each method. Results here are averaged over ten randomly drawn 5-partitions of the training set (the same partitions are used for the different methods). All optimizations are started from $v_c = 10$, $w_c = (\sum_j \mathrm{Var}[x_j])^{-1} = 0.017$, $\mathrm{Var}[x_j]$ being the empirical variance of attribute j. We set $\sigma^2 = 16$ throughout. The parameters determining the running time (see Section 2, Section 4) are set to $k_1 = 13, k_2 = 25, k_3 = 40$ during hyperparameter learning, and $k_1 = 30, k_2 = 50$ for final fitting (these are very conservative settings). Error-reject curves are shown in Figure 3.

Test errors are 7.95%($\pm$0.15%) for *mc-sep*, 8.00%($\pm$0.10%) for *1rest*, 8.10%($\pm$0.13%) for *mc-semi*, and 8.35%($\pm$0.20%) for *mc-tied*. Therefore, using a single fixed kernel for all $K^{(c)}$ does significantly worse than allowing for an individual $K^{(c)}$ per class. The test error difference between *mc-sep* and *1rest* is not significant here, but the error-reject curve is significantly better for our method *mc-sep* than for one-against-rest, especially in the domain $\alpha \in [0.025, 0.25]$, arguably most important in practice (where the rejection of a small fraction of test cases may often be an option). This indicates that the predictive probability estimates from our method are better than from one-against-rest, at least w.r.t. their ranking property. The curves for *mc-semi*, *mc-tied* are closer to *1rest*, underlining that different kernels $K^{(c)}$ should be used for each class. The result for *mc-sep* is state-of-the-art. The best SVM technique tested in Hsu and Lin (2002) attained 7.65% (no error-reject curves were given there), and SVM one-against-rest attained 8.3% in this study. To put this into perspective, note that extensive hyperparameter selection by cross-validation is done in Hsu and Lin (2002), in what seems to be a quite user-intensive process, while our method is completely automatic.

15. Available at `http://www.niaad.liacc.up.pt/old/statlog/`.
16. Asymptotically, $\hat{P}_c(+1|x_*)$ converges to the true $P(y_* = c|x_*)$, and this combination rule is optimal. We use our method with $C = 2$ in order to implement the binary classifiers.

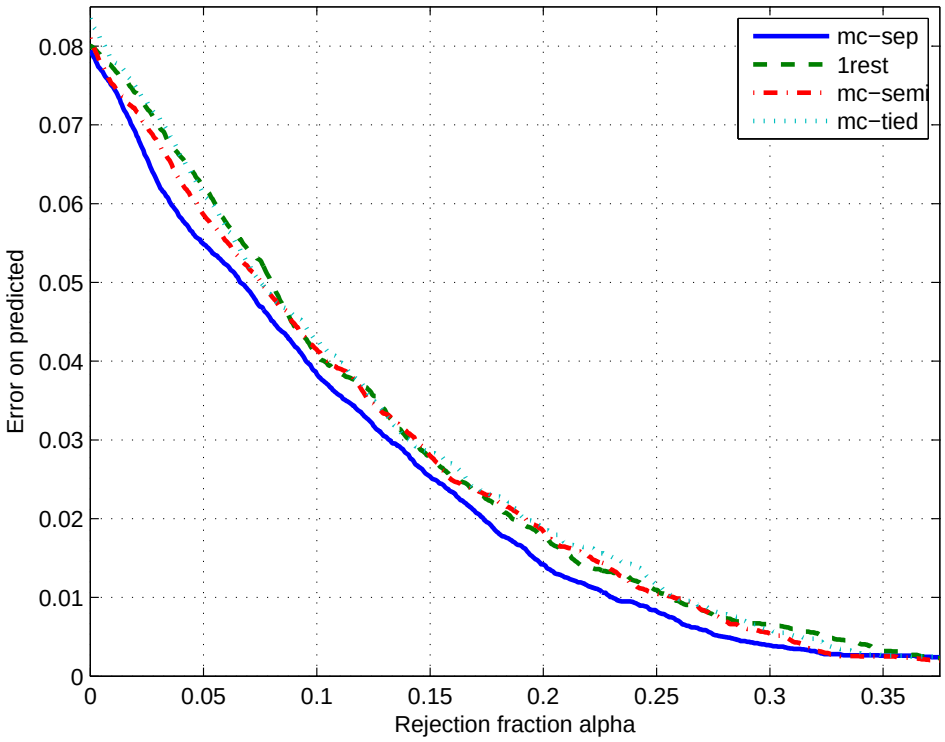


Figure 3: Error-reject curves (averaged over 10 runs) for different methods on the *satimage* task. Curve obtained by allowing the method to abstain from prediction on fraction α of test set, counting errors for predictions only. Depends on ranking of test points. Ranking score (over test points x_*): $\max_c \hat{P}(y_* = c|x_*)$ (*mc-sep*, *mc-semi*, *mc-tied*), $\max_c \hat{P}_c(+1|x_*)$ (*1rest*).

6.3 Flat Classification: Handwritten Digits

We use the *USPS* handwritten digits recognition task (LeCun et al., 1989). The covariates x are 16×16 gray-scale images with values in $\{16k+15\,|\,k = 0,\ldots,30\}$. The task has $n = 7291$ training, $m = 2007$ test cases, and $C = 10$ classes. No preprocessing was done.

We use Gaussian kernels (7) once more, different ones for each class. We do not optimize the 5-fold CV criterion Ψ using the full training set, but subsets of size $n' = 2000$. Our results are averaged over five runs with different randomly drawn training subsets for hyperparameter learning, while we use the full training set for final fitting. All optimizations are started from $v_c = 10$, $w_c = (\sum_j \mathrm{Var}[x_j])^{-1} = 0.0166$, and we set $\sigma^2 = 4$ throughout. The parameters determining the running time are $k_1 = 25$, $k_2 = 35$, $k_3 = 40$ during hyperparameter learning (on $n' = 2000$ points), and $k_1 = 45$, $k_2 = 80$ for final fitting (on $n = 7291$ points). The settings for hyperparameter learning are quite conservative, and the final fitting ones were sufficient for convergence on three of the five runs, whereas on two we had to add another $k_1 = 25$ iterations with $k_2 = 90$. An error-reject curve is shown in Figure 4.

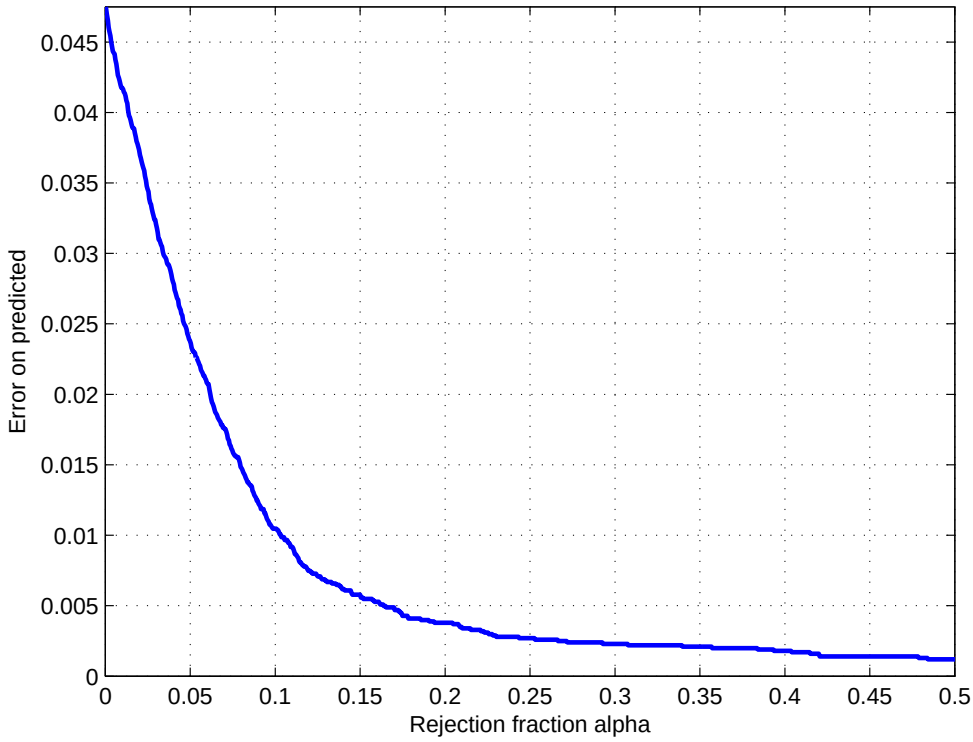


Figure 4: Error-reject curves (averaged over 5 runs) for different methods on the *USPS* task. Curve obtained by allowing the method to abstain from prediction on fraction α of test set, counting errors for predictions only. Depends on ranking of test points.

Test errors are $4.77\%(\pm 0.18\%)$. These results are state-of-the-art for kernel classification. Seeger (2003) reports 4.98% for the IVM (Sect. 4.8.4), where hyperparameters are learned automatically. Csató (2002) states 5.15% for his sparse online method with multiple sweeps over the data (Sect. 5.2). Results for the support vector machine are given in Schölkopf and Smola (2002), Table 18.1, method SV-254, where a combination heuristic based on kernel PCA was used to attain a test error of 4.4%. Crammer and Singer (2001) quote a test error of 4.38%, kernel parameters having been selected by 5-fold cross-validation. All these used the Gaussian kernel as well. The latter studies do not quote fluctuations w.r.t. choices such as the fold partition in CV, which is not negligible in our case here. The SVM-based methods do not attempt test set rankings or predictive probability estimation, and the corresponding studies do not show error-reject (or ROC) curves. Seeger (2003) gives an error-reject curve, which is very similar to ours here.

Note that the harder settings of k_1, k_2 for the final fitting are necessary due to the problem size, and are motivated in Section 4. There are 72910 parameters, and the hyperparameters found through optimizing Ψ spread by 3 orders of magnitude, so that the corresponding final fitting problems are computationally hard to solve without a good initialization of α (in the absence of such, we start with $\alpha = 0$). If we solve for Newton directions using too few LCG steps, the approximations often do not lead to much (or any) descent. Such "stalling" of NR line searches does happen now and then even

after $k_2 = 80$ LCG steps.[17] Lessons learned from these large scale experiments are commented on in Section 4. There are delicate dependencies between k_1, k_2 and the running time to convergence, which need to be explored in large scale settings, but this was not done thoroughly here.

7. Discussion

We have presented a general framework for learning kernel-based penalized likelihood classification methods from data. A central feature of the framework is its high computational efficiency, even though all classes are treated jointly. This is achieved by employing approximate Newton-Raphson optimization for the parameter fitting, which requires few large steps only for convergence. These steps are reduced to matrix-vector multiplication (MVM) primitives with kernel matrices. For general kernels, these MVM primitives can be reduced to large numerical linear algebra primitives, which can be computed very efficiently on modern computer architectures. This is very much in contrast to many chunking algorithms for kernel method fitting, which have been proposed in machine learning, and the advantages of our approach are detailed in Section 7.2. Dependencies between classes can be encoded *a priori* with minor additional efforts, as has been demonstrated for the case of hierarchical classification. Our method provides estimates of predictive probabilities which are asymptotically correct. Hyperparameters can be adjusted automatically, by optimizing a cross-validation log likelihood score in a gradient-based manner, and these computations are once more reduced to the same MVM primitives. This means that within our framework, all code optimization efforts can be concentrated on these essential primitives (see also Section 7.3), rather than having to tune a set of further heuristics.

7.1 Related Work

Our primary fitting optimization for flat multi-way classification appeared in Williams and Barber (1998), although some fairly essential features are novel here. They also did not consider large scale problems or class structures. Empirical Bayesian criteria such as the marginal likelihood are routinely used for hyperparameter learning in Gaussian process models (Williams and Barber, 1998; Williams and Rasmussen, 1996). However, in cases other than regression estimation with Gaussian noise, the marginal likelihood for a GP model cannot be computed analytically, and approximations differ strongly in terms of accuracy and computational complexity. For the multi-class model, Williams and Barber (1998) use an MAP approximation for fixed hyperparameters, just as we do, but their second-order approximation to the marginal likelihood is quite different from our criterion, conceptually as well as computationally (see below). Approximately solving large linear system by linear conjugate gradients (LCG) is standard in numerical mathematics, and has been used in machine learning as well (Gibbs, 1997; Williams and Barber, 1998; Keerthi and DeCoste, 2005).

The idea of optimizing approximations to a cross-validation score for hyperparameter learning is not novel (Craven and Wahba, 1979; Qi et al., 2004). Our approach is different to these, in that the CV score and gradient computations are reduced to elementary steps of the primary fitting method,

17. We cannot obtain a good initial α value from the final $\alpha_{[J_k]}$ of hyperparameter learning, because this is done on training subsets only. Moreover, in our implementation, the "stalling" (no improvement) of a NR step means that LCG is restarted from its last recent β, so that eventually an improvement in Φ is still obtained. Of course, the stalled NR iterations counts as such, and we do k_1 iterations in total.

so both can be done with the same code.[18] In contrast, scores like GCV (Craven and Wahba, 1979) or second order marginal likelihood (Williams and Barber, 1998) come in terms of the form $\mathrm{tr}\, H^{-1}$ or $\log|H|$ for the Hessian H of size nC. There are approximate reductions of computing these terms to solving linear systems (randomized trace estimator, Lanczos), but they rely on additional sampling of Gaussian noise, which introduces significant inaccuracies. In practice, optimizing such "noisy" criteria is quite difficult, whereas our criterion can be optimized using standard optimization code. Qi et al. (2004) propose an interesting approach of approximating leave-one-out CV using expectation propagation, see also Opper and Winther (2000). They use a sparse approximation for efficiency, but they deal with a single-process model only ($C = 1$), and it is not clear how to implement EP efficiently (scaling linearly in C) for the multi-class model. Interestingly, they observe that optimizing their approximate CV score is more robust to overfitting than the marginal likelihood. Finally, none of these papers propose (or achieve) a complete reduction to kernel MVM primitives only, nor do they deal with representing class structures or work on problems of the scale considered here.

Many different multi-class SVM techniques have been proposed, see Crammer and Singer (2001) and Hsu and Lin (2002) for references. These can be split into joint ("all-together") and decomposition methods. The latter reduce the multi-class problem to a set of binary ones ("one-against-rest" of Section 6.2 is a prominent example), with the advantage that good code is available for the binary case. The problem with these methods is that the binary discriminants are fitted separately without knowledge of each other, or of their role in the final multi-way classifier, so information from data is wasted. Also, their post-hoc combination into a multi-way discriminant is heuristic. Joint methods are like ours here, in that all classes are jointly represented. Fitting is a constrained convex problem, and often fairly sparse solutions (many zeros in α) are found. However, in difficult structured label tasks, the degree of sparsity is usually not high, and in these cases, commonly used chunking algorithms for multi-class SVM can be very inefficient (see Section 7.2). We should note that our approach here cannot be applied directly to multi-class SVMs, since they require the solution of a constrained convex problem, but the principles used here should hold there as well. Some novel suggestions here appear independently in Keerthi et al. (2007). SVM methods typically do not come with efficient automatic kernel parameter learning schemes, and they do not provide estimates of predictive probabilities which are asymptotically correct.

On the other hand, in a direct comparison our implementation would still be slower than the highly optimized multi-class SVM code of Crammer and Singer (2001), at least on standard non-structured tasks such as USPS (Section 6.3) or MNIST. Especially on the latter, sparsity in α is clearly present, and years of experience with the SVM problem led to very effective ways of exploiting it. In contrast, α in our approach is not sparse, and it is not our goal here to find a sparse approximation. Hyperparameters are selected "by hand" in their method, not via gradient-based optimization. For a small number of hyperparameters, this traditional approach is often faster than our optimization-based one here, and importantly, it can be fully parallelized. However, our approach is still workable in situations with many dependent hyperparameters (for example, Section 7.4.1), where CV by hand simply cannot be done.

Our ANOVA setup for hierarchical classification is proposed by Cai and Hofmann (2004), whose use it within a SVM "all-together" method. We compare our method against theirs in Sec-

18. A small drawback of our approach is that our CV score Ψ depends on a partitioning of the training set. In our experiments here, we chose this at random. Leave-one-out (LOO) CV does not depend on a partitioning, but it is not clear how to reduce LOO CV to solving a small number of linear systems.

tion 6.1, achieving quite similar results in an order of magnitude less time. They also do not address the problem of hyperparameter learning.

7.2 Global versus Decomposition Methods

In most kernel methods proposed so far in machine learning, the primary fitting to data (for fixed hyperparameters) translates to a convex minimization problem, where the penalisation terms correspond to quadratic expressions with kernel matrices. While kernel matrices may show a rapidly decaying eigenvalue spectrum, they certainly do couple the optimization variables strongly.[19] While a convex function can be optimized by any method which just guarantees descent in each step, there are huge differences in how fast the minimum is attained to a desired accuracy. In fact, in the absence of local minima, the speed of convergence becomes the most important characteristic of a method, besides robustness and ease of implementation.

In machine learning, the most dominant technique for large scale (structured label) kernel classification is what optimization researchers call *block coordinate descent methods* (BCD), see Bertsekas (1999, Sect. 2.7). The idea is to minimize the objective w.r.t. a few variables at a time, keeping all others fixed, and to iterate this process using some scheduling over the variables. Each step is convex again,[20] yet much smaller than the whole, and often the steps can be solved analytically. Ignoring the aspect of scheduling, such methods are simple to implement.

A complementary approach is to find search directions which lead to as fast a descent as possible, these directions typically involve all degrees of freedom of the optimization variables. If local first and second order information can be computed, the optimal search direction is Newton's, which has to be corrected if constraints are present (conditional gradient or gradient projection methods). If the Newton direction cannot be computed feasibly, approximations may be used. Such Newton-like methods are certainly vastly preferred in the optimization community, due to superior convergence rates, but also because features of modern computer architectures are used more efficiently, as is detailed below. In this paper, we advocate to follow this preference for kernel machine fitting in machine learning. We are encouraged not only by our own experiences, but can refer to the fact that (approximate) Newton methods are standard for fitting generalized linear models in statistics, and that such methods are also routinely used for Gaussian process models (Williams and Barber, 1998; Rasmussen and Williams, 2006), albeit typically on problems of smaller scale than treated here.

The dominance of BCD methods for kernel machine fitting, while somewhat surprising, can be attributed to early success stories with SVM training, culminating in the SMO algorithm (Platt, 1998), where only two variables are changed at a time. If an SVM is fitted to a task with low noise, the solution can be highly sparse, and if the active set of "support vectors" is detected early in the optimization process, methods like SMO can be very efficient. Importantly, SMO or other BCD methods are easily implemented. On the other hand, as SVMs are increasingly applied to hard structured label problems which usually do not have very sparse solutions, or whose active sets are hard to find, weaknesses of BCD methods become apparent.

Block coordinate descent methods are often referred to as using the "divide-and-conquer" principle, but this is not the case for kernel method fitting. BCD methods "are often useful in contexts where the cost function and the constraints have a partially decomposable structure with respect to

19. An almost low-rank kernel matrix translates into a coupling of a simple structure, but the dominant couplings are typically strong and not sparse.
20. If $f(x)$ is convex, so is $f(Ax)$ for any matrix A. The same is true for linear constraints.

the problem's optimization variables" (Bertsekas, 1999, p. 269). In kernel methods, such a decomposable structure is not present, because the penalisation terms couple all variables strongly via the kernel matrices. In such cases, chunking techniques *divide without conquering*, often running for very many steps, because improvements w.r.t. some block of variables tend to erase earlier improvements. This central problem of block coordinate descent methods is well known as "zig-zagging" in optimization. It occurs whenever variables not in the same block are significantly coupled, a situation which is to be expected for kernel machines in general. The situation is similar to a number of cases in machine learning and statistics. Iterative proportional fitting (Della Pietra et al., 1997) is a BCD method for learning the potentials of an undirected graphical model (Markov random field), which is all but superseded now by modern global direction methods such as limited memory Quasi-Newton, running orders of magnitude faster. Gibbs sampling is a basic Markov chain Monte Carlo technique for approximate Bayesian inference, which typically is very simple to implement, but is exceedingly slow in the presence of many coupled variables. Modern techniques such as Hybrid Monte Carlo, or Swendsen-Wang can be seen as "global direction" variants of "block coordinate" Gibbs sampling, and while they are harder to implement, they typically run orders of magnitude faster.

Another significant problem with BCD methods may come more as a surprise, namely because it concerns a characteristic which is often sold as advantage of these methods: they make each step as small as possible. Such small steps can often be dealt with analytically, or by using simple methods. This characteristic certainly makes BCD methods easy to implement. However, in light of modern computer architectures, the advice must be to make each step as *large as possible*, with the aim of requiring fewer steps. Modern systems use many internal parallelisms and hierarchies of caching, with the aim of processing vector operations many times faster than an equivalent loop over scalar operations, and large global steps do make use of these features. In contrast, a method which calls very many small steps in a non-linear ordering, runs contrary to these mechanisms. For example, data transfer between cache levels is done in blocks of significant sizes, and a method which accesses memory element-wise from all over the place, leads to inefficiencies up to cache thrashing, where the majority of cache accesses are misses (see Appendix D.3 for an example).

In well-designed global direction methods, the bottleneck operations (where almost all real-world running time is spent) are large vectorised mappings which access memory contiguously. Even better, these operations should lie in a standard class, for which highly optimized implementations are readily available. In our case here, the bottleneck operations are numerical linear algebra primitives from the *basic linear algebra subroutines* (BLAS), a standardized interface for high-performance dense linear algebra code. Very efficient implementations of the BLAS are available for all computer architectures.[21]

In this paper, we advocate to take a step back and to use global direction methods as approximation to Newton's method for kernel machine fitting. The prospect seems daunting, since there are many thousands of variables with complicated couplings, and the reader may be reminded of early disastrous trials of applying off-the-shelf QP packages to SVM fitting, or of ongoing efforts to formulate otherwise tractable machine learning problems as semidefinite programs and "solving" them using $O(n^7)$ SDP packages. This association is wrong. Our advice is to *approximate* the global Newton direction, making use of all structure in the model in order to gain efficiency, which is exactly the opposite of running a black box solver or implementing Newton's method straight

21. ATLAS, a self-tuning BLAS implementation, is available as free software, see `http://math-atlas.sourceforge.net/`.

out of a textbook. In the context of kernel machines, where couplings through large unstructured matrices are present, the large steps of approximating the Newton direction should be reduced to standard linear algebra primitives on dense or sparse matrices, operating on contiguous chunks of memory *as large as possible*, since highly optimized code for such primitives is readily available.

7.3 Matrix-Vector Multiplication

The computational load in our framework is determined by applications of the MVM primitives $v \mapsto K^{(c)}v$ and $v \mapsto (\partial K^{(c)}/\partial h_p)v$. A user only needs to provide those for a kernel of choice. The generic representation of Appendix D.1 applies to general covariance functions, but much more efficient alternatives may be used in special cases (see Appendix D.3).

If the cost for a direct evaluation of these primitives is prohibitive, several well-known approximations may be applied. Its has been suggested to use specialized data structures to approximate MVM with matrices from isotropic kernels (Yang et al., 2005; Shen et al., 2006) such as the Gaussian one (7). For such kernels, the derivative MVM can often be addressed using the same techniques. For the Gaussian kernel, we have $K = v\exp(wA)$, $A = (-(1/2)\|x_i - x_j\|^2)_{i,j}$, in which case $(\partial K/\partial \log v) = K$ and $(\partial K/\partial \log w) = wK \circ A$, where $\circ$ denotes component-wise product. Since the specialized data structures concentrate on approximating A, they apply to all required MVM variants. Our public code could fairly easily be extended, given specialized approximate kernel MVM code is available.

7.4 Extensions and Future Work

Some concrete extensions are mentioned just below. In general, we think that many structured label kernel methods proposed in the SVM context can be addressed in our framework as well. For example, the kernel conditional random field (CRF) (Lafferty et al., 2004) for label sequence learning can be treated by recognizing that MVM with the Hessian of the CRF log likelihood can be implemented efficiently using the method described in Pearlmutter (1994). We also plan to address hierarchical multi-label problems, which differ from hierarchical multi-class in that each instance can have multiple associated labels.

7.4.1 Modelling Dependencies between Classes

In the flat classification application of Section 2, we do not explicitly represent dependencies between classes. This is done in the hierarchical classification application of Section 3, but the dependency structure is fixed *a priori*. In this section, we motivate how dependencies between classes can be learned from data.

Let $B \in \mathbb{R}^{C,C}$ be a nonsingular coupling matrix, which will be a part of the model. In fact, B should be regarded as hyperparameters. In the flat model, we have $u_i = f_i + b$, which is now replaced by $u_i = Bf_i + b$, or

$$u = (B \otimes I)K\alpha + b \otimes 1.$$

This is the same modification which led to the hierarchical extension, only that the fixed coupling matrix Φ is replaced by the variable B. Therefore, the same modification of our method can be done, replacing K by $K^{(B)} = (B \otimes I)K(B^T \otimes I)$. Again, MVM with $K^{(B)}$ is of the same complexity as with K, because MVM with $(B \otimes I)$ can be done in $O(C^2 n)$. Note that B represents *conditional* depen-

dencies between the $u_c(\cdot)$, its role is comparable to the "mixing matrix" in independent components analysis.

We are also interested in learning B, whose elements are taken as hyperparameters. The corresponding gradient of Ψ is obtained in the same way as described in Section 5.2, only that $dK_{[B]}$ now further decomposes into parts for dB and for dK. Note that learning B by non-automatic cross-validation would not be possible, due to the large number of components.

Consider the case where we have many classes C, but not much data for most of them. We can postulate the assumptions that the behaviour of the C class functions $u_c(\cdot)$ is represented by $p \ll C$ underlying latent factors, which are then modelled as independent. This is achieved in our framework by having a non-square mixing matrix $B \in \mathbb{R}^{C,p}$ (the "factor loadings"). It is not hard to adapt our framework to this case, in which it is obviously necessary to learn B as hyperparameters from data. A related model in a Bayesian context was considered in Teh et al. (2005).

7.4.2 UNCERTAIN TARGETS IN HIERARCHICAL CLASSIFICATION

Recall the hierarchical classification setup of Section 3. Suppose that for some patterns x_i, the target is unknown, but we know that the path from its class to the root goes through an inner node p. Denote by L_p the set of leaf nodes of the subtree rooted at p, so that $L_p = \{p\}$ for a leaf node $p \in L$, and $L_0 = L$.

We can allow for such uncertain target information by using pseudo-targets $\tilde{y}_i \in \{1, \ldots, P\}$. If $\tilde{y}_i \notin L$, it is the lowest inner node we are certain about. The corresponding likelihood factor is

$$\sum_{c \in L_{\tilde{y}_i}} P(y_i = c | u_i).$$

Note that the log likelihood is not a concave function anymore, whenever $|L_{\tilde{y}_i}| > 1$, and in the presence of such factors, primary fitting is not a convex problem. However, an expectation-maximization (EM) (Dempster et al., 1977) approach can be used to deal with uncertain targets. Namely, in "E steps" we compute

$$q_{ic} \propto \mathrm{I}_{\{c \in L_{\tilde{y}_i}\}} P(y_i = c | u_i)$$

for the current $u_c(\cdot)$, where $q_i = (q_{ic})_c$ are distributions. "M steps" consists of Newton-Raphson iterations as before, but using $\sum_c q_{ic} \log P(y_i = c | u_i)$ as log likelihood factors. To this end, we just have to replace the vector $y \in \mathbb{R}^{nC}$ by q. Importantly, we only used the properties $1^T y_i = 1$, $y_i \geq 0$ above (but not that $y_{ic} \in \{0, 1\}$), which are true for q just as well.

7.4.3 LOW RANK APPROXIMATIONS

Our generic kernel matrix representation is described in Appendix D.1. If the data set size n is large, we may not be able to keep the correlation matrices $M^{(l)}$ in memory, and MVM with them becomes prohibitively expensive. We can use standard low rank matrix approximations to deal with this problem (see also Section 7.3). Namely, suppose that $M^{(l)}$ is approximated by $P^{(l)} L^{(l)} L^{(l)T} P^{(l)T}$, where $P^{(l)}$ is a permutation matrix, and $L^{(l)} \in \mathbb{R}^{n,d_l}$ for $d_l \ll n$. Denote $I_l \subset \{1, \ldots, n\}$ the active set of size d_l. The approximation may be obtained by an incomplete Cholesky factorization[22] (ICF), which has the special property that only a small set of d_l columns of $M^{(l)}$ (along with its diagonal)

22. Matlab code for ICF (in the form required here) can be downloaded from `http://www.kyb.tuebingen.mpg.de/bs/people/seeger/software.html`.

ever have to be evaluated (Bach and Jordan, 2002). In this case, $L^{(l)}_{1\ldots d_l,\cdot}$ is the lower triangular Cholesky factor of $M^{(l)}_{I_l} \in \mathbb{R}^{d_l,d_l}$, so that $P^{(l)T} M^{(l)}_{\cdot,I_l} = L^{(l)} {L^{(l)}_{1\ldots d_l,\cdot}}^T$. Note that in the ICF case, we have that

$$\mathrm{diag}\left(P^{(l)T} M^{(l)} P^{(l)} - L^{(l)} L^{(l)T}\right) \geq 0$$

point-wise, because the elements are simply the squared pivots for a potential continuation of the factorization (which has been stopped after d_l steps). Therefore, we can correct the approximation by replacing the diagonal of $L^{(l)}L^{(l)T}$ by the true one, ending up with the approximation

$$M^{(l)} \approx \tilde{M}^{(l)} := (\mathrm{diag}^2 M^{(l)}) + P^{(l)}\left(L^{(l)}L^{(l)T} - (\mathrm{diag}^2 L^{(l)}L^{(l)T})\right)P^{(l)T}.$$

Snelson and Ghahramani (2006) motivate this diagonal correction in another context. It is clear that MVM with $\tilde{M}^{(l)}$ can be done in $O(n\,d_l)$.

If $*$ indexes test points different from the training points, then the test-training correlation matrix is

$$M^{(l)}_{*,\cdot} = M^{(l)}_{*,I}(L^{(l)}_{1\ldots d_l,\cdot})^{-T} L^{(l)T} P^{(l)T}.$$

We can also learn parameters of the $M^{(l)}$ functions in this low rank setup by gradient-based optimization, assuming that the choice of I_l does not depend on these kernel parameters, but this is not discussed here.

Acknowledgments

We would like to thank Olivier Chapelle for many useful discussions on approximations of cross-validation optimization, and Peter Gehler for help with preprocessing the WIPO-alpha data, furthermore L. Cai and T. Hofmann for providing us with counts and dictionary data. Dilan Görur gave helpful comments on the NIPS paper related to this work. Supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778.

Appendix A. Notation

In this section, we describe the notation used in this paper. We denote vectors and matrices by boldface lower-case and upper-case letters, scalars and scalar functions are set normally. Subscripts select parts of objects, they can be single indexes or index sets. For example, $a = (a_i)_i$ is a vector with components a_i, $A = (a_{i,j})_{i,j}$ a matrix with entries $a_{i,j}$. We also write $a = (a_i)$, $A = (a_{i,j})$ if the indexes are clear from context. $A_{\cdot,i}$ is the i-th column of A ("$\cdot$" is short for the full index set). $\otimes$ denotes the Kronecker product, $A \otimes B = (a_{i,j}B)_{i,j}$, 1 (0) the vector of all ones (vector/matrix of all zeros), I the identity matrix, and $\delta_j = (\mathrm{I}_{\{i=j\}})_i$ (columns of I). For a matrix A, $\mathrm{diag}A = (a_{i,i})_i$ extracts the diagonal. For a vector v, $\mathrm{diag}v$ is the corresponding diagonal matrix. We also use this for matrix-valued vectors, an example is the diagonal kernel matrix $K = \mathrm{diag}(K^{(c)})_c$ in flat classification.

Many vectors and matrices are indexed by data points (i) and classes (c) at the same time, for example $u = (u_{ic}) \in \mathbb{R}^{nC}$. We use double indexes ic for these, which are flattened as $i + (c-1)n$, so the component ordering[23] is $u = (u_{11}, u_{21}, \ldots, u_{n1}, u_{12}, \ldots)$. In this context, selection index sets

23. In `Matlab`, `reshape(u,n,C)` would give a matrix in $\mathbb{R}^{n,C}$.

I are applied to the i (data point) index only: $u_I = (u_{ic})_{i\in I,c} \in \mathbb{R}^{|I|C}$. Kronecker product notation works nicely with this double index convention. If $A \otimes B$ is applied to u, A has C, B n columns. We frequently use $(1^T \otimes I)u = \sum_c u^{(c)}$, where $u^{(c)} = (u_{ic})_i \in \mathbb{R}^n$, or $(1 \otimes I)v$ for $v \in \mathbb{R}^n$, which stacks v on top of each other C times. The matrix $P_{cls} = (1 \otimes I)(1^T \otimes I)$ (introduced in Section 5.1) combines these operations:

$$P_{cls}x = \left.\begin{pmatrix} \bar{x} \\ \bar{x} \\ \vdots \end{pmatrix}\right\} C, \quad \bar{x} = \sum_c x^{(c)},$$

and P_{data} does the same, but operating on the i rather than the c index.

All major notational definitions are listed in Table 3 for reference. For kernel matrices (for example, $K^{(c)}$), we do not list the kernel functions (here: $K^{(c)}$), and for evaluation vectors (for example, u), we do not list the underlying functions (here: $u^{(c)}(\cdot)$).

n	Number data points	2	LCG	Linear Conjugate Gradients	2
C	Number classes	2	P	Number nodes (hierarchy)	3
y	Targets (zero-one)	2	L	Leaf nodes (hierarchy)	3
x_i	Input points	2	$\breve{u}$	Latent output (before mixing)	3
u	Latent output (after mixing)	2	Φ	Hierarchy mixing matrix	3
b	Intercepts	2	$\breve{K}$	Kernel matrix (before mixing)	(2)
σ^2	Penalizing constant for b	2	I_k, J_k	Partitions for CV criterion	4
Φ	Criterion for primary fitting	2	Ψ	CV criterion	4
α	Dual variables	2	q	Number of folds	4
K	Kernel matrix (after mixing)	2	h	Hyperparameters	4
$K^{(c)}$	Kernel matrix block	2	k_3	Complexity parameter	4
$\tilde{K}$	Kernel matrix (b eliminated)	2	g, W	Gradient, Hessian Φ_{lh}	5.1
l	Logsumexp vector	(1)	P_{cls}	Sum-distribute matrix	5.1
k_1, k_2	Complexity parameters	2	P_{data}	Sum-distribute matrix	5.1
NR	Newton-Raphson	2	E, F	Accumulation matrices	5.2

Table 3: Reference for notational definitions. k: Section of definition; (k): Equation of definition.

Appendix B. Details for Primary Fitting Algorithm

In this section, we discuss further details of the primary fitting algorithm of Section 2, in addition to Section 5.1.

We need to counter the problem that roundoff errors may lead to numerical instabilities. The criterion we minimize is strictly convex, even if the kernel matrix K is singular (or nearly so). However, problems could arise from components in π becoming very small. Recall that $\log \pi_{ic} = u_{ic} - l_i$. We make use of a threshold $\kappa < 0$ and define

$$I = \{(i,c) \,|\, \log \pi_{ic} < \kappa,\, y_{ic} > 0\}, \quad I_0 = \{(i,c) \,|\, \log \pi_{ic} < \kappa,\, y_{ic} = 0\}.$$

The indices in I can be problematic due to the corresponding component $\tilde{g}_{ic} \approx y_{ic}/\pi_{ic}^{1/2}$ becoming large. Note that this happens only if (x_i, y_i) is a strong outlier w.r.t. the current predictor. Now, from

the system (5) we see that $D^{1/2}\beta = DA^T(u - \tilde{K}\alpha') - g$. Therefore, if $(i,c) \in I$, then $(D^{1/2}\beta)_{ic} \approx -g_{ic} \approx y_{ic}$. The idea is to solve the reduced system on the components in $\setminus I$ for $(D^{1/2}\beta)_{\setminus I}$ and to plug in $(D^{1/2}\beta)_I = y_I$. Finally, within $\setminus I$, the components in I_0 may be problematic when computing the starting value $\beta = D^{-1/2}\alpha$ for the CG run. However, in this case $\tilde{g}_{ic} \approx 0$, leading to $\beta_{ic} \approx 0$ from (5). The corresponding components in the starting value β can therefore be set to zero.

Next, the LCG algorithm for solving systems of the form (5) needs to be preconditioned. Suppose we want to solve $Ax = b$. If we have an approximation $\tilde{A}$ to A so that $v \mapsto \tilde{A}^{-1}v$ can be computed efficiently (essentially in linear time in the size of v), the preconditioned CG algorithm solves the system $\tilde{A}^{-1}Ax = \tilde{A}^{-1}b$ instead. The idea is that $\tilde{A}^{-1}A$ typically has a lower condition number than A, and LCG converges faster and less erratically. Our implementation does preconditioning with the diagonal of the system matrix $I + V^T\tilde{K}V$. Note that $V\delta_{ic} = \pi_{ic}^{1/2}(\delta_c - \pi_i) \otimes \delta_i$, so that

$$\left(I + V^T\tilde{K}V\right)_{ic,ic} = 1 + \pi_{ic}\left((1 - 2\pi_{ic})(K_i^{(c)} + \sigma^2) + \sum_{c'} \pi_i^{(c')2}(K_i^{(c')} + \sigma^2)\right).$$

Therefore, the diagonal can be computed based on the $\mathrm{diag}\,K^{(c)}$ vectors. If the joint kernel matrix K is not block-diagonal (as in hierarchical classification, see Section 3), $\mathrm{diag}\,K$ is not sufficient for computing the system matrix diagonal. Let $v \in \mathbb{R}^{nC}$ be defined via $v_i = K_i\pi_i$, where $K_i = (I \otimes \delta_i^T)K(I \otimes \delta_i) \in \mathbb{R}^{C,C}$. Then, the system matrix diagonal has elements

$$1 + \pi_{ic}\left(K_{ic} + \sigma^2 - 2w_{ic} + \pi_i^T w_i\right), \quad w = v + \sigma^2\pi.$$

Appendix C. Solving Systems Exactly

In this section, we show how to implement our flat multi-class scheme using exact rather than approximate solutions of linear systems, yet still scaling linearly in C (at present, we do not know how to implement hierarchical classification exactly with such scaling).

For a Newton step, we need to solve $(I + W\tilde{K})\alpha' = r$ with $W = D - DP_{cls}D$. This can be written as

$$\left(A - UV^T\right)D^{-1/2}\alpha' = D^{-1/2}r, \quad A = I + D^{1/2}\tilde{K}D^{1/2},$$
$$U = D^{1/2}(1 \otimes I), \quad V = (A - I)U.$$

We now use the Sherman-Morrison-Woodbury formula together with the fact that $U^TU = \sum_c D^{(c)} = I$ to obtain

$$\alpha' = D^{1/2}\left(A^{-1} + A^{-1}U(U^TA^{-1}U)^{-1}U^T(I - A^{-1})\right)D^{-1/2}r.$$

We used that $V^TA^{-1} = U^T(I - A^{-1})$. Note that A^{-1} is block-diagonal, and that

$$U^TA^{-1}U = \sum_c D^{(c)1/2}A^{(c)-1}D^{(c)1/2}.$$

We maintain Cholesky factors of all $A^{(c)}$, as well as the Cholesky decomposition $U^TA^{-1}U = RR^T$ (where $A^{(c)-1}$ are obtained from the Cholesky factors).

For hyperparameter learning, we consider the partitions (I,J) sequentially. Since the $\alpha_{[J]}$ are different across folds, we cannot obtain the $A_{[J]}, H_{[J]}$ as parts of underlying common matrices. Recall Section 5.2. $\alpha_{[J]} + g_{[J]} = 0$ gives $H_{[J]}(d\alpha_{[J]}) = -W_{[J]}(dK_J)\alpha_{[J]}$. With

$$f = I_{\cdot,I}(\pi_{[I]} - y_I) - I_{\cdot,J}W_{[J]}H_{[J]}^{-T}\tilde{K}_{J,I}(\pi_{[I]} - y_I),$$

we have that $d\Psi_I = (I_{\cdot,J}\alpha_{[J]})^T (dK) f$. Again, these vectors are accumulated in matrices E, F. Solving a system with $H_{[J]}^T$ is an obvious variant of the procedure discussed above.

Appendix D. Further Details of the Implementation

Our implementation is designed to be as efficient as possible, while still being general and easy to extend to novel situations. This is achieved mainly by breaking down the problems to calling sequences of MVM primitives. These are then reduced to large numerical linear algebra primitives, where matrices are organized contiguously in memory, in order to exploit modern caching architectures (see Section 7.2).

D.1 A Generic Kernel Matrix Representation

A kernel matrix representation is some data structure which allows to compute kernel matrix MVMs $v \mapsto K^{(c)}v$ efficiently, being the principal primitives of our primary fitting method. Further requirements arise if additional features of our framework are used. For example, if hyperparameters are to be learned as well, derivative MVMs $v \mapsto (\partial K^{(c)}/\partial h_p)v$ are required as well, and "covariance shuffling" should be possible (see Section 5.2).

An efficient representation depends strongly on the covariance function used, and also on whether kernel matrix MVMs are approximated rather than computed exactly. For example, for linear kernels a special representation is used (see Appendix D.3). In this section, we describe a generic representation, which is part of our implementation.

The generic representation can be used with any covariance function, in that no special structure is assumed. It requires kernel matrices to be stored explicitly, which may not be possible for very large n. In general, we allow for different covariance functions $K^{(c)}$ for each class c, although sharing of kernels is supported, in that $K^{(c)}(\cdot,\cdot) = v_c M^{(l_c)}(\cdot,\cdot)$ and $v_c > 0$. Here, $l_c = l_{c'}$ is allowed for $c \neq c'$. The matrices $M^{(l)}$ are stored explicitly. Note that the flexibility of using different variance parameters v_c with the same $M^{(l)}$ does come at no extra cost, except for the fact that these have to be adjusted individually.

Since the $M^{(l)}$ are symmetric, two can be stored each in a $n \times n$ block, say the odd-numbered ones in the lower triangles. Here, the $\mathrm{diag}\, M^{(l)}$ are stored separately, and whenever a specific $M^{(l)}$ is required explicitly, the diagonal is copied into the block. It is important to note that the BLAS (see Section 7.2) directly supports symmetric matrices which are stored in the lower or upper triangle of a rectangular block.

The reader may wonder whether space could be saved by storing intermediates of the $M^{(l)}$ instead. For example, if the $M^{(l)}$ are isotropic kernels of the form $f^{(l)}(\|x - x'\|)$, we could store the inner product matrix $(x_i^T x_j)_{i,j}$ only. In practice, this turns out to be significantly slower (by a factor), the reason being that the optimized BLAS primitives are many times more efficient than applying a non-linear function $f^{(l)}$ point-wise to a matrix, even if the matrix is stored contiguously. For the same reason, computing MVMs on the fly without storing matrices is even more costly.

D.2 Shuffling the Kernel Matrix Representation

Covariance matrix shuffling has been motivated in Section 5.2. It is required during hyperparameter optimization, because the MVM primitives for sub-matrices K_{J_k} have to be driven by a single representation of the complete K (note that each K_{J_k} is of size $n(q-1)/q$, thus almost as large as K).

A simple approach would be to use sub-indexed matrix-vector multiplication code, but this is very inefficient (usually more than one order of magnitude slower than the flat BLAS functions).

Instead, when dealing with fold k, we shuffle the representation so that K_{J_k} moves to the upper left corner of the matrix. How this is done, depends on the representation. In this context, it is important to note that the underlying BLAS explicitly allows working on sub-matrices within upper left corners of larger frames, with virtually no loss in efficiency.[24] In the generic representation of Appendix D.1, we simply permute the kernel matrices $K^{(c)}$ using the index (J_k, I_k). A corresponding de-shuffling operation has to restore the old representation for K.

D.3 The Linear Kernel

Our application described in Section 6.1 uses the linear kernel $K^{(c)}(x,x') = v_c x^T x'$, where x is very high-dimensional (word counts over a dictionary), but also extremely sparse (by far the most entries are zero). The linear kernel fits the setup of Appendix D.1 with a single $M^{(1)} = XX^T$, where $X \in \mathbb{R}^{n,d}$ is the design matrix. X is very sparse, and in our implementation is represented using a standard sparse matrix format.

An MVM is done as $v \mapsto v_c(XX^T v)$, where X is sparse. More generally, we do $S \mapsto XX^T S$ with large matrices S. Kernel matrix shuffling (Appendix D.2) is implemented by simply reordering the non-zero positions for X. In this context, it is interesting to remark a finding which underlines the arguments in Section 7.2. The sparse matrix format is such that $XX^T S$ is reduced to so-called *daxpy* operations ($a = a + \alpha b$) on the *rows* of S. By Fortran (and BLAS) convention, S is stored in column-order, so that rows can only be accessed directly by using a striding value > 1 (the distance between consecutive vector elements in memory). We added a simple trick (called *dimension flipping*) to the implementation, which in essence switches our default ordering of Cn vectors $v = (v_{11}, v_{21}, v_{31}, \ldots)^T$ to $(v_{11}, v_{12}, v_{13}, \ldots)^T$ before major kernel MVM computations are done. This simple modification led to a direct five-times speedup, which underlines the importance of contiguous memory access in the bottleneck computations of a method (which allows optimal usage of cache hierarchies).

References

F. Bach and M. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

P. Bartlett and A. Tewari. Sparseness vs estimating conditional probabilities: Some asymptotic results. In *Conference on Computational Learning Theory 17*, pages 564–578. Springer, 2004.

D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2002.

L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *CIKM 13*, pages 78–87, 2004.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

24. Matrices in BLAS are stored column-wise, each column has to be contiguous in memory, but the striding value (offset in memory required to jump to the next column) can be larger than the number of rows.

P. Craven and G. Wahba. Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, 31:377–403, 1979.

L. Csató. *Gaussian Processes — Iterative Sparse Approximations*. PhD thesis, Aston University, Birmingham, UK, March 2002.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Roy. Stat. Soc. B*, 39:1–38, 1977.

M. Gibbs. *Bayesian Gaussian Processes for Regression and Classification*. PhD thesis, University of Cambridge, 1997.

P.J. Green and B. Silverman. *Nonparametric Regression and Generalized Linear Models*. Monographs on Statistics and Probability. Chapman & Hall, 1994.

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.

S. Keerthi and D. DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

S. Keerthi, V. Sindhwani, and O. Chapelle. An efficient method for gradient-based adaptation of hyperparameters in SVM models. In B. Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

J. Lafferty, Y. Liu, and X. Zhu. Kernel conditional random fields: Representation, clique selection, and semi-supervised learning. Technical Report CMU-CS-04-115, Carnegie Mellon University, 2004.

Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

P. McCullach and J.A. Nelder. *Generalized Linear Models*. Number 37 in Monographs on Statistics and Applied Probability. Chapman & Hall, 1st edition, 1983.

M. Opper and O. Winther. Gaussian processes for classification: Mean field algorithms. *Neural Computation*, 12(11):2655–2684, 2000.

B. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.

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 185–208. MIT Press, 1998.

Y. Qi, T. Minka, R. Picard, and Z. Ghahramani. Predictive automatic relevance determination by expectation propagation. In C. Brodley, editor, *International Conference on Machine Learning 21*. Morgan Kaufmann, 2004.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Y. Saad. *Iterative Methods for Sparse Linear Systems*. International Thomson Publishing, 1st edition, 1996.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 1st edition, 2002.

M. Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, July 2003. See `www.kyb.tuebingen.mpg.de/bs/people/seeger`.

M. Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14 (2):69–106, 2004.

M. Seeger. Cross-validation optimization for large scale hierarchical classification kernel methods. In B. Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*, pages 1233–1240. MIT Press, 2007.

B. Shahbaba and R. Neal. Improving classification when a class hierarchy is available using a hierarchy-based prior. *Bayesian Analysis*, 2(1):221–238, 2007.

Y. Shen, A. Ng, and M. Seeger. Fast Gaussian process regression using KD-trees. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*. MIT Press, 2006.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*. MIT Press, 2006.

Y.-W. Teh, M. Seeger, and M. I. Jordan. Semiparametric latent factor models. In Z. Ghahramani and R. Cowell, editors, *Workshop on Artificial Intelligence and Statistics 10*, 2005.

G. Wahba. *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series. Society for Industrial and Applied Mathematics, 1990.

C. Williams and C. Rasmussen. Gaussian processes for regression. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*. MIT Press, 1996.

C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

C. Yang, R. Duraiswami, and L. Davis. Efficient kernel machines using the improved fast Gauss transform. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1561–1568. MIT Press, 2005.

Journal of Machine Learning Research 9 (2008) 1179-1225 Submitted 7/07; Revised 3/08; Published 06/08

Consistency of the Group Lasso and Multiple Kernel Learning

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

Editor: Nicolas Vayatis

Abstract

We consider the least-square regression problem with regularization by a block ℓ_1-norm, that is, a sum of Euclidean norms over spaces of dimensions larger than one. This problem, referred to as the group Lasso, extends the usual regularization by the ℓ_1-norm where all spaces have dimension one, where it is commonly referred to as the Lasso. In this paper, we study the asymptotic group selection consistency of the group Lasso. We derive necessary and sufficient conditions for the consistency of group Lasso under practical assumptions, such as model misspecification. When the linear predictors and Euclidean norms are replaced by functions and reproducing kernel Hilbert norms, the problem is usually referred to as multiple kernel learning and is commonly used for learning from heterogeneous data sources and for non linear variable selection. Using tools from functional analysis, and in particular covariance operators, we extend the consistency results to this infinite dimensional case and also propose an adaptive scheme to obtain a consistent model estimate, even when the necessary condition required for the non adaptive scheme is not satisfied.

Keywords: sparsity, regularization, consistency, convex optimization, covariance operators

1. Introduction

Regularization has emerged as a dominant theme in machine learning and statistics. It provides an intuitive and principled tool for learning from high-dimensional data. Regularization by squared Euclidean norms or squared Hilbertian norms has been thoroughly studied in various settings, from approximation theory to statistics, leading to efficient practical algorithms based on linear algebra and very general theoretical consistency results (Tikhonov and Arsenin, 1997; Wahba, 1990; Hastie et al., 2001; Steinwart, 2001; Cucker and Smale, 2002).

In recent years, regularization by non Hilbertian norms has generated considerable interest in linear supervised learning, where the goal is to predict a response as a linear function of covariates; in particular, regularization by the ℓ_1-norm (equal to the sum of absolute values), a method commonly referred to as the *Lasso* (Tibshirani, 1996; Osborne et al., 2000), allows to perform variable selection. However, regularization by non Hilbertian norms cannot be solved empirically by simple linear algebra and instead leads to general convex optimization problems and much of the early effort has been dedicated to algorithms to solve the optimization problem efficiently. In particular, the *Lars* algorithm of Efron et al. (2004) allows to find the entire regularization path (i.e., the set of solutions for all values of the regularization parameters) at the cost of a single matrix inversion.

As the consequence of the optimality conditions, regularization by the ℓ_1-norm leads to *sparse* solutions, that is, loading vectors with many zeros. Recent works (Zhao and Yu, 2006; Yuan and

Lin, 2007; Zou, 2006; Wainwright, 2006) have looked precisely at the model consistency of the Lasso, that is, if we know that the data were generated from a sparse loading vector, does the Lasso actually recover it when the number of observed data points grows? In the case of a fixed number of covariates, the Lasso does recover the sparsity pattern if and only if a certain simple condition on the generating covariance matrices is verified (Yuan and Lin, 2007). In particular, in low correlation settings, the Lasso is indeed consistent. However, in presence of strong correlations between relevant variables and irrelevant variables, the Lasso cannot be consistent, shedding light on potential problems of such procedures for variable selection. Adaptive versions where data-dependent weights are added to the ℓ_1-norm then allow to keep the consistency in all situations (Zou, 2006).

A related Lasso-type procedure is the *group Lasso*, where the covariates are assumed to be clustered in groups, and instead of summing the absolute values of each individual loading, the sum of Euclidean norms of the loadings in each group is used. Intuitively, this should drive all the weights in one group to zero *together*, and thus lead to group selection (Yuan and Lin, 2006). In Section 2, we extend the consistency results of the Lasso to the group Lasso, showing that similar correlation conditions are necessary and sufficient conditions for consistency. Note that we only obtain results in terms of group consistency, with no additional information regarding variable consistency inside each group. Also, when the groups have size one, then we get back similar conditions than for the Lasso. The passage from groups of size one to groups of larger sizes leads however to a slightly weaker result as we can not get a single necessary and sufficient condition (in Section 2.6, we show that the stronger result similar to the Lasso is not true as soon as one group has dimension larger than one). Also, in our proofs, we relax the assumptions usually made for such consistency results, that is, that the model is completely well-specified (conditional expectation of the response which is linear in the covariates and constant conditional variance). In the context of *misspecification*, which is a common situation when applying methods such as the ones presented in this paper, we simply prove convergence to the best linear predictor (which is assumed to be sparse), both in terms of loading vectors and sparsity patterns.

The group Lasso essentially replaces groups of size one by groups of size larger than one. It is natural in this context to allow the size of each group to grow unbounded, that is, to replace the sum of Euclidean norms by a sum of appropriate Hilbertian norms. When the Hilbert spaces are reproducing kernel Hilbert spaces (RKHS), this procedure turns out to be equivalent to learn the best convex combination of a set of basis positive definite kernels, where each kernel corresponds to one Hilbertian norm used for regularization (Bach et al., 2004a). This framework, referred to as *multiple kernel learning* (Bach et al., 2004a), has applications in kernel selection, data fusion from heterogeneous data sources and non linear variable selection (Lanckriet et al., 2004a). In this latter case, multiple kernel learning can exactly be seen as variable selection in a *generalized additive model* (Hastie and Tibshirani, 1990). We extend the consistency results of the group Lasso to this nonparametric case, by using covariance operators and appropriate notions of functional analysis. These notions allow to carry out the analysis entirely in *"primal/input"* space, while the algorithm has to work in *"dual/feature"* space to avoid infinite dimensional optimization. Throughout the paper, we will always go back and forth between primal and dual formulations, primal formulation for analysis and dual formulation for algorithms.

The paper is organized as follows: in Section 2, we present the consistency results for the group Lasso, while in Section 3, we extend these to Hilbert spaces. Finally, we present the adaptive

schemes in Section 4 and illustrate our set of results with simulations on synthetic examples in Section 5.

2. Consistency of the Group Lasso

We consider the problem of predicting a response $Y \in \mathbb{R}$ from covariates $X \in \mathbb{R}^p$, where X has a block structure with m blocks, that is, $X = (X_1^\top, \ldots, X_m^\top)^\top$ with each $X_j \in \mathbb{R}^{p_j}$, $j = 1, \ldots, m$, and $\sum_{j=1}^m p_j = p$. Throughout this paper, unless otherwise specified, $\|a\|$ will denote the Euclidean norm of a vector a (for all possible dimensions of a, for example, p, n or p_j). The only assumptions that we make on the joint distribution P_{XY} of (X,Y) are the following:

(A1) X and Y have finite fourth order moments: $\mathbb{E}\|X\|^4 < \infty$ and $\mathbb{E}Y^4 < \infty$.

(A2) The joint covariance matrix $\Sigma_{XX} = \mathbb{E}XX^\top - (\mathbb{E}X)(\mathbb{E}X)^\top \in \mathbb{R}^{p \times p}$ is invertible.

(A3) We denote by $(\mathbf{w}, \mathbf{b}) \in \mathbb{R}^p \times \mathbb{R}$ any minimizer of $\mathbb{E}(Y - X^\top w - b)^2$. We assume that $\mathbb{E}((Y - \mathbf{w}^\top X - \mathbf{b})^2 | X)$ is almost surely greater than $\sigma_{\min}^2 > 0$. We denote by $\mathbf{J} = \{j, \mathbf{w}_j \neq 0\}$ the sparsity pattern of $\mathbf{w}$.[1]

The assumption **(A3)** does not state that $\mathbb{E}(Y|X)$ is an affine function of X and that the conditional variance is constant, as it is commonly done in most works dealing with consistency for linear supervised learning. We simply assume that given the best affine predictor of Y given X (defined by $\mathbf{w} \in \mathbb{R}^p$ and $\mathbf{b} \in \mathbb{R}$), there is still a strictly positive amount of variance in Y. If **(A2)** is satisfied, then the full loading vector $\mathbf{w}$ is uniquely defined and is equal to $\mathbf{w} = \Sigma_{XX}^{-1}\Sigma_{XY}$, where $\Sigma_{XY} = \mathbb{E}(XY) - (\mathbb{E}X)(\mathbb{E}Y) \in \mathbb{R}^p$. Note that throughout this paper, we do include a non regularized constant term b but since we use a square loss it will optimized out in closed form by centering the data. Thus all our consistency statements will be stated only for the loading vector w; corresponding results for b then immediately follow.

We often use the notation $\varepsilon = Y - \mathbf{w}^\top X - \mathbf{b}$. In terms of covariance matrices, our assumption **(A3)** leads to: $\Sigma_{\varepsilon\varepsilon|X} = \mathbb{E}(\varepsilon\varepsilon|X) \geqslant \sigma_{\min}^2$ and $\Sigma_{\varepsilon X} = 0$ (but ε might not in general be independent from X).

2.1 Applications of Grouped Variables

In this paper, we assume that the groupings of the univariate variables are known and fixed, that is, the group structure is given and we wish to achieve sparsity at the level of groups. This has numerous applications, for example, in speech and signal processing, where groups may represent different frequency bands (McAuley et al., 2005), or bioinformatics (Lanckriet et al., 2004a) and computer vision (Varma and Ray, 2007; Harchaoui and Bach, 2007) where each group may correspond to different data sources or data types. Note that those different data sources are sometimes referred to as *views* (see, e.g., Zhou and Burges, 2007).

Moreover, we always assume that the number m of groups is fixed and finite. Considering cases where m is allowed to grow with the number of observed data points, in the line of Meinshausen and Yu (2006), is outside the scope of this paper.

1. Note that throughout this paper, we use boldface fonts for population quantities.

2.2 Notations

Throughout this paper, we consider the block covariance matrix Σ_{XX} with m^2 blocks $\Sigma_{X_iX_j}$, $i,j = 1,\dots,m$. We refer to the submatrix composed of all blocks indexed by sets I, J as $\Sigma_{X_IX_J}$. Similarly, our loadings are vectors defined following block structure, $w = (w_1^\top,\dots,w_m^\top)^\top$ and we denote by w_I the elements indexed by I. Moreover we denote by 1_q the vector in $\mathbb{R}^q$ with constant components equal to one, and by I_q the identity matrix of size q.

2.3 Group Lasso

We consider *independent and identically distributed* (i.i.d.) data $(x_i,y_i) \in \mathbb{R}^p \times \mathbb{R}$, $i = 1,\dots,n$, sampled from P_{XY} and the data are given in the form of matrices $\bar{Y} \in \mathbb{R}^n$ and $\bar{X} \in \mathbb{R}^{n\times p}$ and we write $\bar{X} = (\bar{X}_1,\dots,\bar{X}_m)$ where each $\bar{X}_j \in \mathbb{R}^{n\times p_j}$ represents the data associated with group j (i.e., the i-th row of $\bar{X}_j$ is the j-th group variable for x_i, while $\bar{Y}_i = y_i$). Throughout this paper, we make the same i.i.d. assumption; dealing with non identically distributed or dependent data and extending our results in those situations are left for future research.

We use the square loss, that is, $\frac{1}{2n}\sum_{i=1}^n (y_i - w^\top x_i - b)^2 = \frac{1}{2n}\|\bar{Y} - \bar{X}w - b1_n\|^2$, and thus consider the following optimization problem:

$$\min_{w\in\mathbb{R}^p,\ b\in\mathbb{R}} \ \frac{1}{2n}\|\bar{Y} - \bar{X}w - b1_n\|^2 + \lambda_n \sum_{j=1}^m d_j \|w_j\|,$$

where $d = (d_1,\dots,d_m)^\top \in \mathbb{R}^m$ is a vector of strictly positive fixed weights. Note that considering weights in the block ℓ_1-norm is important in practice as those have an influence regarding the consistency of the estimator (see Section 4 for further details). Since b is not regularized, we can minimize in closed form with respect to b, by setting $b = \frac{1}{n}1_n^\top(\bar{Y} - \bar{X}w)$. This leads to the following reduced optimization problem in w:

$$\min_{w\in\mathbb{R}^p} \ \frac{1}{2}\hat{\Sigma}_{YY} - \hat{\Sigma}_{XY}^\top w + \frac{1}{2} w^\top \hat{\Sigma}_{XX} w + \lambda_n \sum_{j=1}^m d_j \|w_j\|, \tag{1}$$

where $\hat{\Sigma}_{YY} = \frac{1}{n}\bar{Y}^\top \Pi_n \bar{Y}$, $\hat{\Sigma}_{XY} = \frac{1}{n}\bar{X}^\top \Pi_n \bar{Y}$ and $\hat{\Sigma}_{XX} = \frac{1}{n}\bar{X}^\top \Pi_n \bar{X}$ are empirical covariance matrices (with the centering matrix Π_n defined as $\Pi_n = I_n - \frac{1}{n}1_n 1_n^\top$). We denote by $\hat{w}$ any minimizer of Eq. (1). We refer to $\hat{w}$ as the *group Lasso* estimate.[2] Note that with probability tending to one, if (**A2**) is satisfied (i.e., if Σ_{XX} is invertible), there is a unique minimum.

Problem (1) is a non-differentiable convex optimization problem, for which classical tools from convex optimization (Boyd and Vandenberghe, 2003) lead to the following optimality conditions (see proof by Yuan and Lin, 2006, and in Appendix A.1):

Proposition 1 *A vector $w \in \mathbb{R}^p$ with sparsity pattern $J = J(w) = \{j,\ w_j \neq 0\}$ is optimal for problem (1) if and only if*

$$\forall j \in J^c, \qquad \left\|\hat{\Sigma}_{X_jX} w - \hat{\Sigma}_{X_jY}\right\| \leqslant \lambda_n d_j, \tag{2}$$

$$\forall j \in J, \qquad \hat{\Sigma}_{X_jX} w - \hat{\Sigma}_{X_jY} = -w_j \frac{\lambda_n d_j}{\|w_j\|}. \tag{3}$$

2. We use the convention that all "hat" notations correspond to data-dependent and thus n-dependent quantities, so we do not need the explicit dependence on n.

2.4 Algorithms

Efficient *exact* algorithms exist for the regular Lasso, that is, for the case where all group dimensions p_j are equal to one. They are based on the piecewise linearity of the set of solutions as a function of the regularization parameter λ_n (Efron et al., 2004). For the group Lasso, however, the path is only piecewise differentiable, and following such a path is not as efficient as for the Lasso. Other algorithms have been designed to solve problem (1) for a single value of λ_n, in the original group Lasso setting (Yuan and Lin, 2006) and in the multiple kernel setting (Bach et al., 2004a,b; Sonnenburg et al., 2006; Rakotomamonjy et al., 2007). In this paper, we study path consistency of the group Lasso and of multiple kernel learning, and in simulations we use the publicly available code for the algorithm of Bach et al. (2004b), that computes an approximate but entire path, by following the piecewise smooth path with predictor-corrector methods.

2.5 Consistency Results

We consider the following two conditions:

$$\max_{i \in \mathbf{J}^c} \frac{1}{d_i} \left\| \Sigma_{X_i X_\mathbf{J}} \Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1} \operatorname{Diag}(d_j / \|\mathbf{w}_j\|) \mathbf{w}_\mathbf{J} \right\| < 1, \tag{4}$$

$$\max_{i \in \mathbf{J}^c} \frac{1}{d_i} \left\| \Sigma_{X_i X_\mathbf{J}} \Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1} \operatorname{Diag}(d_j / \|\mathbf{w}_j\|) \mathbf{w}_\mathbf{J} \right\| \leqslant 1, \tag{5}$$

where $\operatorname{Diag}(d_j/\|\mathbf{w}_j\|)$ denotes the block-diagonal matrix (with block sizes p_j) in which each diagonal block is equal to $\frac{d_j}{\|\mathbf{w}_j\|} I_{p_j}$ (with I_{p_j} the identity matrix of size p_j), and $\mathbf{w}_\mathbf{J}$ denotes the concatenation of the loading vectors indexed by $\mathbf{J}$. Note that the conditions involve the covariance between all active groups X_j, $j \in \mathbf{J}$ and all non active groups X_i, $i \in \mathbf{J}^c$.

These are conditions on both the input (through the joint covariance matrix Σ_{XX}) and on the weight vector $\mathbf{w}$. Note that, when all blocks have size 1, this corresponds to the conditions derived for the Lasso (Zhao and Yu, 2006; Yuan and Lin, 2007; Zou, 2006). Note also the difference between the *strong condition* (4) and the *weak condition* (5). For the Lasso, with our assumptions, Yuan and Lin (2007) has shown that the strong condition (4) is necessary and sufficient for path consistency of the Lasso; that is, the path of solutions consistently contains an estimate which is both consistent for the ℓ_2-norm (regular consistency) and the ℓ_0-norm (consistency of patterns), if and only if condition (4) is satisfied.

In the case of the group Lasso, even with a finite fixed number of groups, our results are not as strong, as we can only get the strict condition as sufficient and the weak condition as necessary. In Section 2.6, we show that this cannot be improved in general. More precisely the following theorem, proved in Appendix B.1, shows that if the condition (4) is satisfied, any regularization parameter that satisfies a certain decay conditions will lead to a consistent estimator; thus the strong condition (4) is sufficient for path consistency:

Theorem 2 *Assume (**A1-3**). If condition (4) is satisfied, then for any sequence λ_n such that $\lambda_n \to 0$ and $\lambda_n n^{1/2} \to +\infty$, the group Lasso estimate $\hat{w}$ defined in Eq. (1) converges in probability to $\mathbf{w}$ and the group sparsity pattern $J(\hat{w}) = \{j, \hat{w}_j \neq 0\}$ converges in probability to $\mathbf{J}$ (i.e., $\mathbb{P}(J(\hat{w}) = \mathbf{J}) \to 1$).*

The following theorem, proved in Appendix B.2, states that if there is a consistent solution on the path, then the weak condition (5) must be satisfied.

Theorem 3 *Assume (**A1-3**). If there exists a (possibly data-dependent) sequence λ_n such that $\hat{w}$ converges to $\mathbf{w}$ and $J(\hat{w})$ converges to $\mathbf{J}$ in probability, then condition (5) is satisfied.*

On the one hand, Theorem 2 states that under the "low correlation between variables in $\mathbf{J}$ and variables in $\mathbf{J}^c$" condition (4), the group Lasso is indeed consistent. On the other hand, the result (and the similar one for the Lasso) is rather disappointing regarding the applicability of the group Lasso as a practical group selection method, as Theorem 3 states that if the weak correlation condition (5) is not satisfied, we cannot have consistency.

Moreover, this is to be contrasted with a thresholding procedure of the joint least-square estimator, which is also consistent with no conditions (but the invertibility of Σ_{XX}), if the threshold is properly chosen (smaller than the smallest norm $\|\mathbf{w}_j\|$ for $j \in \mathbf{J}$ or with appropriate decay conditions). However, the Lasso and group Lasso do not have to set such a threshold; moreover, further analysis show that the Lasso has additional advantages over regular regularized least-square procedure (Meinshausen and Yu, 2006), and empirical evidence shows that in the finite sample case, they do perform better (Tibshirani, 1996), in particular in the case where the number m of groups is allowed to grow. In this paper we focus on the extension from uni-dimensional groups to multi-dimensional groups for finite number of groups m and leave the possibility of letting m grow with n for future research.

Finally, by looking carefully at condition (4) and (5), we can see that if we were to increase the weight d_j for $j \in \mathbf{J}^c$ and decrease the weights otherwise, we could always be consistent: this however requires the (potentially empirical) knowledge of $\mathbf{J}$ and this is exactly the idea behind the adaptive scheme that we present in Section 4. Before looking at these extensions, we discuss in the next Section, qualitative differences between our results and the corresponding ones for the Lasso.

2.6 Refinements of Consistency Conditions

Our current results state that the strict condition (4) is sufficient for joint consistency of the group Lasso, while the weak condition (5) is only necessary. When all groups have dimension one, then the strict condition turns out to be also necessary (Yuan and Lin, 2007).

The main technical reason for those differences is that in dimension one, the set of vectors of unit norm is finite (two possible values), and thus regular squared norm consistency leads to estimates of the signs of the loadings (i.e., their normalized versions $\hat{w}_j/\|\hat{w}_j\|$) which are ultimately constant. When groups have size larger than one, then $\hat{w}_j/\|\hat{w}_j\|$ will not be ultimately constant (just consistent) and this added dependence on data leads to the following refinement of Theorem 2 (see proof in Appendix B.3):

Theorem 4 *Assume (**A1-3**). Assume the weak condition (5) is satisfied and that for all $i \in \mathbf{J}^c$ such that $\frac{1}{d_i}\left\|\Sigma_{X_i X_\mathbf{J}}\Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1}\operatorname{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_\mathbf{J}\right\| = 1$, we have*

$$\Delta^\top \Sigma_{X_\mathbf{J} X_i}\Sigma_{X_i X_\mathbf{J}}\Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1}\operatorname{Diag}\left[d_j/\|\mathbf{w}_j\|\left(I_{p_j} - \frac{\mathbf{w}_j\mathbf{w}_j^\top}{\mathbf{w}_j^\top\mathbf{w}_j}\right)\right]\Delta > 0, \tag{6}$$

with $\Delta = -\Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1}\operatorname{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_\mathbf{J}$. Then for any sequence λ_n such that $\lambda_n \to 0$ and $\lambda_n n^{1/4} \to +\infty$, the group Lasso estimate $\hat{w}$ defined in Eq. (1) converges in probability to $\mathbf{w}$ and the group sparsity pattern $J(\hat{w}) = \{j, \hat{w}_j \neq 0\}$ converges in probability to $\mathbf{J}$.

This theorem is of lower practical significance than Theorem 2 and Theorem 3. It merely shows that the link between strict/weak conditions and sufficient/necessary conditions are in a sense tight (as soon as there exists $j \in \mathbf{J}$ such that $p_j > 1$, it is easy to exhibit examples where Eq. (6) is or is not satisfied). The previous theorem does not contradict the fact that condition (4) is necessary for path-consistency in the Lasso case: indeed, if w_j has dimension one (i.e., $p_j = 1$), then $I_{p_j} - \frac{\mathbf{w}_j \mathbf{w}_j^\top}{\mathbf{w}_j^\top \mathbf{w}_j}$ is always equal to zero, and thus Eq. (6) is never satisfied. Note that when condition (6) is an equality, we could still refine the condition by using higher orders in the asymptotic expansions presented in Appendix B.3.

We can also further refined the *necessary* condition results in Theorem 3: as stated in Theorem 3, the group Lasso estimator may be both consistent in terms of norm and sparsity patterns only if the condition (5) is satisfied. However, if we require only the consistent sparsity pattern estimation, then we may allow the convergence of the regularization parameter λ_n to a strictly positive limit λ_0. In this situation, we may consider the following population problem:

$$\min_{w \in \mathbb{R}^p} \frac{1}{2}(w - \mathbf{w})^\top \Sigma_{XX}(w - \mathbf{w}) + \lambda_0 \sum_{j=1}^{m} d_j \|w_j\|. \tag{7}$$

If there exists $\lambda_0 > 0$ such that the solution has the correct sparsity pattern, then the group Lasso estimate with $\lambda_n \to \lambda_0$, will have a consistent sparsity pattern. The following proposition, which can be proved with standard M-estimation arguments, make this precise:

Proposition 5 *Assume (**A1-3**). If λ_n tends to $\lambda_0 > 0$, then the group Lasso estimate $\hat{w}$ is sparsity-consistent if and only if the solution of Eq. (7) has the correct sparsity pattern.*

Thus, even when condition (5) is not satisfied, we may have consistent estimation of the sparsity pattern but inconsistent estimation of the loading vectors. We provide in Section 5 such examples.

2.7 Probability of Correct Pattern Selection

In this section, we focus on regularization parameters that tend to zero, at the rate $n^{-1/2}$, that is, $\lambda_n = \lambda_0 n^{-1/2}$ with $\lambda_0 > 0$. For this particular setting, we can actually compute the limit of the probability of correct pattern selection (proposition proved in Appendix B.4). Note that in order to obtain a simpler result, we assume constant conditional variance of Y given $\mathbf{w}^\top X$:

Proposition 6 *Assume (**A1-3**) and $\operatorname{var}(Y|\mathbf{w}^\top x) = \sigma^2$ almost surely. Assume moreover $\lambda_n = \lambda_0 n^{-1/2}$ with $\lambda_0 > 0$. Then, the group Lasso $\hat{w}$ converges in probability to $\mathbf{w}$ and the probability of correct sparsity pattern selection has the following limit:*

$$\mathbb{P}\left(\max_{i \in \mathbf{J}^c} \frac{1}{d_i} \left\| \frac{\sigma}{\lambda_0} t_i - \Sigma_{X_i X_\mathbf{J}} \Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1} \operatorname{Diag}\left(\frac{d_j}{\|\mathbf{w}_j\|} \right) \mathbf{w}_\mathbf{J} \right\| \leqslant 1 \right), \tag{8}$$

where t is normally distributed with mean zero and covariance matrix $\Sigma_{X_{\mathbf{J}^c} X_{\mathbf{J}^c} | X_\mathbf{J}} = \Sigma_{X_{\mathbf{J}^c} X_{\mathbf{J}^c}} - \Sigma_{X_{\mathbf{J}^c} X_\mathbf{J}} \Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1} \Sigma_{X_\mathbf{J} X_{\mathbf{J}^c}}$ (which is the conditional covariance matrix of $X_{\mathbf{J}^c}$ given $X_\mathbf{J}$).

The previous theorem states that the probability of correct selection tends to the mass under a non degenerate multivariate distribution of the intersection of cylinders. Under our assumptions, this set is never empty and thus the limiting probability is strictly positive, that is, there is (asymptotically)

always a positive probability of estimating the correct pattern of groups (see Bach, 2008a, for application of this result to model consistent estimation of a bootstrapped version of the Lasso, with no consistency condition).

Moreover, additional insights may be gained from Proposition 6, namely in terms of the dependence on σ, λ_0 and the tightness of the consistency conditions. First, when λ_0 tends to infinity, then the limit defined in Eq. (8) tends to one if the strict consistency condition (4) is satisfied, and tends to zero if one of the conditions is strictly not met. This corroborates the results of Theorem 2 and 3. Note however, that only an extension of Proposition 6 to λ_n that may deviate from a $n^{-1/2}$ would actually lead to a proof of Theorem 2, which is a subject of ongoing research.

Finally, Eq. (8) shows that σ has a smoothing effect on the probability of correct pattern selection, that is, if condition (4) is satisfied, then this probability is a decreasing function of σ (and an increasing function of λ_0). Finally, the stricter the inequality in Eq. (4), the larger the probability of correct rank selection, which is illustrated in Section 5 on synthetic examples.

2.8 Loading Independent Sufficient Condition

Condition (4) depends on the loading vector $\mathbf{w}$ and on the sparsity pattern $\mathbf{J}$, which are both a priori unknown. In this section, we consider sufficient conditions that do not depend on the loading vector, but only on the sparsity pattern $\mathbf{J}$ and of course on the covariance matrices. The following condition is sufficient for consistency of the group Lasso, for all possible loading vectors $\mathbf{w}$ with sparsity pattern $\mathbf{J}$:

$$C(\Sigma_{XX}, d, \mathbf{J}) = \max_{i \in \mathbf{J}^c} \max_{\forall j \in \mathbf{J},\ \|u_j\|=1} \left\| \frac{1}{d_i} \Sigma_{X_i X_\mathbf{J}} \Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1} \operatorname{Diag}(d_j) u_\mathbf{J} \right\| < 1. \tag{9}$$

As opposed to the Lasso case, $C(\Sigma_{XX}, d, \mathbf{J})$ cannot be readily computed in closed form, but we have the following upper bound:

$$C(\Sigma_{XX}, d, \mathbf{J}) \leqslant \max_{i \in \mathbf{J}^c} \frac{1}{d_i} \sum_{j \in \mathbf{J}} d_j \left\| \sum_{k \in \mathbf{J}} \Sigma_{X_i X_k} \left(\Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1} \right)_{kj} \right\|,$$

where for a matrix M, $\|M\|$ denotes its maximal singular value (also known as its spectral norm). This leads to the following sufficient condition for consistency of the group Lasso (which extends the condition of Yuan and Lin, 2007):

$$\max_{i \in \mathbf{J}^c} \frac{1}{d_i} \sum_{j \in \mathbf{J}} d_j \left\| \sum_{k \in \mathbf{J}} \Sigma_{X_i X_k} \left(\Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1} \right)_{kj} \right\| < 1. \tag{10}$$

Given a set of weights d, better sufficient conditions than Eq. (10) may be obtained by solving a semidefinite programming problem (Boyd and Vandenberghe, 2003):

Proposition 7 *The quantity* $\max_{\forall j \in \mathbf{J},\ \|u_j\|=1} \left\| \Sigma_{X_i X_\mathbf{J}} \Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1} \operatorname{Diag}(d_j) u_\mathbf{J} \right\|^2$ *is upperbounded by*

$$\max_{M \succcurlyeq 0,\ \operatorname{tr} M_{ii}=1} \operatorname{tr} M \left(\operatorname{Diag}(d_j) \Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1} \Sigma_{X_\mathbf{J} X_i} \Sigma_{X_i X_\mathbf{J}} \Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1} \operatorname{Diag}(d_j) \right), \tag{11}$$

where M is a matrix defined by blocks following the block structure of $\Sigma_{X_J X_J}$. Moreover, the bound is also equal to

$$\min_{\lambda\in\mathbb{R}^m,\ \mathrm{Diag}(d_j)\Sigma^{-1}_{X_J X_J}\Sigma_{X_J X_i}\Sigma_{X_i X_J}\Sigma^{-1}_{X_J X_J}\mathrm{Diag}(d_j)\preccurlyeq\mathrm{Diag}(\lambda)} \sum_{j=1}^m \lambda_j.$$

Proof We denote $M = uu^\top \succcurlyeq 0$. Then if all u_j for $j \in \mathbf{J}$ have norm 1, then we have $\mathrm{tr} M_{jj} = 1$ for all $j \in \mathbf{J}$. This implies the convex relaxation. The second problem is easily obtained as the convex dual of the first problem (Boyd and Vandenberghe, 2003). ■

Note that for the Lasso, the convex bound in Eq. (11) is tight and leads to the bound given above in Eq. (10) (Yuan and Lin, 2007; Wainwright, 2006). For the Lasso, Zhao and Yu (2006) consider several particular patterns of dependencies using Eq. (10). Note that this condition (and not the condition in Eq. 9) is independent from the dimension and thus does not readily lead to rules of thumbs allowing to set the weight d_j as a function of the dimension p_j; several rules of thumbs have been suggested, that loosely depend on the dimension on the blocks, in the context of the linear group Lasso (Yuan and Lin, 2006) or multiple kernel learning (Bach et al., 2004b); we argue in this paper, that weights should also depend on the response as well (see Section 4).

2.9 Alternative Formulation of the Group Lasso

Following Bach et al. (2004a), we can instead consider regularization by the square of the block ℓ_1-norm:

$$\min_{w\in\mathbb{R}^p,\ b\in\mathbb{R}} \frac{1}{2n}\|\bar{Y} - \bar{X}w - b1_n\|^2 + \frac{1}{2}\mu_n\left(\sum_{j=1}^m d_j\|w_j\|\right)^2.$$

This leads to the same path of solutions, but it is better behaved because each variable which is not zero is still regularized by the squared norm. The alternative version has also two advantages: (a) it has very close links to more general frameworks for learning the kernel matrix from data (Lanckriet et al., 2004b), and (b) it is essential in our proof of consistency in the functional case. We also get the equivalent formulation to Eq. (1), by minimizing in closed form with respect to b, to obtain:

$$\min_{w\in\mathbb{R}^p} \frac{1}{2}\hat{\Sigma}_{YY} - \hat{\Sigma}_{YX}w + \frac{1}{2}w^\top\hat{\Sigma}_{XX}w + \frac{1}{2}\mu_n\left(\sum_{j=1}^m d_j\|w_j\|\right)^2. \tag{12}$$

The following proposition gives the optimality conditions for the convex optimization problem defined in Eq. (12) (see proof in Appendix A.2):

Proposition 8 *A vector $w \in \mathbb{R}^p$ with sparsity pattern $J = \{j,\ w_j \neq 0\}$ is optimal for problem (12) if and only if*

$$\begin{aligned}
\forall j \in J^c, \qquad & \left\|\hat{\Sigma}_{X_jX}w - \hat{\Sigma}_{X_jY}\right\| \leqslant \mu_n d_j\left(\textstyle\sum_{i=1}^n d_i\|w_i\|\right),\\
\forall j \in J, \qquad & \hat{\Sigma}_{X_jX}w - \hat{\Sigma}_{X_jY} = -\mu_n\left(\textstyle\sum_{i=1}^n d_i\|w_i\|\right)\dfrac{d_j w_j}{\|w_j\|}.
\end{aligned}$$

Note the correspondence at the optimum between optimal solutions of the two optimization problems in Eq. (1) and Eq. (12) through $\lambda_n = \mu_n\left(\sum_{i=1}^n d_i\|w_i\|\right)$. As far as consistency results are concerned, Theorem 3 immediately applies to the alternative formulation because the regularization

paths are the same. For Theorem 2, it does not readily apply. But since the relationship between λ_n and μ_n at optimum is $\lambda_n = \mu_n(\sum_{i=1}^n d_i \|w_i\|)$ and that $\sum_{i=1}^n d_i \|\hat{w}_i\|$ converges to a constant whenever $\hat{w}$ is consistent, it does apply as well with minor modifications (in particular, to deal with the case where $\mathbf{J}$ is empty, which requires $\mu_n = \infty$).

3. Covariance Operators and Multiple Kernel Learning

We now extend the previous consistency results to the case of nonparametric estimation, where each group is a potentially infinite dimensional space of functions. Namely, the nonparametric group Lasso aims at estimating a sparse linear combination of functions of separate random variables, and can then be seen as a variable selection method in a generalized additive model (Hastie and Tibshirani, 1990). Moreover, as shown in Section 3.5, the nonparametric group Lasso may also be seen as equivalent to learning a convex combination of kernels, a framework referred to as multiple kernel learning (MKL). In this context it is customary to have a single input space with several kernels (and hence Hilbert spaces) defined on the same input space (Lanckriet et al., 2004b; Bach et al., 2004a).[3] Our framework accommodates this case as well, but our assumption (**A5**) regarding the invertibility of the joint correlation operator states that the kernels cannot span Hilbert spaces which intersect.

In this nonparametric context, covariance operators constitute appropriate tools for the statistical analysis and are becoming standard in the theoretical analysis of kernel methods (Fukumizu et al., 2004; Gretton et al., 2005; Fukumizu et al., 2007; Caponnetto and de Vito, 2005). The following section reviews important concepts. For more details, see Baker (1973) and Fukumizu et al. (2004).

3.1 Review of Covariance Operator Theory

In this section, we first consider a single set $\mathcal{X}$ and a positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, associated with the reproducing kernel Hilbert space (RKHS) $\mathcal{F}$ of functions from $\mathcal{X}$ to $\mathbb{R}$ (see, e.g., Schölkopf and Smola 2001 or Berlinet and Thomas-Agnan 2003 for an introduction to RKHS theory). The Hilbert space and its dot product $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ are such that for all $x \in \mathcal{X}$, then $k(\cdot, x) \in \mathcal{F}$ and for all $f \in \mathcal{F}$, $\langle k(\cdot, x), f \rangle_{\mathcal{F}} = f(x)$, which leads to the *reproducing property* $\langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{F}} = k(x, y)$ for any $(x, y) \in \mathcal{X} \times \mathcal{X}$.

3.1.1 Covariance Operator and Norms

Given a random variable X on $\mathcal{X}$ with bounded second order moment, that is, such that $\mathbb{E}k(X, X) < \infty$, we can define the covariance operator as the bounded linear operator Σ_{XX} from $\mathcal{F}$ to $\mathcal{F}$ such that for all $(f, g) \in \mathcal{F} \times \mathcal{F}$,

$$\langle f, \Sigma_{XX} g \rangle_{\mathcal{F}} = \operatorname{cov}(f(X), g(X)) = \mathbb{E}(f(X)g(X)) - (\mathbb{E}f(X))(\mathbb{E}g(X)).$$

The operator Σ_{XX} is *auto-adjoint*, *non-negative* and *Hilbert-Schmidt*, that is, for any orthonormal basis $(e_p)_{p \geqslant 1}$ of $\mathcal{F}$, then $\sum_{p=1}^{\infty} \|\Sigma_{XX} e_p\|_{\mathcal{F}}^2$ is finite; in this case, the value does not depend on the chosen basis and is referred to as the square of the Hilbert-Schmidt norm. The norm that we use by default in this paper is the operator norm $\|\Sigma_{XX}\|_{\mathcal{F}} = \sup_{f \in \mathcal{F},\ \|f\|_{\mathcal{F}} = 1} \|\Sigma_{XX} f\|_{\mathcal{F}}$, which is dominated by the Hilbert-Schmidt norm. Note that in the finite dimensional case where $\mathcal{X} = \mathbb{R}^p$, $p > 0$ and the

3. Note that the grouplasso can be explicitly seen as a special case of multiple kernel learning. Using notations from Section 2, this is done by considering $X = (X_1, \ldots, X_m)^\top \in \mathbb{R}^m$ and the m kernels $k_m(X, Y) = X_m^\top Y_m$.

kernel is linear, the covariance operator is exactly the covariance matrix, and the Hilbert-Schmidt norm is the Frobenius norm, while the operator norm is the maximum singular value (also referred to as the spectral norm).

The null space of the covariance operator is the space of functions $f \in \mathcal{F}$ such that $\operatorname{var} f(X) = 0$, that is, such that f is constant on the support of X.

3.1.2 EMPIRICAL ESTIMATORS

Given data $x_i \in \mathcal{X}, i = 1, \ldots, n$, sampled i.i.d. from P_X, then the empirical estimate $\hat{\Sigma}_{XX}$ of Σ_{XX} is defined such that $\langle f, \hat{\Sigma}_{XX} g \rangle_{\mathcal{F}}$ is the empirical covariance between $f(X)$ and $g(X)$, which leads to:

$$\hat{\Sigma}_{XX} = \frac{1}{n}\sum_{i=1}^{n} k(\cdot, x_i) \otimes k(\cdot, x_i) - \frac{1}{n}\sum_{i=1}^{n} k(\cdot, x_i) \otimes \frac{1}{n}\sum_{i=1}^{n} k(\cdot, x_i),$$

where $u \otimes v$ is the operator defined by $\langle f, (u \otimes v) g \rangle_{\mathcal{F}} = \langle f, u \rangle_{\mathcal{F}} \langle g, v \rangle_{\mathcal{F}}$. If we further assume that the fourth order moment is finite, that is, $\mathbb{E} k(X, X)^2 < \infty$, then the estimate is uniformly consistent, that is, $\|\hat{\Sigma}_{XX} - \Sigma_{XX}\|_{\mathcal{F}} = O_p(n^{-1/2})$ (see Fukumizu et al., 2007, and Appendix C.1), which generalizes the usual result from finite dimension.[4]

3.1.3 CROSS-COVARIANCE AND JOINT COVARIANCE OPERATORS

Covariance operator theory can be extended to cases with more than one random variables (Baker, 1973). In our situation, we have m input spaces $\mathcal{X}_1, \ldots, \mathcal{X}_m$ and m random variables $X = (X_1, \ldots, X_m)$ and m RKHS $\mathcal{F}_1, \ldots, \mathcal{F}_m$ associated with m kernels $k_1, \ldots, k_m$.

If we assume that $\mathbb{E} k_j(X_j, X_j) < \infty$, for all $j = 1, \ldots, m$, then we can naturally define the cross-covariance operators $\Sigma_{X_i X_j}$ from $\mathcal{F}_j$ to $\mathcal{F}_i$ such that $\forall (f_i, f_j) \in \mathcal{F}_i \times \mathcal{F}_j$,

$$\langle f_i, \Sigma_{X_i X_j} f_j \rangle_{\mathcal{F}_i} = \operatorname{cov}(f_i(X_i), f_j(X_j)) = \mathbb{E}(f_i(X_i) f_j(X_j)) - (\mathbb{E} f_i(X_i))(\mathbb{E} f_j(X_j)).$$

These are also Hilbert-Schmidt operators, and if we further assume that $\mathbb{E} k_j(X_j, X_j)^2 < \infty$, for all $j = 1, \ldots, m$, then the natural empirical estimators converges to the population quantities in Hilbert-Schmidt and operator norms at rate $O_p(n^{-1/2})$. We can now define a joint block covariance operator on $\mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_m$ following the block structure of covariance matrices in Section 2. As in the finite dimensional case, it leads to a joint covariance operator Σ_{XX} and we can refer to sub-blocks as $\Sigma_{X_I X_J}$ for the blocks indexed by I and J.

Moreover, we can define the bounded (i.e., with finite operator norm) correlation operators through $\Sigma_{X_i X_j} = \Sigma_{X_i X_i}^{1/2} C_{X_i X_j} \Sigma_{X_j X_j}^{1/2}$ (Baker, 1973). Throughout this paper we will make the assumption that those operators $C_{X_i X_j}$ are *compact* for $i \neq j$: compact operators can be characterized as limits of finite rank operators or as operators that can be diagonalized on a countable basis with spectrum composed of a sequence tending to zero (see, e.g., Brezis, 1980). This implies that the joint operator C_{XX}, naturally defined on $\mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_m$, is of the form "identity plus compact". It thus has a minimum and a maximum eigenvalue which are both between 0 and 1 (Brezis, 1980). If those eigenvalues are strictly greater than zero, then the operator is invertible, as are all the square sub-blocks. Moreover, the joint correlation operator is lower-bounded by a strictly positive constant times the identity operator.

4. A random variable Z_n is said to be of order $O_p(a_n)$ if for any $\eta > 0$, there exists $M > 0$ such that $\sup_n \mathbb{P}(|Z_n| > M a_n) < \eta$. See Van der Vaart (1998) for further definitions and properties of asymptotics in probability.

3.1.4 TRANSLATION INVARIANT KERNELS

A particularly interesting ensemble of RKHS in the context of nonparametric estimation is the set of translation invariant kernels defined over $\mathcal{X} = \mathbb{R}^p$, where $p \geqslant 1$, of the form $k(x,x') = q(x'-x)$ where q is a function on $\mathbb{R}^p$ with pointwise nonnegative integrable Fourier transform (which implies that q is continuous). In this case, the associated RKHS is $\mathcal{F} = \{q_{1/2} * g,\ g \in L^2(\mathbb{R}^p)\}$, where $q_{1/2}$ denotes the inverse Fourier transform of the square root of the Fourier transform of q and $*$ denotes the convolution operation, and $L^2(\mathbb{R}^p)$ denotes the space of square integrable functions. The norm is then equal to

$$\|f\|_{\mathcal{F}}^2 = \int \frac{|F(\omega)|^2}{Q(\omega)} d\omega,$$

where F and Q are the Fourier transforms of f and q (Wahba, 1990; Schölkopf and Smola, 2001). Functions in the RKHS are functions with appropriately integrable derivatives. In this paper, when using infinite dimensional kernels, we use the Gaussian kernel $k(x,x') = q(x-x') = \exp(-b\|x-x'\|^2)$, with $b > 0$.

3.1.5 ONE-DIMENSIONAL HILBERT SPACES

In this paper, we also consider real random variables Y and ε embedded in the natural Euclidean structure of real numbers (i.e., we consider the linear kernel on $\mathbb{R}$). In this setting the covariance operator Σ_{X_jY} from $\mathbb{R}$ to $\mathcal{F}_j$ can be canonically identified as an element of $\mathcal{F}_j$. Throughout this paper, we always use this identification.

3.2 Problem Formulation

We assume in this section and in the remaining of the paper that for each $j = 1,\dots,m$, $X_j \in \mathcal{X}_j$ where $\mathcal{X}_j$ is any set on which we have a reproducible kernel Hilbert spaces $\mathcal{F}_j$, associated with the positive kernel $k_j : \mathcal{X}_j \times \mathcal{X}_j \to \mathbb{R}$. We now make the following assumptions, that extend the assumptions (**A1**), (**A2**) and (**A3**). For each of them, we detail the main implications as well as common natural sufficient conditions. The first two conditions (**A4**) and (**A5**) depend solely on the input variables, while the two other ones, (**A6**) and (**A7**) consider the relationship between X and Y.

(**A4**) For each $j = 1\dots,m$, $\mathcal{F}_j$ is a separable reproducing kernel Hilbert space associated with kernel k_j, and the random variables $k_j(\cdot,X_j)$ are not constant and have finite fourth-order moments, that is, $\mathbb{E}k_j(X_j,X_j)^2 < \infty$.

This is a non restrictive assumption in many situations; for example, when (a) $\mathcal{X}_j = \mathbb{R}^{p_j}$ and the kernel function (such as the Gaussian kernel) is bounded, or when (b) $\mathcal{X}_j$ is a compact subset of $\mathbb{R}^{p_j}$ and the kernel is any continuous function such as linear or polynomial. This implies notably, as shown in Section 3.1, that we can define covariance, cross-covariance and correlation operators that are all Hilbert-Schmidt (Baker, 1973; Fukumizu et al., 2007) and can all be estimated at rate $O_p(n^{-1/2})$ in operator norm.

(**A5**) All cross-correlation operators are compact and the joint correlation operator C_{XX} is invertible.

This is also a condition uniquely on the input spaces and not on Y. Following Fukumizu et al. (2007), a simple sufficient condition is that we have measurable spaces and distributions with joint

density p_X (and marginal distributions $p_{X_i}(x_i)$ and $p_{X_iX_j}(x_i,x_j)$) and that the *mean square contingency* between all pairs of variables is finite, that is,

$$\mathbb{E}\left\{\frac{p_{X_iX_j}(x_i,x_j)}{p_{X_i}(x_i)p_{X_j}(x_j)}-1\right\}<\infty.$$

The contingency is a measure of statistical dependency (Renyi, 1959), and thus this sufficient condition simply states that two variables X_i and X_j cannot be too dependent. In the context of multiple kernel learning for heterogeneous data fusion, this corresponds to having sources which are heterogeneous enough. On top of compacity we impose the invertibility of the joint correlation operator; we use this assumption to make sure that the functions $\mathbf{f}_1,\dots,\mathbf{f}_m$ are unique. This ensures the non existence of any set of functions $f_1,\dots,f_m$ in the closures of $\mathcal{F}_1,\dots,\mathcal{F}_m$, such that $\operatorname{var} f_j(X_j)>0$, for all j, and a linear combination is constant on the support of the random variables. In the context of generalized additive models, this assumption is referred to as the empty *concurvity space* assumption (Hastie and Tibshirani, 1990).

(A6) There exists functions $\mathbf{f}=(\mathbf{f}_1,\dots,\mathbf{f}_m)\in\mathcal{F}=\mathcal{F}_1\times\cdots\times\mathcal{F}_m$, $\mathbf{b}\in\mathbb{R}$, and a function $\mathbf{h}$ of $X=(X_1,\dots,X_m)$ such that $\mathbb{E}(Y|X)=\sum_{j=1}^m\mathbf{f}_j(X_j)+\mathbf{b}+\mathbf{h}(X)$ with $\mathbb{E}\mathbf{h}(X)^2<\infty$, $\mathbb{E}\mathbf{h}(X)=0$ and $\mathbb{E}\mathbf{h}(X)f_j(X_j)=0$ for all $j=1,\dots,m$ and $f_j\in\mathcal{F}_j$. We assume that $\mathbb{E}((Y-\mathbf{f}(X)-\mathbf{b})^2|X)$ is almost surely greater than $\sigma^2_{\min}>0$ and smaller than $\sigma^2_{\max}<\infty$. We denote by $\mathbf{J}=\{j,\mathbf{f}_j\neq 0\}$ the sparsity pattern of $\mathbf{f}$.

This assumption on the conditional expectation of Y given X is not the most general and follows common assumptions in approximation theory (see, e.g., Caponnetto and de Vito, 2005; Cucker and Smale, 2002, and references therein). It allows misspecification, but it essentially requires that the conditional expectation of Y given sums of measurable functions of X_j is attained at functions in the RKHS, and not merely measurable functions. Dealing with more general assumptions in the line of Ravikumar et al. (2008) requires to consider consistency for norms weaker than the RKHS norms (Caponnetto and de Vito, 2005; Steinwart, 2001), and is left for future research. Note also, that to simplify proofs, we assume a finite upper-bound $\sigma^2_{\max}$ on the residual variance.

(A7) For all $j\in\{1,\dots,m\}$, there exists $\mathbf{g}_j\in\mathcal{F}_j$ such that $\mathbf{f}_j=\Sigma^{1/2}_{X_jX_j}\mathbf{g}_j$, that is, each $\mathbf{f}_j$ is in the range of $\Sigma^{1/2}_{X_jX_j}$.

This technical condition, already used by Caponnetto and de Vito (2005), which concerns all RKHS independently, ensures that we obtain consistency for the norm of the RKHS (and not another weaker norm) for the least-squares estimates. Note also that it implies that $\operatorname{var} f_j(X_j)>0$, that is, f_j is not constant on the support of X_j.

This assumption might be checked (at least) in two ways; first, if $(e_p)_{p\geqslant 1}$ is a sequence of eigenfunctions of Σ_{XX}, associated with strictly positive eigenvalues $\lambda_p>0$, then f is in the range of Σ_{XX} if and only if f is constant outside the support of the random variable X and $\sum_{p\geqslant 1}\frac{1}{\lambda_p}\langle f,e_p\rangle^2$ is finite (i.e., the decay of the sequence $\langle f,e_p\rangle^2$ is strictly faster than λ_p).

We also provide another sufficient condition that sheds additional light on this technical condition which is always true for finite dimensional Hilbert spaces. For the common situation where $\mathcal{X}_j=\mathbb{R}^{p_j}$, P_{X_j} (the marginal distribution of X_j) has a density $p_{X_j}(x_j)$ with respect to the Lebesgue measure and the kernel is of the form $k_j(x_j,x_j')=q_j(x_j-x_j')$, we have the following proposition (proved in Appendix C.5):

Proposition 9 *Assume $\mathcal{X} = \mathbb{R}^p$ and X is a random variable on $\mathcal{X}$ with distribution P_X that has a* strictly positive *density $p_X(x)$ with respect to the Lebesgue measure. Assume $k(x,x') = q(x-x')$ for a function $q \in L^2(\mathbb{R}^p)$ has an integrable pointwise positive Fourier transform, with associated RKHS $\mathcal{F}$. If f can be written as $f = q * g$ (convolution of q and g) with $\int_{\mathbb{R}^p} g(x)dx = 0$ and $\int_{\mathbb{R}^p} \frac{g(x)^2}{p_X(x)}dx < \infty$, then $f \in \mathcal{F}$ is in the range of the square root $\Sigma_{XX}^{1/2}$ of the covariance operator.*

The previous proposition gives natural conditions regarding f and p_X. Indeed, the condition $\int \frac{g(x)^2}{p_X(x)}dx < \infty$ corresponds to a natural support condition, that is, f should be zero where X has no mass, otherwise, we will not be able to estimate f; note the similarity with the usual condition regarding the variance of importance sampling estimation (Brémaud, 1999). Moreover, f should be even smoother than a regular function in the RKHS (convolution by q instead of the square root of q). Finally, we provide in Appendix E detailed covariance structures for Gaussian kernels with Gaussian variables.

3.2.1 NOTATIONS

Throughout this section, we refer to functions $f = (f_1, \dots, f_m) \in \mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_m$ and the joint covariance operator Σ_{XX}. In the following, we always use the norms of the RKHS. When considering operators, we use the operator norm. We also refer to a subset of f indexed by J through f_J. Note that the Hilbert norm $\|f_J\|_{\mathcal{F}_J}$ is equal to $\|f_J\|_{\mathcal{F}_J} = (\sum_{j\in J} \|f_j\|_{\mathcal{F}_j})^{1/2}$. Finally, given a nonnegative auto-adjoint operator S, we denote by $S^{1/2}$ its nonnegative autoadjoint square root (Baker, 1973).

3.3 Nonparametric Group Lasso

Given i.i.d data (x_{ij}, y_i), $i = 1, \dots, n$, $j = 1, \dots, m$, where each $x_{ij} \in \mathcal{X}_j$, our goal is to estimate consistently the functions $\mathbf{f}_j$ and which of them are zero. We denote by $\overline{Y} \in \mathbb{R}^n$ the vector of responses. We consider the following optimization problem:

$$\min_{f\in\mathcal{F},\, b\in\mathbb{R}} \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{m} f_j(x_{ij}) - b\right)^2 + \frac{\mu_n}{2}\left(\sum_{j=1}^{m} d_j \|f_j\|_{\mathcal{F}_j}\right)^2.$$

By minimizing with respect to b in closed form, we obtain a similar formulation to Eq. (12), where empirical covariance matrices are replaced by empirical covariance operators:

$$\min_{f\in\mathcal{F}} \frac{1}{2}\hat{\Sigma}_{YY} - \langle f, \hat{\Sigma}_{XY}\rangle_{\mathcal{F}} + \frac{1}{2}\langle f, \hat{\Sigma}_{XX} f\rangle_{\mathcal{F}} + \frac{\mu_n}{2}\left(\sum_{j=1}^{m} d_j \|f_j\|_{\mathcal{F}_j}\right)^2. \tag{13}$$

We denote by $\hat{f}$ any minimizer of Eq. (13), and we refer to it as the nonparametric group Lasso estimate, or also the multiple kernel learning estimate. By Proposition 13, the previous problem has indeed minimizers, and by Proposition 14 this global minimum is unique with probability tending to one.

Note that formally, the finite and infinite dimensional formulations in Eq. (12) and Eq. (13) are the same, and this is the main reason why covariance operators are very practical tools for the analysis. Furthermore, we have the corresponding proposition regarding optimality conditions (see proof in Appendix A.3):

Proposition 10 *A function $f \in \mathcal{F}$ with sparsity pattern $J = J(f) = \{j,\ f_j \neq 0\}$ is optimal for problem (13) if and only if*

$$\forall j \in J^c, \qquad \left\|\hat{\Sigma}_{X_jX} f - \hat{\Sigma}_{X_jY}\right\|_{\mathcal{F}_j} \leqslant \mu_n d_j \left(\sum_{i=1}^n d_i \|f_i\|_{\mathcal{F}_i}\right), \tag{14}$$

$$\forall j \in J, \qquad \hat{\Sigma}_{X_jX} f - \hat{\Sigma}_{X_jY} = -\mu_n \left(\sum_{i=1}^n d_i \|f_i\|_{\mathcal{F}_i}\right) \frac{d_j f_j}{\|f_j\|_{\mathcal{F}_j}}. \tag{15}$$

A consequence (and in fact the first part of the proof) is that an optimal function f must be in the range of $\hat{\Sigma}_{XY}$ and $\hat{\Sigma}_{XX}$, that is, an optimal f is supported by the data; that is, each f_j is a linear combination of functions $k_j(\cdot, x_{ij})$, $i = 1, \ldots, n$. This is a rather circumvoluted way of presenting the representer theorem (Wahba, 1990), but this is the easiest for the theoretical analysis of consistency. However, to actually compute the estimate $\hat{f}$ from data, we need the usual formulation with dual parameters (see Section 3.5).

Moreover, one important conclusion is that all our optimization problems in spaces of functions can be in fact transcribed into finite-dimensional problems. In particular, all notions from multivariate differentiable calculus may be used without particular care regarding the infinite dimension.

3.4 Consistency Results

We consider the following strict and weak conditions, which correspond to condition (4) and (5) in the finite dimensional case:

$$\max_{i \in \mathbf{J}^c} \frac{1}{d_i} \left\| \Sigma_{X_iX_i}^{1/2} C_{X_iX_\mathbf{J}} C_{X_\mathbf{J}X_\mathbf{J}}^{-1} \operatorname{Diag}(d_j / \|\mathbf{f}_j\|_{\mathcal{F}_j}) \mathbf{g}_\mathbf{J} \right\|_{\mathcal{F}_i} < 1, \tag{16}$$

$$\max_{i \in \mathbf{J}^c} \frac{1}{d_i} \left\| \Sigma_{X_iX_i}^{1/2} C_{X_iX_\mathbf{J}} C_{X_\mathbf{J}X_\mathbf{J}}^{-1} \operatorname{Diag}(d_j / \|\mathbf{f}_j\|_{\mathcal{F}_j}) \mathbf{g}_\mathbf{J} \right\|_{\mathcal{F}_i} \leqslant 1, \tag{17}$$

where $\operatorname{Diag}(d_j / \|\mathbf{f}_j\|_{\mathcal{F}_j})$ denotes the block-diagonal operator with operators $\frac{d_j}{\|\mathbf{f}_j\|_{\mathcal{F}_j}} I_{\mathcal{F}_j}$ on the diagonal. Note that this is well-defined because C_{XX} is invertible and that it reduces to Eq. (4) and Eq. (5) when the input spaces $\mathcal{X}_j$, $j = 1, \ldots, m$ are of the form $\mathbb{R}^{p_j}$ and the kernels are linear. The main reason of rewriting the conditions in terms of correlation operators rather than covariance operators is that correlation operators are invertible by assumption, while covariance operators are not as soon as the Hilbert spaces have infinite dimensions. The following theorems give necessary and sufficient conditions for the path consistency of the nonparametric group Lasso (see proofs in Appendix C.2 and Appendix C.3):

Theorem 11 *Assume (**A4-7**) and that $\mathbf{J}$ is not empty. If condition (16) is satisfied, then for any sequence μ_n such that $\mu_n \to 0$ and $\mu_n n^{1/2} \to +\infty$, any sequence of nonparametric group Lasso estimates $\hat{f}$ converges in probability to $\mathbf{f}$ and the sparsity pattern $J(\hat{f}) = \{j, \hat{f}_j \neq 0\}$ converges in probability to $\mathbf{J}$.*

Theorem 12 *Assume (**A4-7**) and that $\mathbf{J}$ is not empty. If there exists a (possibly data-dependent) sequence μ_n such $\hat{f}$ converges to $\mathbf{f}$ and $\hat{J}$ converges to $\mathbf{J}$ in probability, then condition (17) is satisfied.*

Essentially, the results in finite dimension also hold when groups have infinite dimensions. We leave the extensions of the refined results in Section 2.6 to future work. Condition (16) might be hard to check in practice since it involves inversion of correlation operators; see Section 3.6 for an estimate from data.

3.5 Multiple Kernel Learning Formulation

Proposition 10 does not readily lead to an algorithm for computing the estimate $\hat{f}$. In this section, following Bach et al. (2004a), we link the group Lasso to the multiple kernel learning framework (Lanckriet et al., 2004b). Problem (13) is an optimization problem on a potentially infinite dimensional space of functions. However, the following proposition shows that it reduces to a finite dimensional problem that we now precise (see proof in Appendix A.4):

Proposition 13 *The dual of problem (13) is*

$$\max_{\alpha\in\mathbb{R}^n,\ \alpha^\top 1_n=0} \left\{ -\frac{1}{2n}\|\bar{Y}-n\mu_n\alpha\|^2 - \frac{1}{2\mu_n}\max_{i=1,\ldots,m}\frac{\alpha^\top K_i\alpha}{d_i^2}\right\}, \tag{18}$$

where $(K_i)_{ab}=k_i(x_a,x_b)$ are the kernel matrices in $\mathbb{R}^{n\times n}$, for $i=1,\ldots,m$. Moreover, the dual variable $\alpha\in\mathbb{R}^n$ is optimal if and only if $\alpha^\top 1_n=0$ and there exists $\eta\in\mathbb{R}_+^m$ such that $\sum_{j=1}^m \eta_j d_j^2=1$ and

$$\left(\sum_{j=1}^m \eta_j K_j + n\mu_n I_n\right)\alpha = \bar{Y}, \tag{19}$$

$$\forall j\in\{1,\ldots,m\},\ \frac{\alpha^\top K_j\alpha}{d_j^2} < \max_{i=1,\ldots,m}\frac{\alpha^\top K_i\alpha}{d_i^2} \Rightarrow \eta_j=0.$$

The optimal function may then be written as $f_j=\eta_j\sum_{i=1}^n \alpha_i k_j(\cdot,x_{ij})$.

Since the problem in Eq. (18) is strictly convex, there is a unique dual solution α. Note that Eq. (19) corresponds to the optimality conditions for the least-square problem:

$$\min_{f\in\mathcal{F}} \frac{1}{2}\hat{\Sigma}_{YY} - \langle f,\hat{\Sigma}_{XY}\rangle_{\mathcal{F}} + \frac{1}{2}\langle f,\hat{\Sigma}_{XX}f\rangle_{\mathcal{F}} + \frac{1}{2}\mu_n \sum_{j,\ \eta_j>0}\frac{\|f_j\|_{\mathcal{F}_j}^2}{\eta_i},$$

whose dual problem is:

$$\max_{\alpha\in\mathbb{R}^n,\ \alpha^\top 1_n=0}\left\{-\frac{1}{2n}\|\bar{Y}-n\mu_n\alpha\|^2 - \frac{1}{2\mu_n}\alpha^\top\left(\sum_{j=1}^m \eta_i K_i\right)\alpha\right\},$$

and unique solution is $\alpha=\Pi_n(\sum_{j=1}^m \eta_j\Pi_n K_j\Pi_n + n\mu_n I_n)^{-1}\Pi_n\bar{Y}$. That is, the solution of the MKL problem leads to dual parameters α and set of weights $\eta\geqslant 0$ such that α is the solution to the least-square problem with kernel $K=\sum_{j=1}^m \eta_j K_j$. Bach et al. (2004a) has shown in a similar context (hinge loss instead of the square loss) that the optimal η in Proposition 13 can be obtained as the minimizer of the optimal value of the regularized least-square problem with kernel matrix $\sum_{j=1}^m \eta_j K_j$, that is:

$$J(\eta)=\max_{\alpha\in\mathbb{R}^n,\ \alpha^\top 1_n=0}\left\{-\frac{1}{2n}\|\bar{Y}-n\mu_n\alpha\|^2 - \frac{1}{2\mu_n}\alpha^\top\left(\sum_{j=1}^m \eta_j K_j\right)\alpha\right\},$$

with respect to $\eta\geqslant 0$ such that $\sum_{j=1}^m \eta_j d_j^2=1$. This formulation allows to derive probably approximately correct error bounds (Lanckriet et al., 2004b; Bousquet and Herrmann, 2003). Besides,

this formulation allows η to be negative, as long as the matrix $\sum_{j=1}^m \eta_j K_j$ is positive semi-definite. However, theoretical advantages of such a possibility still remain unclear.

Finally, we state a corollary of Proposition 13 that shows that under our assumptions regarding the correlation operator, we have a unique solution to the nonparametric groups Lasso problem with probability tending to one (see proof in Appendix A.5):

Proposition 14 *Assume (**A4-5**). The problem (13) has a unique solution with probability tending to one.*

3.6 Estimation of Correlation Condition (16)

Condition (4) is simple to compute while the nonparametric condition (16) might be hard to check even if all densities are known (we provide however in Section 5 a specific example where we can compute in closed form all covariance operators). The following proposition shows that we can consistently estimate the quantities $\left\| \Sigma_{X_i X_i}^{1/2} C_{X_i X_\mathbf{J}} C_{X_\mathbf{J} X_\mathbf{J}}^{-1} \operatorname{Diag}(d_j / \|\mathbf{f}_j\|_{\mathcal{F}_j}) \mathbf{g}_\mathbf{J} \right\|_{\mathcal{F}_i}$ given an i.i.d. sample (see proof in Appendix C.4):

Proposition 15 *Assume (**A4-7**), and $\kappa_n \to 0$ and $\kappa_n n^{1/2} \to \infty$. Let*

$$\alpha = \Pi_n \left(\sum_{j \in \mathbf{J}} \Pi_n K_j \Pi_n + n\kappa_n I_n \right)^{-1} \Pi_n \bar{Y}$$

and $\hat{\eta}_j = \frac{1}{d_j} (\alpha^\top K_j \alpha)^{1/2}$. Then, for all $i \in \mathbf{J}^c$, the norm $\left\| \Sigma_{X_i X_i}^{1/2} C_{X_i X_\mathbf{J}} C_{X_\mathbf{J} X_\mathbf{J}}^{-1} \operatorname{Diag}(d_j / \|\mathbf{f}_j\|) \mathbf{g}_\mathbf{J} \right\|_{\mathcal{F}_i}$ is consistently estimated by:

$$\left\| (\Pi_n K_i \Pi_n)^{1/2} \left(\sum_{j \in \mathbf{J}} \Pi_n K_j \Pi_n + n\kappa_n I_n \right)^{-1} \left(\sum_{j \in \mathbf{J}} \frac{1}{\hat{\eta}_j} \Pi_n K_j \Pi_n \right) \alpha \right\|. \tag{20}$$

4. Adaptive Group Lasso and Multiple Kernel Learning

In previous sections, we have shown that specific necessary and sufficient conditions are needed for path consistency of the group Lasso and multiple kernel learning. The following procedures, adapted from the adaptive Lasso of Zou (2006), lead to two-step procedures that always achieve both consistency, with no condition such as Eq. (4) or Eq. (16). As before, results are a bit different when groups have finite sizes and groups may have infinite sizes.

4.1 Adaptive Group Lasso

The following theorem extends the similar theorem of Zou (2006), and shows that we can get both $O_p(n^{-1/2})$ consistency and correct pattern estimation:

Theorem 16 *Assume (**A1-3**) and $\gamma > 0$. We denote by $\hat{w}^{LS} = \hat{\Sigma}_{XX}^{-1} \hat{\Sigma}_{XY}$ the (unregularized) least-square estimate. We denote by $\hat{w}^A$ any minimizer of*

$$\frac{1}{2} \hat{\Sigma}_{YY} - \hat{\Sigma}_{YX} w + \frac{1}{2} w^\top \hat{\Sigma}_{XX} w + \frac{\mu_n}{2} \left(\sum_{j=1}^m \|\hat{w}_j^{LS}\|^{-\gamma} \|w_j\| \right)^2.$$

If $n^{-1/2} \gg \mu_n \gg n^{-1/2-\gamma/2}$, then $\hat{w}^A$ converges in probability to $\mathbf{w}$, $J(\hat{w}^A)$ converges in probability to $\mathbf{J}$, and $n^{1/2}(\hat{w}^A_{\mathbf{J}} - \mathbf{w}_{\mathbf{J}})$ tends in distribution to a normal distribution with mean zero and covariance matrix $\Sigma^{-1}_{X_{\mathbf{J}}X_{\mathbf{J}}}$.

This theorem, proved in Appendix D.1, shows that the adaptive group Lasso exhibit all important asymptotic properties, both in terms of errors and selected models. In the nonparametric case, we obtain a weaker result.

4.2 Adaptive Multiple Kernel Learning

We first begin with the consistency of the least-square estimate (see proof in Appendix D.2):

Proposition 17 *Assume (**A4-7**). The unique minimizer $\hat{f}^{LS}_{\kappa_n}$ of*

$$\frac{1}{2}\hat{\Sigma}_{YY} - \langle \hat{\Sigma}_{XY}, f\rangle_{\mathcal{F}} + \frac{1}{2}\langle f, \hat{\Sigma}_{XX} f\rangle_{\mathcal{F}} + \frac{\kappa_n}{2}\sum_{j=1}^m \|f_j\|^2_{\mathcal{F}_j},$$

converges in probability to f if $\kappa_n \to 0$ and $\kappa_n n^{1/2} \to 0$. Moreover, we have $\|\hat{f}^{LS}_{\kappa_n} - f\|_{\mathcal{F}} = O_p(\kappa_n^{1/2} + \kappa_n^{-1} n^{-1/2})$.

Since the least-square estimate is consistent and we have an upper bound on its convergence rate, we follow Zou (2006) and use it to defined adaptive weights d_j for which we get both sparsity and regular consistency without any conditions on the value of the correlation operators.

Theorem 18 *Assume (**A4-7**) and $\gamma > 1$. Let $\hat{f}^{LS}_{n^{-1/3}}$ be the least-square estimate with regularization parameter proportional to $n^{-1/3}$, as defined in Proposition 17. We denote by $\hat{f}^A$ any minimizer of*

$$\frac{1}{2}\hat{\Sigma}_{YY} - \langle \hat{\Sigma}_{XY}, f\rangle_{\mathcal{F}} + \frac{1}{2}\langle f, \hat{\Sigma}_{XX} f\rangle_{\mathcal{F}} + \frac{\mu_0 n^{-1/3}}{2}\left(\sum_{j=1}^m \|(\hat{f}^{LS}_{\kappa_n})_j\|^{-\gamma}_{\mathcal{F}_j}\|f_j\|_{\mathcal{F}_j}\right)^2.$$

Then $\hat{f}^A$ converges to $\mathbf{f}$ and $J(\hat{f}^A)$ converges to $\mathbf{J}$ in probability.

Theorem 18 allows to set up a specific vector of weights d. This provides a principled way to define data adaptive weights, that allows to solve (at least theoretically) the potential consistency problems of the usual MKL framework (see Section 5 for illustration on synthetic examples). Note that we have no result concerning the $O_p(n^{-1/2})$ consistency of our procedure (as we have for the finite dimensional case) and obtaining precise convergence rates is the subject of ongoing research.

The following proposition gives the expression for the solution of the least-square problem, necessary for the computation of adaptive weights in Theorem 18.

Proposition 19 *The solution of the least-square problem in Proposition 17 is given by*

$$\forall j \in \{1,\dots,m\},\ f^{LS}_j = \sum_{i=1}^n \alpha_i k_j(\cdot, x_{ij}) \text{ with } \alpha = \Pi_n\left(\sum_{j=1}^m \Pi_n K_j \Pi_n + n\kappa_n I_n\right)^{-1}\Pi_n \bar{Y},$$

with norms $\|\hat{F}^{LS}_j\|_{\mathcal{F}_j} = \left(\alpha^\top K_j \alpha\right)^{1/2}$, $j = 1,\dots,m$.

Other weighting schemes have been suggested, based on various heuristics. A notable one (which we use in simulations) is the normalization of kernel matrices by their trace (Lanckriet et al., 2004b), which leads to $d_j = (\mathrm{tr}\hat{\Sigma}_{X_jX_j})^{1/2} = (\frac{1}{n}\mathrm{tr}\Pi_n K_j \Pi_n)^{1/2}$. Bach et al. (2004b) have observed empirically that such normalization might lead to suboptimal solutions and consider weights d_j that grow with the empirical ranks of the kernel matrices. In this paper, we give theoretical arguments that indicate that weights which do depend on the data are more appropriate and work better (see Section 5 for examples).

5. Simulations

In this section, we illustrate the consistency results obtained in this paper with a few simple simulations on synthetic examples.

5.1 Groups of Finite Sizes

In the finite dimensional group case, we sampled $X \in \mathbb{R}^p$ from a normal distribution with zero mean vector and a covariance matrix of size $p = 8$ for $m = 4$ groups of size $p_j = 2$, $j = 1,\ldots,m$, generated as follows: (a) sample an $p \times p$ matrix G with independent standard normal distributions, (b) form $\Sigma_{XX} = GG^\top$, (c) for each $j \in \{1,\ldots,m\}$, rescale $X_j \in \mathbb{R}^2$ so that $\mathrm{tr}\Sigma_{X_jX_j} = 1$. We selected $\mathrm{Card}(\mathbf{J}) = 2$ groups at random and sampled non zero loading vectors as follows: (a) sample each loading from from independent standard normal distributions, (b) rescale those to unit norm, (c) rescale those by a scaling which is uniform at random between $\frac{1}{3}$ and 1. Finally, we chose a constant noise level of standard deviation σ equal to 0.2 times $(\mathbb{E}(\mathbf{w}^\top X)^2)^{1/2}$ and sampled Y from a conditional normal distribution with constant variance. The joint distribution on (X,Y) thus defined satisfies with probability one assumptions (**A1-3**).

For cases when the correlation conditions (4) and (5) were or were not satisfied, we consider two different weighting schemes, that is, different ways of setting the weights d_j of the block ℓ_1-norm: unit weights (which correspond to the unit trace weighting scheme) and adaptive weights as defined in Section 4.

In Figure 1, we plot the regularization paths corresponding to 200 i.i.d. samples, computed by the algorithm of Bach et al. (2004b). We only plot the values of the estimated variables $\hat{\eta}_j, j = 1,\ldots,m$ for the alternative formulation in Section 3.5, which are proportional to $\|\hat{w}_j\|$ and normalized so that $\sum_{j=1}^m \hat{\eta}_j = 1$. We compare them to the population values η_j: both in terms of values, and in terms of their sparsity pattern (η_j is zero for the weights which are equal to zero). Figure 1 illustrates several of our theoretical results: (a) the top row corresponds to a situation where the strict consistency condition is satisfied and thus we obtain model consistent estimates with also a good estimation of the loading vectors (in the figure, only the behavior of the norms of these loading vectors are represented); (b) the right column corresponds to the adaptive weighting schemes which also always achieve the two type of consistency; (c) in the middle and bottom rows, the consistency condition was not satisfied, and in the bottom row, the condition of Proposition 5, that ensures that we can get model consistent estimates without regular consistency, is met, while it is not in the middle row: as expected, in the bottom row, we get some model consistent estimates but with bad norm estimation.

In Figure 2, 3 and 4, we consider the three joint distributions used in Figure 1 and compute regularization paths for several number of samples (10 to 10^5) with 200 replications. This allows

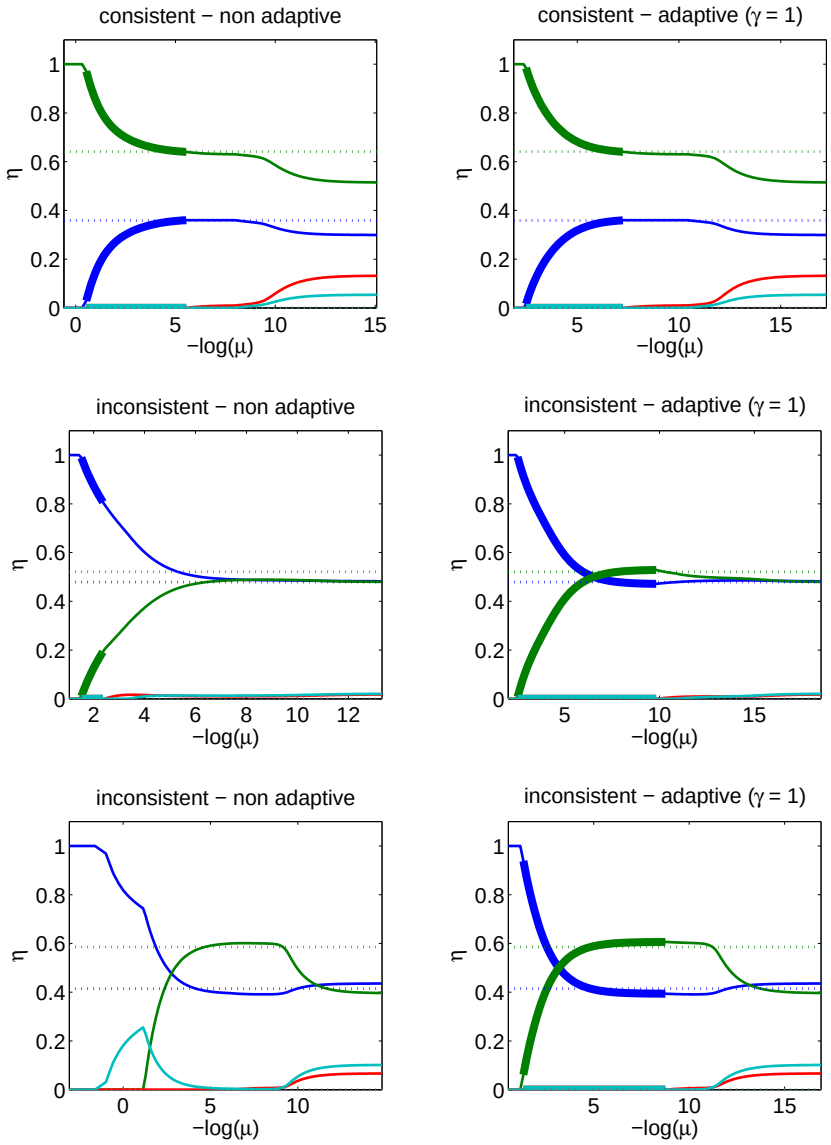


Figure 1: Regularization paths for the group Lasso for two weighting schemes (*left*: non adaptive, *right*: adaptive) and three different population densities (*top*: strict consistency condition satisfied, *middle*: weak condition not satisfied, no model consistent estimates, *bottom*: weak condition not satisfied, some model consistent estimates but without regular consistency). For each of the plots, plain curves correspond to values of estimated $\hat{\eta}_j$, dotted curves to population values η_j, and bold curves to model consistent estimates.

us to estimate both the probability of correct pattern estimation $\mathbb{P}(J(\hat{w}) = \mathbf{J})$ which is considered in Section 2.7, and the logarithm of the expected error $\log \mathbb{E}\|\hat{w} - \mathbf{w}\|^2$.

From Figure 2, it is worth noting (a) the regular spacing between the probability of correct pattern selection for several equally spaced (in log scale) numbers of samples, which corroborates

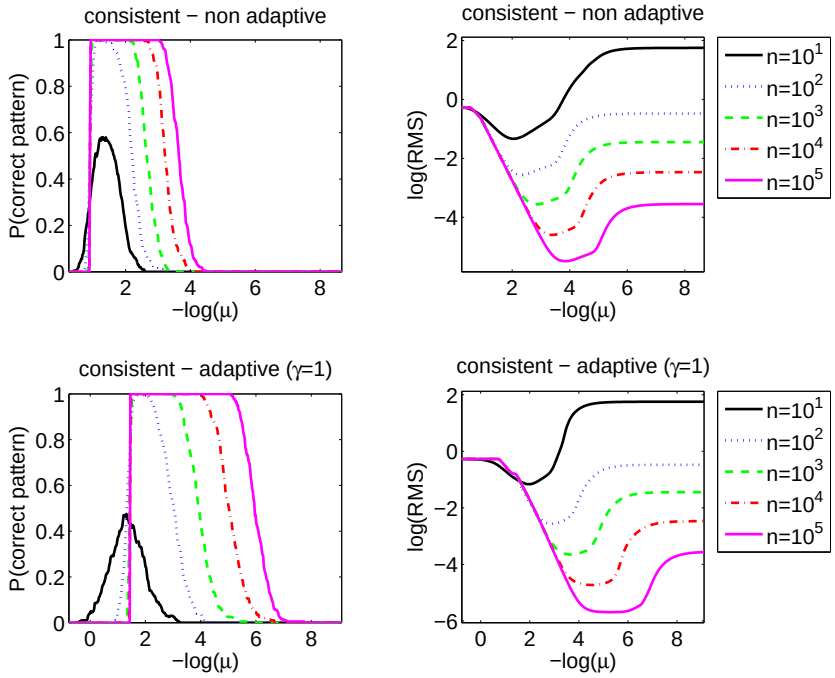


Figure 2: Synthetic example where consistency condition in Eq. (4) is satisfied (same example as the top of Figure 1: probability of correct pattern selection (*left*) and logarithm of the expected mean squared estimation error (*right*), for several number of samples as a function of the regularization parameter, for regular regularization (*top*), adaptive regularization with $\gamma = 1$ (*bottom*).

the asymptotic result in Section 2.7. Moreover, (b) in both rows, we get model consistent estimates with increasingly smaller norms as the number of samples grows. Finally, (c) the mean square errors are smaller for the adaptive weighting scheme.

From Figure 3, it is worth noting that (a) in the non adaptive case, we have two regimes for the probability of correct pattern selection: a regime corresponding to Proposition 6 where this probability can take values in $(0,1)$ for increasingly smaller regularization parameters (when n grows); and a regime corresponding to non vanishing limiting regularization parameters corresponding to Proposition 5: we have model consistency without regular consistency. Also, (b) the adaptive weighting scheme allows both consistencies. In Figure 4 however, the second regime (correct model estimates, inconsistent estimation of loadings) is not present.

In Figure 5, we sampled 10,000 different covariance matrices and loading vectors using the procedure described above. For each of these we computed the regularization paths from 1000 samples, and we classify each path into three categories: (1) existence of model consistent estimates with estimation error $\|\hat{w} - \mathbf{w}\|$ less than 10^{-1}, (2) existence of model consistent estimates but none with estimation error $\|\hat{w} - \mathbf{w}\|$ less than 10^{-1} and (3) non existence of model consistent estimates. In Figure 5 we plot the proportion of each of the three class as a function of the logarithm of $\max_{i \in \mathbf{J}^c} \frac{1}{d_i} \left\| \Sigma_{X_i X_\mathbf{J}} \Sigma_{X_\mathbf{J} X_\mathbf{J}}^{-1} \operatorname{Diag}(d_j / \|\mathbf{w}_j\|) \mathbf{w}_\mathbf{J} \right\|$. The position of the previous value with respect to 1 is indicative of the expected model consistency. When it is less than one, then we get with

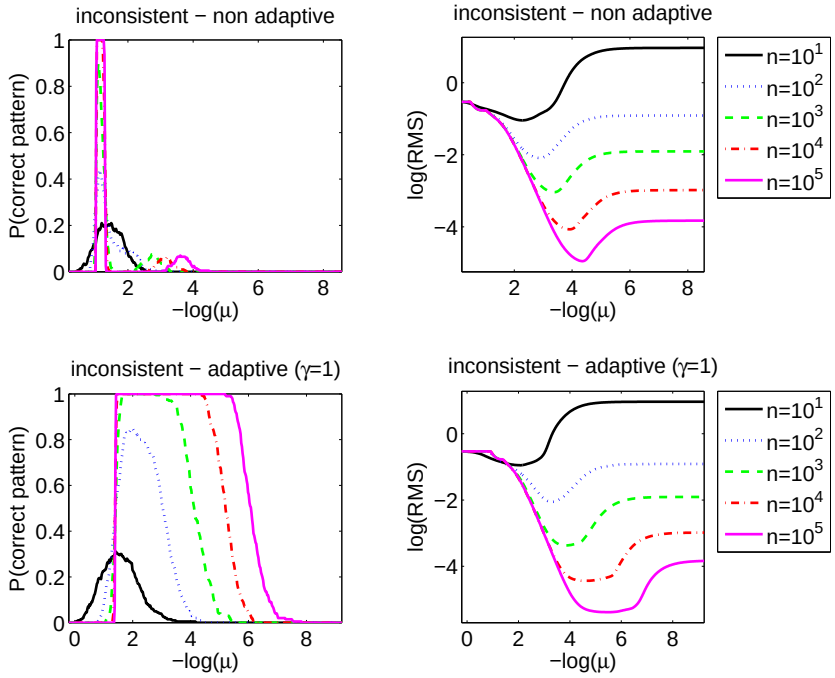


Figure 3: Synthetic example where consistency condition in Eq. (5) is not satisfied (same example as the middle of Figure 1: probability of correct pattern selection (*left*) and logarithm of the expected mean squared estimation error (*right*), for several number of samples as a function of the regularization parameter, for regular regularization (*top*), adaptive regularization with $\gamma = 1$ (*bottom*).

overwhelming probability model consistent estimates with good errors. As the condition gets larger than one, we get fewer such good estimates and more and more model inconsistent estimates.

5.2 Nonparametric Case

In the infinite dimensional group case, we sampled $X \in \mathbb{R}^m$ from a normal distribution with zero mean vector and a covariance matrix of size $m = 4$, generated as follows: (a) sample a $m \times m$ matrix G with independent standard normal distributions, (b) form $\Sigma_{XX} = GG^\top$, (c) for each $j \in \{1,\dots,m\}$, rescale $X_j \in \mathbb{R}$ so that $\Sigma_{X_jX_j} = 1$.

We use the same Gaussian kernel for each variable X_j, $k_j(x_j,x'_j) = e^{-(x_j-x'_j)^2}$, for $j \in \{1,\dots,m\}$. In this situation, as shown in Appendix E we can compute in closed form the eigenfunctions and eigenvalues of the marginal covariance operators; moreover, assumptions (**A4-7**) are satisfied. We then sample functions from random independent components on the first 10 eigenfunctions. Examples are given in Figure 6. Note that although we consider univariate variables, we still have infinite dimensional Hilbert spaces.

In Figure 7, we plot the regularization paths corresponding to 1000 i.i.d. samples, computed by the algorithm of Bach et al. (2004b). We only plot the values of the estimated variables $\hat{\eta}_j, j = 1,\dots,m$ for the alternative formulation in Section 2.9, which are proportional to $\|\hat{w}_j\|$ and normalized so that $\sum_{j=1}^m \hat{\eta}_j = 1$. We compare them to the population values η_j: both in terms of values,

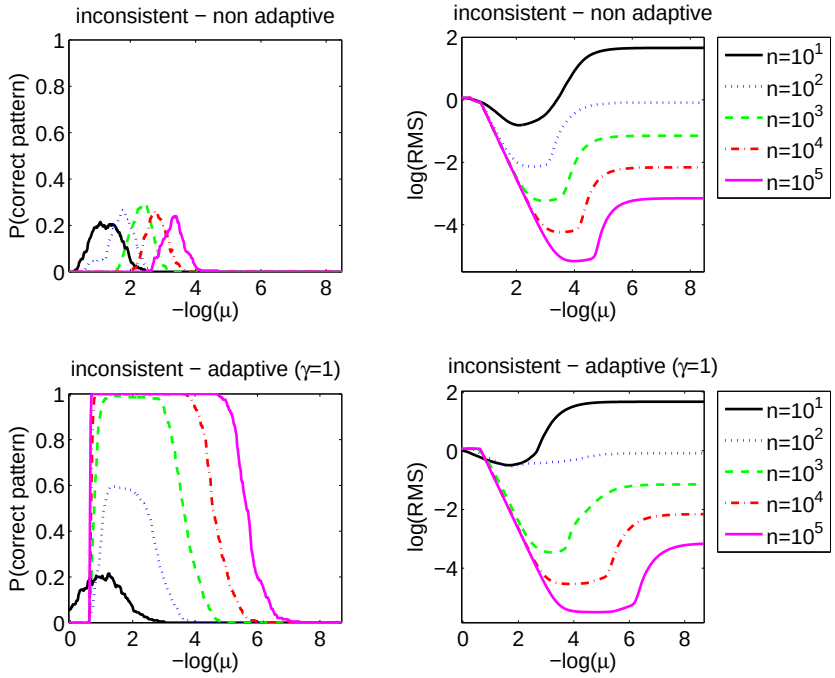


Figure 4: Synthetic example where consistency condition in Eq. (5) is not satisfied (same example as the bottom of Figure 1: probability of correct pattern selection (*left*) and logarithm of the expected mean squared estimation error (*right*), for several number of samples as a function of the regularization parameter, for regular regularization (*top*), adaptive regularization with $\gamma = 1$ (*bottom*).

and in terms of their sparsity pattern (η_j is zero for the weights which are equal to zero). Figure 7 illustrates several of our theoretical results: (a) the top row corresponds to a situation where the strict consistency condition is satisfied and thus we obtain model consistent estimates with also a good estimation of the loading vectors (in the figure, only the behavior of the norms of these loading vectors are represented); (b) in the bottom row, the consistency condition was not satisfied, and we do not get good model estimates. Finally, (b) the right column corresponds to the adaptive weighting schemes which also always achieve the two type of consistency. However, such schemes should be used with care, as there is one added free parameter (the regularization parameter κ of the least-square estimate used to define the weights): if chosen too large, all adaptive weights are equal, and thus there is no adaptation, while if chosen too small, the least-square estimate may overfit.

6. Conclusion

In this paper, we have extended some of the theoretical results of the Lasso to the group Lasso, for finite dimensional groups and infinite dimensional groups. In particular, under practical assumptions regarding the distributions the data are sampled from, we have provided necessary and sufficient conditions for model consistency of the group Lasso and its nonparametric version, multiple kernel learning.

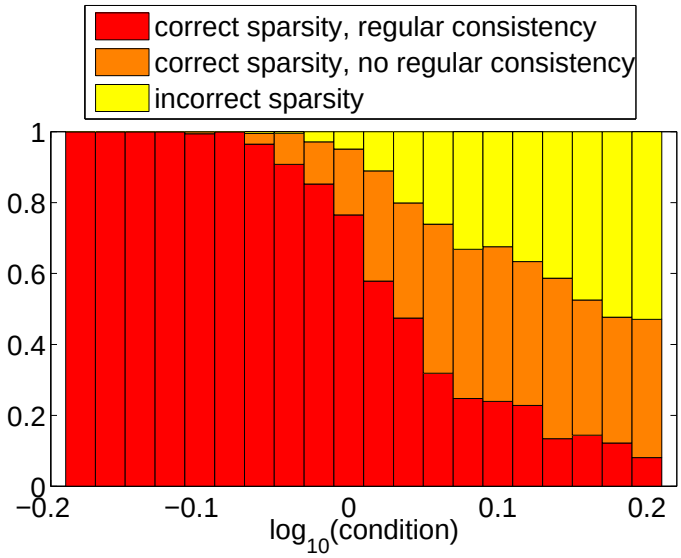


Figure 5: Consistency of estimation vs. consistency condition. See text for details.

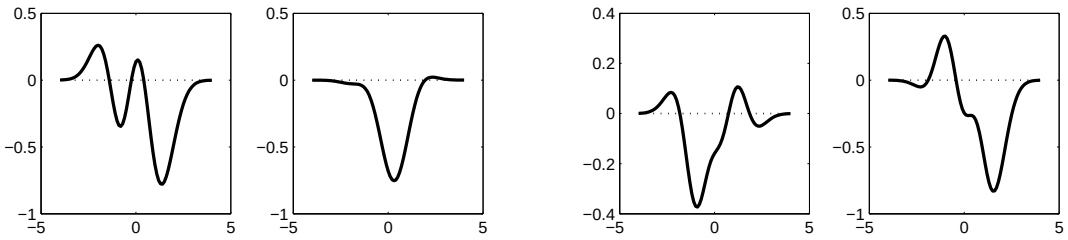


Figure 6: Functions to be estimated in the synthetic nonparametric group Lasso experiments (left: consistent case, right: inconsistent case).

The current work could be extended in several ways: first, a more detailed study of the limiting distributions of the group Lasso and adaptive group Lasso estimators could be carried and then extend the analysis of Zou (2006) or Juditsky and Nemirovski (2000) and Wu et al. (2007), in particular regarding convergence rates. Second, our results should extend to generalized linear models, such as logistic regression (Meier et al., 2006). Also, it is of interest to let the number m of groups or kernels to grow unbounded and extend the results of Zhao and Yu (2006) and Meinshausen and Yu (2006) to the group Lasso. Finally, similar analysis may be carried through for more general norms with different sparsity inducing properties (Bach, 2008b).

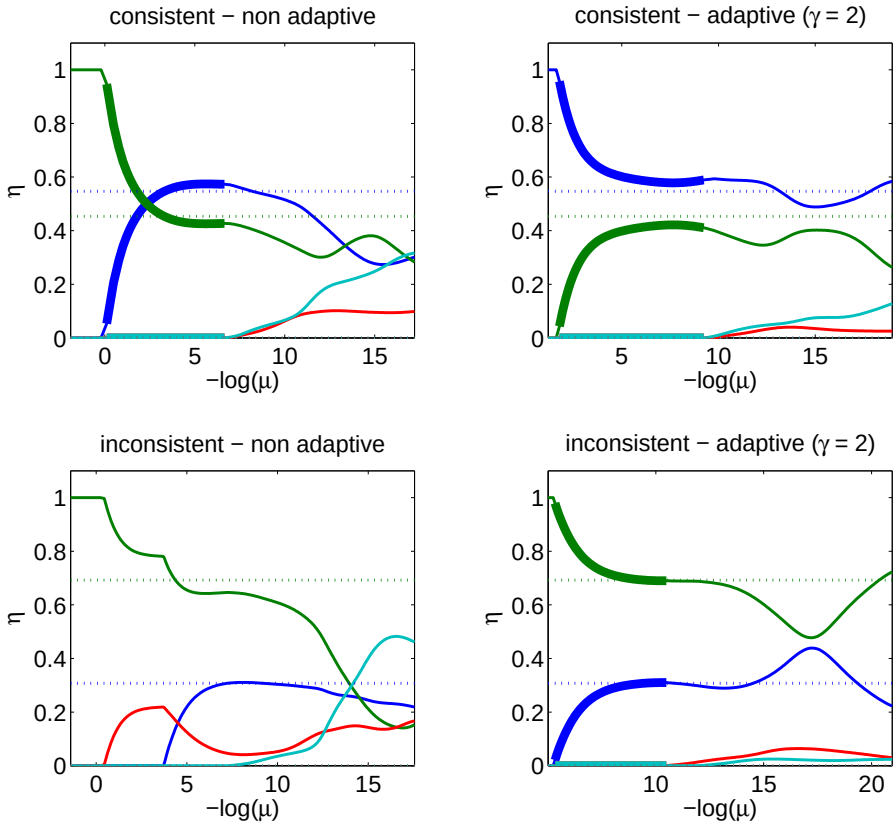


Figure 7: Regularization paths for the group Lasso for two weighting schemes (*left*: non adaptive, *right*: adaptive) and two different population densities (*top*: strict consistency condition satisfied, *bottom*: weak condition not satisfied. For each of the plots, plain curves correspond to values of estimated $\hat{\eta}_j$, dotted curves to population values η_j, and bold curves to model consistent estimates.

Acknowledgments

I would like to thank Zaïd Harchaoui for fruitful discussions related to this work. This work was supported by a French grant from the Agence Nationale de la Recherche (MGA Project).

Appendix A. Proof of Optimization Results

In this appendix, we give detailed proofs of the various propositions on optimality conditions and dual problems.

A.1 Proof of Proposition 1

We rewrite problem in Eq. (1), in the form

$$\min_{w\in\mathbb{R}^p,\ v\in\mathbb{R}^m} \frac{1}{2}\hat{\Sigma}_{YY} - \hat{\Sigma}_{YX}w + \frac{1}{2}w^\top\hat{\Sigma}_{XX}w + \lambda_n\sum_{j=1}^m d_j v_j,$$

with added constraints that $\forall j, \|w_j\| \leqslant v_j$. In order to deal with these constraints we use the tools from conic programming with the second-order cone, also known as the "ice cream" cone (Boyd and Vandenberghe, 2003). We consider the Lagrangian with dual variables $(\beta_j,\gamma_j)\in\mathbb{R}^{p_j}\times\mathbb{R}$ such that $\|\beta_j\|\leqslant\gamma_j$:

$$\mathcal{L}(w,v,\beta,\gamma) = \frac{1}{2}\hat{\Sigma}_{YY} - \hat{\Sigma}_{YX}w + \frac{1}{2}w^\top\hat{\Sigma}_{XX}w + \lambda_n d^\top v - \sum_{j=1}^m \begin{pmatrix} w_j \\ v_j\end{pmatrix}^\top \begin{pmatrix}\beta_j\\ \gamma_j\end{pmatrix}.$$

The derivatives with respect to primal variables are

$$\begin{aligned}\nabla_w\mathcal{L}(w,v,\beta,\gamma) &= \hat{\Sigma}_{XX}w - \hat{\Sigma}_{XY} - \beta,\\ \nabla_v\mathcal{L}(w,v,\beta,\gamma) &= \lambda_n d - \gamma.\end{aligned}$$

At optimality, primal and dual variables are completely characterized by w and β. Since the dual and the primal problems are strictly feasible, strong duality holds and the KKT conditions for reduced primal/dual variables (w,β) are

$$\begin{aligned}\forall j,\ \|\beta_j\| &\leqslant \lambda_n d_j && \text{(dual feasibility)},\\ \forall j,\ \beta_j &= \hat{\Sigma}_{X_jX}w - \hat{\Sigma}_{X_jY} && \text{(stationarity)},\\ \forall j,\ \beta_j^\top w_j + \|w_j\|\lambda_n d_j &= 0 && \text{(complementary slackness)}.\end{aligned}$$

Complementary slackness for the second order cone has special consequences: $w_j^\top\beta_j + \|w_j\|\lambda_n d_j = 0$ if and only if (Boyd and Vandenberghe, 2003; Lobo et al., 1998), either (a) $w_j = 0$, or (b) $w_j\neq 0$, $\|\beta_j\| = \lambda_n d_j$ and $\exists\eta_j>0$ such that $w_j = -\frac{\eta_j}{\lambda_n}\beta_j$ (anti-proportionality), which implies $\beta_j = -w_j\frac{\lambda_n d_j}{\|w_j\|}$ and $\eta_j = \|w_j\|/d_j$. This leads to the proposition.

A.2 Proof of Proposition 8

We follow the proof of Proposition 1 and of Bach et al. (2004a). We rewrite problem in Eq. (12), in the form

$$\min_{w\in\mathbb{R}^p,\ v\in\mathbb{R}^m,\ t\in\mathbb{R}} \frac{1}{2}\hat{\Sigma}_{YY} - \hat{\Sigma}_{YX}w + \frac{1}{2}w^\top\hat{\Sigma}_{XX}w + \frac{1}{2}\mu_n t^2,$$

with constraints that $\forall j, \|w_j\| \leqslant v_j$ and $d^\top v \leqslant t$. We consider the Lagrangian with dual variables $(\beta_j, \gamma_j) \in \mathbb{R}^{p_j} \times \mathbb{R}$ and $\delta \in \mathbb{R}_+$ such that $\|\beta_j\| \leqslant \gamma_j$, $j = 1, \ldots, m$:

$$\mathcal{L}(w, v, \beta, \gamma, \delta) = \frac{1}{2}\hat{\Sigma}_{YY} - \hat{\Sigma}_{YX} w + \frac{1}{2} w^\top \hat{\Sigma}_{XX} w + \frac{1}{2}\mu_n t^2 - \beta^\top w - \gamma^\top v + \delta(d^\top v - t).$$

The derivatives with respect to primal variables are

$$\begin{aligned}
\nabla_w \mathcal{L}(w, v, \beta, \gamma) &= \hat{\Sigma}_{XX} w - \hat{\Sigma}_{XY} - \beta, \\
\nabla_v \mathcal{L}(w, v, \beta, \gamma) &= \delta d - \gamma, \\
\nabla_t \mathcal{L}(w, v, \beta, \gamma) &= \mu_n t - \delta.
\end{aligned}$$

At optimality, primal and dual variables are completely characterized by w and β. Since the dual and the primal problems are strictly feasible, strong duality holds and the KKT conditions for reduced primal/dual variables (w, β) are

$$\begin{aligned}
\forall j, \beta_j &= \hat{\Sigma}_{X_j X} w - \hat{\Sigma}_{X_j Y} && \text{(stationarity - 1)}, \\
\forall j, \sum_{j=1}^m d_j \|w_j\| &= \frac{1}{\mu_n} \max_{i=1,\ldots,m} \frac{\|\beta_i\|}{d_i} && \text{(stationarity - 2)}, \\
\forall j, \left(\frac{\beta_j}{d_j}\right)^\top w_j + \|w_j\| \max_{i=1,\ldots,m} \frac{\|\beta_i\|}{d_i} &= 0 && \text{(complementary slackness)}.
\end{aligned} \tag{21}$$

Complementary slackness for the second order cone implies that:

$$\left(\frac{\beta_j}{d_j}\right)^\top w_j + \|w_j\| \max_{i=1,\ldots,m} \frac{\|\beta_i\|}{d_i} = 0,$$

if and only if, either (a) $w_j = 0$, or (b) $w_j \neq 0$ and $\frac{\|\beta_j\|}{d_j} = \max_{i=1,\ldots,m} \frac{\|\beta_i\|}{d_i}$, and $\exists \eta_j \geqslant 0$ such that $w_j = -\eta_j \beta_j / \mu_n$, which implies $\|w_j\| = \frac{\eta_j d_j}{\mu_n} \max_{i=1,\ldots,m} \frac{\|\beta_i\|}{d_i}$.

By writing $\eta_j = 0$ if $w_j = 0$ (i.e., in order to cover all cases), we have from Eq. (21) $\sum_{j=1}^m d_j \|w_j\| = \frac{1}{\mu_n} \max_{i=1,\ldots,m} \frac{\|\beta_i\|}{d_i}$, which implies $\sum_{j=1}^m d_j^2 \eta_j = 1$ and thus $\forall j,\ \eta_j = \frac{\|w_j\|/d_j}{\sum_i d_i \|w_i\|}$. This leads to $\forall j, \beta_j = -w_j \mu_n / \eta_j = -\frac{w_j}{\|w_j\|} \sum_{i=1}^n d_i \|w_i\|$. The proposition follows.

A.3 Proof of Proposition 10

By following the usual proof of the representer theorem (Wahba, 1990), we obtain that each optimal function f_j must be supported by the data points, that is, there exists $\alpha = (\alpha_1, \ldots, \alpha_m) \in \mathbb{R}^{n \times m}$ such that for all $j = 1, \ldots, m$, $f_j = \sum_{i=1}^n \alpha_{ij} k_j(\cdot, x_{ij})$. When using this representation back into Eq. (13), we obtain an optimization problem that only depends on $\phi_j = G_j^\top \alpha_j$ for $j = 1, \ldots, m$ where G_j denotes any square root of the kernel matrix K_j, that is, $K_j = G_j G_j^\top$. This problem is exactly the finite dimensional problem in Eq. (12), where X_j is replaced by G_j and w_j by ϕ_j. Thus Proposition 8 applies and we can easily derive the current proposition by expressing all terms through the functions f_j. Note that in this proposition, we do not show that the α_j, $j = 1, \ldots, m$, are all proportional to the same vector, as is done in Appendix A.4.

A.4 Proof of Proposition 13

We prove the proposition in the linear case. Going to the general case, can be done in the same way as done in Appendix A.3. We denote by $\bar{X}$ the covariate matrix in $\mathbb{R}^{n\times p}$; we simply need to add a new variable $u = \bar{X}w + b1_n$ and to "dualize" the added equality constraint. That is, we rewrite problem in Eq. (12), in the form

$$\min_{w\in\mathbb{R}^p,\ b\in\mathbb{R},\ v\in\mathbb{R}^m,\ t\in\mathbb{R},\ u\in\mathbb{R}^n} \frac{1}{2n}\|\bar{Y}-u\|^2 + \frac{1}{2}\mu_n t^2,$$

with constraints that $\forall j, \|w_j\| \leqslant v_j$, $d^\top v \leqslant t$ and $\bar{X}w + b1_n = u$. We consider the Lagrangian with dual variables $(\beta_j,\gamma_j) \in \mathbb{R}^{p_j}\times\mathbb{R}$ and $\delta\in\mathbb{R}_+$ such that $\|\beta_j\| \leqslant \gamma_j$, and $\alpha\in\mathbb{R}^n$:

$$\mathcal{L}(w,b,v,u,\beta,\gamma,\alpha,\delta) = \frac{1}{2n}\|\bar{Y}-u\|^2 + \mu_n\alpha^\top(u-\bar{X}w) + \frac{1}{2}\mu_n t^2 - \sum_{j=1}^m \Big\{\beta_j^\top w_j + \gamma_j v_j\Big\} + \delta(d^\top v - t).$$

The derivatives with respect to primal variables are

$$\begin{aligned}
\nabla_w\mathcal{L}(w,v,u,\beta,\gamma,\alpha) &= -\mu_n\bar{X}^\top\alpha - \beta,\\
\nabla_v\mathcal{L}(w,v,u,\beta,\gamma,\alpha) &= \delta d - \gamma,\\
\nabla_t\mathcal{L}(w,v,u,\beta,\gamma,\alpha) &= \mu_n t - \delta,\\
\nabla_u\mathcal{L}(w,v,u,\beta,\gamma,\alpha) &= \frac{1}{n}(u - \bar{Y} + \mu_n n\alpha),\\
\nabla_b\mathcal{L}(w,v,u,\beta,\gamma,\alpha) &= \mu_n\alpha^\top 1_n.
\end{aligned}$$

Equating them to zero, we get the dual problem in Eq. (18). Since the dual and the primal problems are strictly feasible, strong duality holds and the KKT conditions for reduced primal/dual variables (w,α) are

$$\begin{aligned}
\forall j, \bar{X}w - \bar{Y} + \mu_n n\alpha = 0 &\qquad \text{(stationarity - 1)},\\
\forall j, \sum_{j=1}^m d_j\|w_j\| = \max_{i=1,\dots,m}\frac{(\alpha^\top K_i\alpha)^{1/2}}{d_i} &\qquad \text{(stationarity - 2)},\\
\alpha^\top 1_n = 0 &\qquad \text{(stationarity - 3)},\\
\forall j, \left(\frac{-\bar{X}_j^\top\alpha}{d_j}\right)^\top w_j + \|w_j\|\max_{i=1,\dots,m}\frac{(\alpha^\top K_i\alpha)^{1/2}}{d_i} = 0 &\qquad \text{(complementary slackness)}.
\end{aligned}$$

Complementary slackness for the second order cone goes leads to:

$$\left(\frac{-\bar{X}_j^\top\alpha}{d_j}\right)^\top w_j + \|w_j\|\max_{i=1,\dots,m}\frac{(\alpha^\top K_i\alpha)^{1/2}}{d_i} = 0,$$

if and only if, either (a) $w_j = 0$, or (b) $w_j \neq 0$ and $\frac{(\alpha^\top K_j\alpha)^{1/2}}{d_j} = \max_{i=1,\dots,m}\frac{(\alpha^\top K_i\alpha)^{1/2}}{d_i}$, and $\exists\eta_j \geqslant 0$ such that $w_j = -\eta_j\left(-\bar{X}_j^\top\alpha\right)$, which implies $\|w_j\| = \eta_j d_j\max_{i=1,\dots,m}\frac{(\alpha^\top K_i\alpha)^{1/2}}{d_i}$.

By writing $\eta_j = 0$ if $w_j = 0$ (to cover all cases), we have from Eq. (22), $\sum_{j=1}^m d_j\|w_j\| = \max_{i=1,\dots,m}\frac{(\alpha^\top K_i\alpha)^{1/2}}{d_i}$, which implies $\sum_{j=1}^m d_j^2\eta_j = 1$. The proposition follows from the fact that at optimality, $\forall j$, $w_j = \eta_j\bar{X}_j^\top\alpha$.

A.5 Proof of Proposition 14

What makes this proposition non obvious is the fact that the covariance operator Σ_{XX} is not invertible in general. From Proposition 13, we know that each f_j must be of the form $f_j = \eta_j \sum_{i=1}^n \alpha_i k_j(x_{ij}, \cdot)$, where α is *uniquely* defined. Moreover, η is such that

$$\left(\sum_{j=1}^m \eta_j K_j + n\mu_n I_n\right) \alpha = \bar{Y}$$

and such that if $\frac{\alpha^\top K_j \alpha}{d_j^2} < A$, then $\eta_j = 0$ (where $A = \max_{i=1,\ldots,m} \frac{\alpha^\top K_i \alpha}{d_i^2}$). Thus, if the solution is not unique, there exists two vectors $\eta \neq \zeta$ such that η and ζ have zero components on indices j such that $\alpha^\top K_j \alpha < A d_j^2$ (we denote by J the active set and thus J^c this set of indices), and $\sum_{j=1}^m (\zeta_j - \eta_j) K_j \alpha = 0$. This implies that the vectors $\Pi_n K_j \alpha = \Pi_n K_j \Pi_n \alpha$, $j \in J$ are linearly dependent. Those vectors are exactly the centered vector of values of the functions $g_j = \sum_{i=1}^n \alpha_i k_j(x_{ij}, \cdot)$ at the observed data points. Thus, non unicity implies that the empirical covariance matrix of the random variables $g_j(X_j)$, $j \in J$, is non invertible. Moreover, we have $\|g_j\|_{\mathcal{F}_j}^2 = \alpha^\top K_j \alpha = d_j^2 A > 0$ and the empirical marginal variance of $g_j(X_j)$ is equal to $\alpha^\top K_j^2 \alpha > 0$ (otherwise $\|g_j\|_{\mathcal{F}_j}^2 = 0$). By normalizing by the (non vanishing) empirical standard deviations, we thus obtain functions such that the empirical covariance matrix is singular, but the marginal empirical variance are equal to one. Because the empirical covariance operator is a consistent estimator of Σ_{XX} and C_{XX} is invertible, we get a contradiction, which proves the unicity of solutions.

Appendix B. Detailed Proofs for the Group Lasso

In this appendix, detailed proofs of the consistency results for the finite dimensional case (Theorems 2 and 3) are presented. Some of the results presented in this appendix are corollaries of the more general results in Appendix C, but their proofs in the finite dimensional case are much simpler.

B.1 Proof of Theorem 2

We begin with a lemma, which states that if we restrict ourselves to the covariates which we are after (i.e., indexed by $\mathbf{J}$), we get a consistent estimate as soon as λ_n tends to zero:

Lemma 20 *Assume (**A1-3**). Let $\tilde{w}_\mathbf{J}$ any minimizer of*

$$\frac{1}{2n}\|\bar{Y} - \bar{X}_\mathbf{J} w_\mathbf{J}\|^2 + \lambda_n \sum_{j \in \mathbf{J}} d_j \|w_j\| = \frac{1}{2}\hat{\Sigma}_{YY} - \hat{\Sigma}_{YX_\mathbf{J}} w_\mathbf{J} + \frac{1}{2} w_\mathbf{J}^\top \hat{\Sigma}_{X_\mathbf{J} X_\mathbf{J}} w_\mathbf{J} + \lambda_n \sum_{j \in \mathbf{J}} d_j \|w_j\|.$$

If $\lambda_n \to 0$, then $\tilde{w}_\mathbf{J}$ converges to $\mathbf{w}_\mathbf{J}$ in probability.

Proof If λ_n tends to zero, then the cost function defining $\tilde{w}_\mathbf{J}$ converges to $F(w_\mathbf{J}) = \frac{1}{2}\Sigma_{YY} - \Sigma_{YX_\mathbf{J}} w_\mathbf{J} + \frac{1}{2} w_\mathbf{J}^\top \Sigma_{X_\mathbf{J} X_\mathbf{J}} w_\mathbf{J}$ whose unique (because $\Sigma_{X_\mathbf{J} X_\mathbf{J}}$ is positive definite) global minimum is $\mathbf{w}_\mathbf{J}$ (true generating value). The convergence of $\tilde{w}_\mathbf{J}$ is thus a simple consequence of standard results in M-estimation (Van der Vaart, 1998; Fu and Knight, 2000). ■

We now prove Theorem 2. Let $\tilde{w}_\mathbf{J}$ be defined as in Lemma 20. We extend it by zeros on $\mathbf{J}^c$. We already know from Lemma 20 that we have consistency in squared norm. Since with probability

tending to one, the problem has a unique solution (because Σ_{XX} is invertible), we now need to prove that the probability that $\tilde{w}$ is optimal for problem in Eq. (1) is tending to one.

By definition of $\tilde{w}_{\mathbf{J}}$, the optimality condition (3) is satisfied. We now need to verify optimality condition (2), that is, that variables in $\mathbf{J}^c$ may actually be left out. Denoting $\varepsilon = Y - \mathbf{w}^\top X - \mathbf{b}$, we have:

$$\hat{\Sigma}_{XY} = \hat{\Sigma}_{XX}\mathbf{w} + \hat{\Sigma}_{X\varepsilon} = \Big(\Sigma_{XX} + O_p(n^{-1/2})\Big)\mathbf{w} + O_p(n^{-1/2}) = \Sigma_{XX_{\mathbf{J}}}\mathbf{w}_{\mathbf{J}} + O_p(n^{-1/2}),$$

because of classical results on convergence of empirical covariances to covariances (Van der Vaart, 1998), which are applicable because we have the fourth order moment condition (**A1**). We thus have:

$$\hat{\Sigma}_{XY} - \hat{\Sigma}_{XX_{\mathbf{J}}}\tilde{w}_{\mathbf{J}} = \Sigma_{XX_{\mathbf{J}}}(\mathbf{w}_{\mathbf{J}} - \tilde{w}_{\mathbf{J}}) + O_p(n^{-1/2}). \tag{22}$$

From the optimality condition $\hat{\Sigma}_{X_{\mathbf{J}}Y} - \hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}\tilde{w}_{\mathbf{J}} = \lambda_n \operatorname{Diag}(d_j/\|\tilde{w}_j\|)\tilde{w}_{\mathbf{J}}$ defining $\tilde{w}_{\mathbf{J}}$ and Eq. (22), we obtain:

$$\tilde{w}_{\mathbf{J}} - \mathbf{w}_{\mathbf{J}} = -\lambda_n \Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}^{-1} \operatorname{Diag}(d_j/\|\tilde{w}_j\|)\tilde{w}_{\mathbf{J}} + O_p(n^{-1/2}). \tag{23}$$

Therefore,

$$\begin{aligned}
\hat{\Sigma}_{X_{\mathbf{J}^c}Y} - \hat{\Sigma}_{X_{\mathbf{J}^c}X_{\mathbf{J}}}\tilde{w}_{\mathbf{J}} &= \Sigma_{X_{\mathbf{J}^c}X_{\mathbf{J}}}(\mathbf{w}_{\mathbf{J}} - \tilde{w}_{\mathbf{J}}) + O_p(n^{-1/2}) \text{ by Eq. (22)}, \\
&= \lambda_n \Sigma_{X_{\mathbf{J}^c}X_{\mathbf{J}}}\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}^{-1} \operatorname{Diag}(d_j/\|\tilde{w}_j\|)\tilde{w}_{\mathbf{J}} + O_p(n^{-1/2}) \text{ by Eq. (23)}.
\end{aligned}$$

Since $\tilde{w}$ is consistent, and $\lambda_n n^{1/2} \to +\infty$, then for each $i \in J^c$,

$$\frac{1}{d_i\lambda_n}\left(\hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_{\mathbf{J}}}\tilde{w}_{\mathbf{J}}\right)$$

converges in probability to $\frac{1}{d_i}\Sigma_{X_iX_{\mathbf{J}}}\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}^{-1}\operatorname{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_{\mathbf{J}}$ which is of norm *strictly* smaller than one because condition (4) is satisfied. Thus the probability that $\tilde{w}$ is indeed optimal, which is equal to

$$\mathbb{P}\left\{\forall i \in \mathbf{J}^c, \frac{1}{d_i\lambda_n}\left\|\hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_{\mathbf{J}}}\tilde{w}_{\mathbf{J}}\right\| \leqslant 1\right\} \geqslant \prod_{i\in\mathbf{J}^c}\mathbb{P}\left\{\frac{1}{d_i\lambda_n}\left\|\hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_{\mathbf{J}}}\tilde{w}_{\mathbf{J}}\right\| \leqslant 1\right\},$$

is tending to 1, which implies the theorem.

B.2 Proof of Theorem 3

We prove the theorem by contradiction, by assuming that there exists $i \in \mathbf{J}^c$ such that

$$\frac{1}{d_i}\left\|\Sigma_{X_iX_{\mathbf{J}}}\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}^{-1}\operatorname{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_{\mathbf{J}}\right\| > 1.$$

Since with probability tending to one $J(\hat{w}) = \mathbf{J}$, with probability tending to one, we have from optimality condition (3):

$$\hat{w}_{\mathbf{J}} = \hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}^{-1}\left(\hat{\Sigma}_{X_{\mathbf{J}}Y} - \lambda_n \operatorname{Diag}(d_j/\|\hat{w}_j\|)\hat{w}_{\mathbf{J}}\right),$$

and thus

$$\begin{aligned}
\hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_{\mathbf{J}}}\hat{w}_{\mathbf{J}} &= (\hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_{\mathbf{J}}}\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}^{-1}\hat{\Sigma}_{X_{\mathbf{J}}Y}) + \lambda_n\hat{\Sigma}_{X_iX_{\mathbf{J}}}\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}^{-1}\operatorname{Diag}(d_j/\|\hat{w}_j\|)\hat{w}_{\mathbf{J}} \\
&= A_n + B_n.
\end{aligned}$$

The second term B_n in the last expression (divided by λ_n) converges to

$$v = \Sigma_{X_i X_{\mathbf{J}}} \Sigma_{X_{\mathbf{J}} X_{\mathbf{J}}}^{-1} \operatorname{Diag}(d_j / \|\mathbf{w}_j\|) \mathbf{w}_{\mathbf{J}} \in \mathbb{R}^{p_i},$$

because $\hat{w}$ is assumed to converge in probability to $\mathbf{w}$ and empirical covariance matrices converge to population covariance matrices. By assumption $\|v\| > d_i$, which implies that the probability $\mathbb{P}\left\{ \left(\frac{v}{\|v\|} \right)^\top (B_n/\lambda_n) \geqslant (d_i + \|v\|)/2) \right\}$ converges to one.

The first term is equal to (with $\varepsilon_k = y_k - \mathbf{w}^\top x_k - \mathbf{b}_k$ and $\bar{\varepsilon} = \frac{1}{n}\sum_{k=1}^n \varepsilon_k$):

$$\begin{aligned}
A_n &= \hat{\Sigma}_{X_i Y} - \hat{\Sigma}_{X_i X_{\mathbf{J}}} \hat{\Sigma}_{X_{\mathbf{J}} X_{\mathbf{J}}}^{-1} \hat{\Sigma}_{X_{\mathbf{J}} Y} \\
&= \hat{\Sigma}_{X_i X_{\mathbf{J}}} \mathbf{w}_{\mathbf{J}} - \hat{\Sigma}_{X_i X_{\mathbf{J}}} \hat{\Sigma}_{X_{\mathbf{J}} X_{\mathbf{J}}}^{-1} \hat{\Sigma}_{X_{\mathbf{J}} X_{\mathbf{J}}} \mathbf{w}_{\mathbf{J}} + \hat{\Sigma}_{X_i \varepsilon} - \hat{\Sigma}_{X_i X_{\mathbf{J}}} \hat{\Sigma}_{X_{\mathbf{J}} X_{\mathbf{J}}}^{-1} \hat{\Sigma}_{X_{\mathbf{J}} \varepsilon} \\
&= \hat{\Sigma}_{X_i \varepsilon} - \hat{\Sigma}_{X_i X_{\mathbf{J}}} \hat{\Sigma}_{X_{\mathbf{J}} X_{\mathbf{J}}}^{-1} \hat{\Sigma}_{X_{\mathbf{J}} \varepsilon} \\
&= \hat{\Sigma}_{X_i \varepsilon} - \Sigma_{X_i X_{\mathbf{J}}} \Sigma_{X_{\mathbf{J}} X_{\mathbf{J}}}^{-1} \hat{\Sigma}_{X_{\mathbf{J}} \varepsilon} + o_p(n^{-1/2}) \\
&= \frac{1}{n} \sum_{k=1}^n (\varepsilon_k - \bar{\varepsilon}) \left(x_{ki} - \Sigma_{X_i X_{\mathbf{J}}} \Sigma_{X_{\mathbf{J}} X_{\mathbf{J}}}^{-1} x_{k\mathbf{J}} \right) + o_p(n^{-1/2}) = C_n + o_p(n^{-1/2}).
\end{aligned}$$

The random variable C_n is a is a U-statistic with square integrable kernel obtained from i.i.d. random vectors; it is thus asymptotically normal (Van der Vaart, 1998). We thus simply need to compute the mean and the variance of C_n. We have $\mathbb{E}C_n = 0$ because $\mathbb{E}(X\varepsilon) = \Sigma_{X\varepsilon} = 0$. We denote $D_k = x_{ki} - \Sigma_{X_i X_{\mathbf{J}}} \Sigma_{X_{\mathbf{J}} X_{\mathbf{J}}}^{-1} x_{k\mathbf{J}} - \frac{1}{n}\sum_{k=1}^n x_{ki} - \Sigma_{X_i X_{\mathbf{J}}} \Sigma_{X_{\mathbf{J}} X_{\mathbf{J}}}^{-1} x_{k\mathbf{J}}$. We have:

$$\begin{aligned}
\operatorname{var}(C_n) &= \mathbb{E}C_n^2 = \mathbb{E}(\mathbb{E}(C_n^2 | \bar{X})) \\
&= \mathbb{E}\left[\frac{1}{n^2} \sum_{k=1}^n E(\varepsilon_k^2 | \bar{X}) D_k D_k^\top \right] \\
&\succcurlyeq \mathbb{E}\left[\frac{1}{n^2} \sum_{k=1}^n \sigma_{\min}^2 D_k D_k^\top \right] \\
&= \frac{1}{n} \sigma_{\min}^2 \mathbb{E}\left(\hat{\Sigma}_{X_i X_i} - \Sigma_{X_i X_{\mathbf{J}}} \Sigma_{X_{\mathbf{J}} X_{\mathbf{J}}}^{-1} \hat{\Sigma}_{X_{\mathbf{J}} X_i} \right) \\
&= \frac{n-1}{n^2} \sigma_{\min}^2 \left(\Sigma_{X_i X_i} - \Sigma_{X_i X_{\mathbf{J}}} \Sigma_{X_{\mathbf{J}} X_{\mathbf{J}}}^{-1} \Sigma_{X_{\mathbf{J}} X_i} \right),
\end{aligned}$$

where $M \succcurlyeq N$ denotes the partial order between symmetric matrices (i.e., equivalent to $M - N$ positive semidefinite).

Thus $n^{1/2} C_n$ is asymptotically normal with mean 0 and covariance matrix larger than

$$\sigma_{\min}^2 \Sigma_{X_i | X_{\mathbf{J}}} = \sigma_{\min}^2 \times (\Sigma_{X_i X_i} - \Sigma_{X_i X_{\mathbf{J}}} \Sigma_{X_{\mathbf{J}} X_{\mathbf{J}}}^{-1} \Sigma_{X_{\mathbf{J}} X_i})$$

which is positive definite (because this is the conditional covariance of X_i given $X_{\mathbf{J}}$ and Σ_{XX} is assumed invertible). Therefore $\mathbb{P}(n^{1/2} v^\top A_n > 0)$ converges to a constant $a \in (0,1)$, which implies that $\mathbb{P}\left\{ \frac{v}{\|v\|}^\top (A_n + B_n)/\lambda_n \geqslant (d_i + \|v\|)/2 \right\}$ is asymptotically bounded below by a. Thus, since $\|(A_n + B_n)/\lambda_n\| \geqslant \frac{v}{\|v\|}^\top (A_n + B_n)/\lambda_n \geqslant (d_i + \|v\|)/2 > d_i$ implies that $\hat{w}$ is not optimal, we get a contradiction, which concludes the proof.

B.3 Proof of Theorem 4

We first prove the following refinement of Lemma 20:

Lemma 21 *Assume (***A1-3***). Let $\tilde{w}_{\mathbf{J}}$ any minimizer of*

$$\frac{1}{2n}\|\bar{Y}-\bar{X}_{\mathbf{J}}w_{\mathbf{J}}\|^2+\lambda_n\sum_{j\in\mathbf{J}}d_j\|w_j\| = \frac{1}{2}\hat{\Sigma}_{YY}-\hat{\Sigma}_{YX_{\mathbf{J}}}w_{\mathbf{J}}+\frac{1}{2}w_{\mathbf{J}}^\top\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}w_{\mathbf{J}}+\lambda_n\sum_{j\in\mathbf{J}}d_j\|w_j\|.$$

If $\lambda_n\to 0$ and $\lambda_n n^{1/2}\to\infty$, then $\frac{1}{\lambda_n}(\tilde{w}_{\mathbf{J}}-\mathbf{w}_{\mathbf{J}})$ converges in probability to

$$\Delta = -\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}^{-1}\operatorname{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_{\mathbf{J}}.$$

Proof We follow Fu and Knight (2000) and write $\tilde{w}_{\mathbf{J}}=\mathbf{w}_{\mathbf{J}}+\lambda_n\tilde{\Delta}$. The vector $\tilde{\Delta}$ is the minimizer of the following function:

$$\begin{aligned}
F(\Delta) &= -\hat{\Sigma}_{YX_{\mathbf{J}}}(\mathbf{w}_{\mathbf{J}}+\lambda_n\Delta)+\frac{1}{2}(\mathbf{w}_{\mathbf{J}}+\lambda_n\Delta)^\top\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}(\mathbf{w}_{\mathbf{J}}+\lambda_n\Delta)+\lambda_n\sum_{j\in\mathbf{J}}d_j\|\mathbf{w}_j+\lambda_n\Delta_j\| \\
&= -\lambda_n\hat{\Sigma}_{YX_{\mathbf{J}}}\Delta+\frac{\lambda_n^2}{2}\Delta^\top\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}\Delta+\lambda_n\mathbf{w}_{\mathbf{J}}^\top\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}\Delta \\
&\qquad\qquad +\lambda_n\sum_{j\in\mathbf{J}}d_j\left(\|\mathbf{w}_j+\lambda_n\Delta_j\|-\|\mathbf{w}_j\|\right)+\text{ cst} \\
&= -\lambda_n\hat{\Sigma}_{\varepsilon X_{\mathbf{J}}}\Delta+\frac{\lambda_n^2}{2}\Delta^\top\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}\Delta+\lambda_n\sum_{j\in\mathbf{J}}d_j\left(\|\mathbf{w}_j+\lambda_n\Delta_j\|-\|\mathbf{w}_j\|\right)+\text{ cst},
\end{aligned}$$

by using $\hat{\Sigma}_{YX_{\mathbf{J}}}=\mathbf{w}_{\mathbf{J}}^\top\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}+\hat{\Sigma}_{\varepsilon X_{\mathbf{J}}}$. The first term is $O_p(n^{-1/2}\lambda_n)=o_p(\lambda_n^2)$, while the last ones are equal to $\|\mathbf{w}_j+\lambda_n\Delta_j\|-\|\mathbf{w}_j\|=\lambda_n\left(\frac{\mathbf{w}_j}{\|\mathbf{w}_j\|}\right)^\top\Delta_j+o_p(\lambda_n)$. Thus,

$$F(\Delta)/\lambda_n^2=\frac{1}{2}\Delta^\top\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}\Delta+\sum_{j\in\mathbf{J}}\frac{d_j\mathbf{w}_j}{\|\mathbf{w}_j\|}^\top\Delta_j+o_p(1).$$

By Lemma 20, $\hat{w}_{\mathbf{J}}$ is $O_p(1)$ and the limiting function has an unique minimum; standard results in M-estimation (Van der Vaart, 1998) shows that $\tilde{\Delta}$ converges in probability to the minimum of the last expression which is exactly $\Delta=-\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}^{-1}\operatorname{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_{\mathbf{J}}$. ■

We now turn to the proof of Theorem 4. We follow the proof of Theorem 2. Given $\tilde{w}$ defined through Lemma 20 and 21, we need to satisfy optimality condition (2) for all $i\in\mathbf{J}^c$, with probability tending to one. For all those i such that $\frac{1}{d_i}\left\|\Sigma_{X_iX_{\mathbf{J}}}\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}^{-1}\operatorname{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_{\mathbf{J}}\right\|<1$, then we know from Appendix B.1, that the optimality condition is indeed satisfied with probability tending to one. We now focus on those i such that $\frac{1}{d_i}\left\|\Sigma_{X_iX_{\mathbf{J}}}\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}^{-1}\operatorname{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_{\mathbf{J}}\right\|=1$, and for which we have the condition in Eq. (6). From Eq. (23) and the few arguments that follow, we get that for all $i\in\mathbf{J}^c$,

$$\hat{\Sigma}_{X_iY}-\hat{\Sigma}_{X_iX_{\mathbf{J}}}\tilde{w}_{\mathbf{J}}=\lambda_n\Sigma_{X_iX_{\mathbf{J}}}\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}^{-1}\operatorname{Diag}(d_j/\|\tilde{w}_j\|)\tilde{w}_{\mathbf{J}}+O_p(n^{-1/2}) \tag{24}$$

Moreover, we have from Lemma 21 and standard differential calculus, that is, the gradient and the Hessian of the function $v \in \mathbb{R}^q \mapsto \|v\| \in \mathbb{R}$ are $v/\|v\|$ and $\frac{1}{\|v\|}\left(I_q - \frac{vv^\top}{v^\top v}\right)$:

$$\frac{\tilde{w}_j}{\|\tilde{w}_j\|} = \frac{\mathbf{w}_j}{\|\mathbf{w}_j\|} + \frac{\lambda_n}{\|\mathbf{w}_j\|}\left(I_{p_j} - \frac{\mathbf{w}_j\mathbf{w}_j^\top}{\mathbf{w}_j^\top\mathbf{w}_j}\right)\Delta_j + o_p(\lambda_n). \tag{25}$$

From Eq. (24) and Eq. (25), we get:

$$\begin{aligned}
\frac{1}{\lambda_n}(\hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_\mathbf{J}}\tilde{w}_\mathbf{J}) &= O_p(n^{-1/2}\lambda_n^{-1}) + \Sigma_{X_iX_\mathbf{J}}\Sigma_{X_\mathbf{J}X_\mathbf{J}}^{-1} \\
&\left\{\mathrm{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_\mathbf{J} + \lambda_n\Sigma_{X_iX_\mathbf{J}}\Sigma_{X_\mathbf{J}X_\mathbf{J}}^{-1}\mathrm{Diag}\left[d_j/\|\mathbf{w}_j\|\left(I_{p_j} - \frac{\mathbf{w}_j\mathbf{w}_j^\top}{\mathbf{w}_j^\top\mathbf{w}_j}\right)\right]\Delta + o_p(\lambda_n)\right\} \\
&= A + \lambda_n B + o_p(\lambda_n) + O_p(n^{-1/2}\lambda_n^{-1}).
\end{aligned}$$

Since $\lambda_n \gg n^{-1/4}$, we have $O_p(n^{-1/2}\lambda_n^{-1}) = o_p(\lambda_n)$. Thus, since we assumed that $\|A\| = \|\Sigma_{X_iX_\mathbf{J}}\Sigma_{X_\mathbf{J}X_\mathbf{J}}^{-1}\mathrm{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_\mathbf{J}\| = d_i$, we have:

$$\begin{aligned}
\left\|\frac{1}{\lambda_n}(\hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_\mathbf{J}}\tilde{w}_\mathbf{J})\right\|^2 &= \|A\|^2 + 2\lambda_n A^\top B + o_p(\lambda_n)d_i^2 + o_p(\lambda_n) \\
&= d_i^2 + o_p(\lambda_n) \\
&\quad -2\lambda_n\Delta^\top\Sigma_{X_\mathbf{J}X_i}\Sigma_{X_iX_\mathbf{J}}\Sigma_{X_\mathbf{J}X_\mathbf{J}}^{-1}\mathrm{Diag}\left(d_j/\|\mathbf{w}_j\|(I_{p_j} - \frac{\mathbf{w}_j\mathbf{w}_j^\top}{\mathbf{w}_j^\top\mathbf{w}_j})\right)\Delta,
\end{aligned}$$

(note that we have $A = -\Sigma_{X_iX_\mathbf{J}}\Delta$) which is asymptotically strictly smaller than d_i^2 if Eq. (6) is satisfied, which proves optimality and concludes the proof.

B.4 Proof of Proposition 6

As in the proof of Theorem 2 in Appendix B.1, we consider the estimate $\tilde{w}$ built from the reduced problem by constraining $\tilde{w}_{\mathbf{J}^c} = 0$. We consider the following event:

$$E_1 = \{\hat{\Sigma}_{XX} \text{ invertible and } \forall j \in \mathbf{J},\ \tilde{w}_j \neq 0\}.$$

This event has a probability converging to one. Moreover, if E_1 is true, then the group Lasso estimate has the correct sparsity pattern if and only if for all $i \in \mathbf{J}^c$,

$$\left\|\hat{\Sigma}_{X_iX_\mathbf{J}}(\tilde{w}_\mathbf{J} - \mathbf{w}_\mathbf{J}) - \hat{\Sigma}_{X_i\varepsilon}\right\| \leqslant \lambda_n d_i = \lambda_0 n^{-1/2} d_i.$$

Moreover we have by definition of $\tilde{w}_\mathbf{J}$: $\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}}(\tilde{w}_\mathbf{J} - \mathbf{w}_\mathbf{J}) - \hat{\Sigma}_{X_\mathbf{J}\varepsilon} = -\lambda_n\mathrm{Diag}(d_j/\|\tilde{w}_j\|)\tilde{w}_\mathbf{J}$, and thus, we get:

$$\begin{aligned}
&\hat{\Sigma}_{X_iX_\mathbf{J}}(\tilde{w}_\mathbf{J} - \mathbf{w}_\mathbf{J}) - \hat{\Sigma}_{X_i\varepsilon} \\
=\ &\hat{\Sigma}_{X_iX_\mathbf{J}}\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}}^{-1}\hat{\Sigma}_{X_\mathbf{J}\varepsilon} - \hat{\Sigma}_{X_i\varepsilon} - \lambda_0 n^{-1/2}\hat{\Sigma}_{X_iX_\mathbf{J}}\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}}^{-1}\mathrm{Diag}(d_j/\|\tilde{w}_j\|)\tilde{w}_\mathbf{J} \\
=\ &\Sigma_{X_iX_\mathbf{J}}\Sigma_{X_\mathbf{J}X_\mathbf{J}}^{-1}\hat{\Sigma}_{X_\mathbf{J}\varepsilon} - \hat{\Sigma}_{X_i\varepsilon} - \lambda_0 n^{-1/2}\Sigma_{X_iX_\mathbf{J}}\Sigma_{X_\mathbf{J}X_\mathbf{J}}^{-1}\mathrm{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_\mathbf{J} + O_p(n^{-1})
\end{aligned}$$

The random vector $\Sigma_{X\varepsilon} \in \mathbb{R}^p$ is a multivariate U-statistic with square integrable kernel obtained from i.i.d. random vectors; it is thus asymptotically normal (Van der Vaart, 1998) and we simply need to compute its mean and variance. The mean is zero, and the variance is $\frac{n-1}{n^2}\sigma^2\Sigma_{XX} = n^{-1}\sigma^2\Sigma_{XX} + o(n^{-1})$. This implies that the random vector s of size $\mathrm{Card}(\mathbf{J}^c)$ defined by

$$s_i = n^{1/2}\|\hat{\Sigma}_{X_iX_\mathbf{J}}(\tilde{w}_\mathbf{J} - \mathbf{w}_\mathbf{J}) - \hat{\Sigma}_{X_i\varepsilon}\|,$$

is equal to

$$\begin{aligned} s_i &= \left\|\sigma\Sigma_{X_iX_\mathbf{J}}\Sigma_{X_\mathbf{J}X_\mathbf{J}}^{-1}u_\mathbf{J} - \sigma u_i - \lambda_0\Sigma_{X_iX_\mathbf{J}}\Sigma_{X_\mathbf{J}X_\mathbf{J}}^{-1}\mathrm{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_\mathbf{J}\right\| + O_p(n^{-1/2}) \\ &= f_i(u) + O_p(n^{-1/2}), \end{aligned}$$

where $u = \sigma^{-1}n^{-1/2}\hat{\Sigma}_{X\varepsilon}$ and f_i are deterministic continuous functions. The vector $f(u)$ converges in distribution to $f(v)$ where v is normally distributed with mean zero and covariance matrix Σ_{XX}. By Slutsky's lemma (Van der Vaart, 1998), this implies that the random vector s has the same limiting distribution. Thus, the probability $\mathbb{P}(\max_{i\in\mathbf{J}^c} s_i/d_i \leqslant \lambda_0)$ converges to

$$\mathbb{P}\left(\max_{i\in\mathbf{J}^c}\frac{1}{d_i}\left\|\sigma(\Sigma_{X_iX_\mathbf{J}}\Sigma_{X_\mathbf{J}X_\mathbf{J}}^{-1}v_\mathbf{J} - v_i) - \lambda_0\Sigma_{X_iX_\mathbf{J}}\Sigma_{X_\mathbf{J}X_\mathbf{J}}^{-1}\mathrm{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_\mathbf{J}\right\| \leqslant \lambda_0\right).$$

Under the event E_1 which has probability tending to one, we have correct pattern selection if and only if $\max_{i\in\mathbf{J}^c} s_i/d_i \leqslant \lambda_0$, which leads to

$$\mathbb{P}\left(\max_{i\in\mathbf{J}^c}\frac{1}{d_i}\left\|\sigma t_i - \lambda_0\Sigma_{X_iX_\mathbf{J}}\Sigma_{X_\mathbf{J}X_\mathbf{J}}^{-1}\mathrm{Diag}(d_j/\|\mathbf{w}_j\|)\mathbf{w}_\mathbf{J}\right\| \leqslant \lambda_0\right),$$

where $t_i = \Sigma_{X_iX_\mathbf{J}}\Sigma_{X_\mathbf{J}X_\mathbf{J}}^{-1}v_\mathbf{J} - v_i$. The vector t is normally distributed and a short calculation shows that its covariance matrix is equal to $\Sigma_{X_{\mathbf{J}^c}X_{\mathbf{J}^c}|X_\mathbf{J}}$, which concludes the proof.

Appendix C. Detailed Proofs for the Nonparametric Formulation

We first prove lemmas that will be useful for further proofs, and then prove the consistency results for the nonparametric case.

C.1 Useful Lemmas on Empirical Covariance Operators

We first have the following lemma, proved by Fukumizu et al. (2007), which states that the empirical covariance estimator converges in probability at rate $O_p(n^{-1/2})$ to the population covariance operators:

Lemma 22 *Assume (**A4**) and (**A6**). Then $\|\hat{\Sigma}_{XX} - \Sigma_{XX}\|_\mathcal{F} = O_p(n^{-1/2})$ (for the operator norm), $\|\hat{\Sigma}_{XY} - \Sigma_{XY}\|_\mathcal{F} = O_p(n^{-1/2})$ and $\|\hat{\Sigma}_{X\varepsilon}\|_\mathcal{F} = O_p(n^{-1/2})$.*

The following lemma is useful in several proofs:

Lemma 23 *Assume (**A4**). Then $\left\|\left(\hat{\Sigma}_{XX} + \mu_n I\right)^{-1}\Sigma_{XX} - \left(\Sigma_{XX} + \mu_n I\right)^{-1}\Sigma_{XX}\right\|_\mathcal{F} = O_p(\frac{\mu_n^{-1}}{n^{1/2}})$, and $\left\|\left(\hat{\Sigma}_{XX} + \mu_n I\right)^{-1}\hat{\Sigma}_{XX} - \left(\Sigma_{XX} + \mu_n I\right)^{-1}\Sigma_{XX}\right\|_\mathcal{F} = O_p(\frac{\mu_n^{-1}}{n^{1/2}})$.*

Proof We have:

$$\begin{aligned} & \left(\hat{\Sigma}_{XX}+\mu_n I\right)^{-1}\Sigma_{XX}-(\Sigma_{XX}+\mu_n I)^{-1}\Sigma_{XX} \\ = \quad & \left(\hat{\Sigma}_{XX}+\mu_n I\right)^{-1}(\Sigma_{XX}-\hat{\Sigma}_{XX})(\Sigma_{XX}+\mu_n I)^{-1}\Sigma_{XX}. \end{aligned}$$

This is the product of operators whose norms are respectively upper bounded by μ_n^{-1}, $O_p(n^{-1/2})$ and 1, which leads to the first inequality (we use $\|AB\|_{\mathcal{F}} \leqslant \|A\|_{\mathcal{F}}\|B\|_{\mathcal{F}}$). The second inequality follows along similar lines. ■

Note that the two previous lemma also hold for any suboperator of Σ_{XX}, that is, for $\Sigma_{X_\mathbf{J}X_\mathbf{J}}$, or $\Sigma_{X_iX_i}$.

Lemma 24 *Assume (**A4**), (**A5**) and (**A7**). There exists $\mathbf{h}_\mathbf{J} \in \mathcal{F}_\mathbf{J}$ such that $\mathbf{f}_\mathbf{J} = \Sigma^{1/2}_{X_\mathbf{J}X_\mathbf{J}}\mathbf{h}_\mathbf{J}$.*

Proof The range condition implies that

$$\mathbf{f}_\mathbf{J} \quad = \quad \mathrm{Diag}(\Sigma^{1/2}_{X_jX_j})\mathbf{g}_\mathbf{J} = \mathrm{Diag}(\Sigma^{1/2}_{X_jX_j})C^{1/2}_{X_\mathbf{J}X_\mathbf{J}}C^{-1/2}_{X_\mathbf{J}X_\mathbf{J}}\mathbf{g}_\mathbf{J}$$

(because C_{XX} is invertible). The result follows from the identity

$$\Sigma_{X_\mathbf{J}X_\mathbf{J}} = \mathrm{Diag}(\Sigma^{1/2}_{X_jX_j})C^{1/2}_{X_\mathbf{J}X_\mathbf{J}}(\mathrm{Diag}(\Sigma^{1/2}_{X_jX_j})C^{1/2}_{X_\mathbf{J}X_\mathbf{J}})^*$$

and the fact that if $\Sigma_{X_\mathbf{J}X_\mathbf{J}} = UU^*$ and $f = U\alpha$ then there exists β such that $f = \Sigma^{1/2}_{X_\mathbf{J}X_\mathbf{J}}\beta$ (Baker, 1973).[5] ■

C.2 Proof of Theorem 11

We now extend Lemma 20 to covariance operators, which requires to use the alternative formulation and a slower rate of decrease for the regularization parameter:

Lemma 25 *Let $\tilde{f}_\mathbf{J}$ be any minimizer of*

$$\frac{1}{2}\hat{\Sigma}_{YY} - \langle \hat{\Sigma}_{X_\mathbf{J}Y}, f_\mathbf{J}\rangle_{\mathcal{F}_\mathbf{J}} + \frac{1}{2}\langle f_\mathbf{J}, \hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}} f_\mathbf{J}\rangle_{\mathcal{F}_\mathbf{J}} + \frac{\mu_n}{2}\left(\sum_{j\in\mathbf{J}} d_j \|f_j\|_{\mathcal{F}_j}\right)^2.$$

If $\mu_n \to 0$ and $\mu_n n^{1/2} \to +\infty$, then $\|\tilde{f}_\mathbf{J} - \mathbf{f}_\mathbf{J}\|_{\mathcal{F}_\mathbf{J}}$ converges to zero in probability. Moreover for any η_n such that $\eta_n \gg \mu_n^{1/2} + \mu_n^{-1}n^{-1/2}$ then $\|\tilde{f}_\mathbf{J} - \mathbf{f}_\mathbf{J}\|_{\mathcal{F}_\mathbf{J}} = O_p(\eta_n)$.

Proof Note that from Cauchy-Schwarz inequality, we have:

$$\begin{aligned} \left(\sum_{j\in\mathbf{J}} d_j\|f_j\|_{\mathcal{F}_j}\right)^2 \quad &= \quad \left(\sum_{j\in\mathbf{J}} d_j^{1/2}\|\mathbf{f}_j\|_{\mathcal{F}_j}^{1/2} \times \frac{d_j^{1/2}\|f_j\|_{\mathcal{F}_j}}{\|\mathbf{f}_j\|_{\mathcal{F}_j}^{1/2}}\right)^2 \\ &\leqslant \quad \left(\sum_{j\in\mathbf{J}} d_j\|\mathbf{f}_j\|_{\mathcal{F}_j}\right)\sum_{j\in\mathbf{J}}\frac{d_j\|f_j\|_{\mathcal{F}_j}^2}{\|\mathbf{f}_j\|_{\mathcal{F}_j}}, \end{aligned}$$

5. The adjoint operator V^* of $V : \mathcal{F}_i \to \mathcal{F}_\mathbf{J}$ is so that for all $f \in \mathcal{F}_i$ and $g \in \mathcal{F}_\mathbf{J}$, $\langle f, Vg\rangle_{\mathcal{F}_i} = \langle V^*f, g\rangle_{\mathcal{F}_\mathbf{J}}$ (Brezis, 1980).

with equality if and only if there exists $\alpha > 0$ such that $\|f_j\|_{\mathcal{F}_j} = \alpha \|\mathbf{f}_j\|_{\mathcal{F}_j}$ for all $j \in \mathbf{J}$. We consider the unique minimizer $\bar{f}_{\mathbf{J}}$ of the following cost function, built by replacing the regularization by its upperbound,

$$F(f_{\mathbf{J}}) = \frac{1}{2}\hat{\Sigma}_{YY} - \langle \hat{\Sigma}_{X_{\mathbf{J}}Y}, f_{\mathbf{J}} \rangle_{\mathcal{F}_{\mathbf{J}}} + \frac{1}{2}\langle f_{\mathbf{J}}, \hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} f_{\mathbf{J}} \rangle_{\mathcal{F}_{\mathbf{J}}} + \frac{\mu_n}{2}\left(\sum_{j \in \mathbf{J}} d_j \|\mathbf{f}_j\|_{\mathcal{F}_j} \right) \sum_{j \in \mathbf{J}} \frac{d_j \|f_j\|^2_{\mathcal{F}_j}}{\|\mathbf{f}_j\|_{\mathcal{F}_j}}.$$

Since it is a regularized least-square problem, we have (with $\varepsilon = Y - \sum_{j \in \mathbf{J}} \mathbf{f}_j(X) - \mathbf{b}$):

$$\bar{f}_{\mathbf{J}} = \left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D \right)^{-1} \left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} \mathbf{f}_{\mathbf{J}} + \hat{\Sigma}_{X_{\mathbf{J}}\varepsilon} \right),$$

where $D = \left(\sum_{j \in \mathbf{J}} d_j \|\mathbf{f}_j\| \right) \mathrm{Diag}(d_j / \|\mathbf{f}_j\|)$. Note that D is upperbounded and lowerbounded, as an auto-adjoint operator, by *strictly positive* constants times the identity operator (with probability tending to one), that is, $D_{\max} I_{\mathcal{F}_{\mathbf{J}}} \succcurlyeq D \succcurlyeq D_{\min} I_{\mathcal{F}_{\mathbf{J}}}$ with $D_{\min}, D_{\max} > 0$. We now prove that $\bar{f}_{\mathbf{J}} - \mathbf{f}_{\mathbf{J}}$ is converging to zero in probability. We have:

$$\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D \right)^{-1} \hat{\Sigma}_{X_{\mathbf{J}}\varepsilon} = O_p(n^{-1/2}\mu_n^{-1}),$$

because of Lemma 22 and $\left\| \left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D \right)^{-1} \right\|_{\mathcal{F}_{\mathbf{J}}} \leqslant D_{\min}^{-1} \mu_n^{-1}$. Moreover, similarly, we have

$$\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D \right)^{-1} \hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} f_{\mathbf{J}} - \left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D \right)^{-1} \Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} \mathbf{f}_{\mathbf{J}} = O_p(n^{-1/2}\mu_n^{-1}).$$

Besides, by Lemma 23,

$$\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D \right)^{-1} \Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} f_{\mathbf{J}} - \left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D \right)^{-1} \Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} f_{\mathbf{J}} = O_p(n^{-1/2}\mu_n^{-1}).$$

Thus $\bar{f}_{\mathbf{J}} - \mathbf{f}_{\mathbf{J}} = V + O_p(n^{-1/2}\mu_n^{-1})$, where

$$V = \left[\left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D \right)^{-1} \Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} - I \right] \mathbf{f}_{\mathbf{J}} = -\left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D \right)^{-1} \mu_n D \mathbf{f}_{\mathbf{J}}.$$

We have

$$\begin{aligned}
\|V\|^2_{\mathcal{F}_{\mathbf{J}}} &= \mu_n^2 \langle \mathbf{f}_{\mathbf{J}}, D\left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D \right)^{-2} D \mathbf{f}_{\mathbf{J}} \rangle_{\mathcal{F}_{\mathbf{J}}} \\
&\leqslant D_{\max}^2 \mu_n^2 \langle \mathbf{f}_{\mathbf{J}}, \left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_{\min} I \right)^{-2} \mathbf{f}_{\mathbf{J}} \rangle_{\mathcal{F}_{\mathbf{J}}} \\
&\leqslant D_{\max}^2 \mu_n \langle \mathbf{f}_{\mathbf{J}}, \left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_{\min} I \right)^{-1} \mathbf{f}_{\mathbf{J}} \rangle_{\mathcal{F}_{\mathbf{J}}} \\
&\leqslant D_{\max}^2 \mu_n \langle \mathbf{h}_{\mathbf{J}}, \Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} \left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_{\min} I \right)^{-1} \mathbf{h}_{\mathbf{J}} \rangle_{\mathcal{F}_{\mathbf{J}}} \text{ by Lemma 24,} \\
&\leqslant D_{\max}^2 \mu_n \|\mathbf{h}_{\mathbf{J}}\|^2_{\mathcal{F}_{\mathbf{J}}}.
\end{aligned}$$

Finally we obtain $\|\bar{f}_{\mathbf{J}} - \mathbf{f}_{\mathbf{J}}\|_{\mathcal{F}_{\mathbf{J}}} = O_p(\mu_n^{1/2} + n^{-1/2}\mu_n^{-1})$.

We now consider the cost function defining $\tilde{f}_{\mathbf{J}}$:

$$F_n(f_{\mathbf{J}}) = \frac{1}{2}\hat{\Sigma}_{YY} - \langle \hat{\Sigma}_{X_{\mathbf{J}}Y}, f_{\mathbf{J}} \rangle_{\mathcal{F}_{\mathbf{J}}} + \frac{1}{2}\langle f_{\mathbf{J}}, \hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} f_{\mathbf{J}} \rangle_{\mathcal{F}_{\mathbf{J}}} + \frac{\mu_n}{2}\left(\sum_{j \in \mathbf{J}} d_j \|f_j\|_{\mathcal{F}_j} \right)^2.$$

We have (note that although we seem to take infinite dimensional derivatives, everything can be done in the finite subspace spanned by the data):

$$\begin{aligned} F_n(f_{\mathbf{J}}) - F(f_{\mathbf{J}}) &= \frac{\mu_n}{2}\left[\left(\sum_{j\in\mathbf{J}} d_j \|f_j\|_{\mathcal{F}_j}\right)^2 - \left(\sum_{j\in\mathbf{J}} d_j \|\mathbf{f}_j\|_{\mathcal{F}_j}\right)\sum_{j\in\mathbf{J}} \frac{d_j\|f_j\|_{\mathcal{F}_j}^2}{\|\mathbf{f}_j\|_{\mathcal{F}_j}}\right], \\ \nabla_{f_i} F_n(f_{\mathbf{J}}) - \nabla_{f_i} F(f_{\mathbf{J}}) &= \mu_n\left[\left(\sum_{j\in\mathbf{J}} d_j \|f_j\|_{\mathcal{F}_j}\right)\frac{d_i f_i}{\|f_i\|_{\mathcal{F}_j}} - \left(\sum_{j\in\mathbf{J}} d_j \|\mathbf{f}_j\|_{\mathcal{F}_j}\right)\frac{d_i f_i}{\|\mathbf{f}_i\|_{\mathcal{F}_j}}\right]. \end{aligned}$$

Since the right hand side of the previous equation corresponds to a continuously differentiable function of $f_{\mathbf{J}}$ around $\mathbf{f}_{\mathbf{J}}$ (with upper-bounded derivatives around $\mathbf{f}_{\mathbf{J}}$), we have:

$$\|\nabla_{f_i} F_n(\bar{f}_{\mathbf{J}}) - 0\|_{\mathcal{F}_i} \leqslant C\mu_n \|\mathbf{f}_{\mathbf{J}} - \bar{f}_{\mathbf{J}}\|_{\mathcal{F}_{\mathbf{J}}} = \mu_n O_p(\mu_n^{1/2} + n^{-1/2}\mu_n^{-1}).$$

for some constant $C > 0$. Moreover, on the ball of center $\bar{f}_{\mathbf{J}}$ and radius η_n such that $\eta_n \gg \mu_n^{1/2} + \mu_n^{-1}n^{-1/2}$ (to make sure that it asymptotically contains $\mathbf{f}_{\mathbf{J}}$, which implies that on the ball each f_j, $j \in \mathbf{J}$ are bounded away from zero), and $\eta_n \ll 1$ (so that we get consistency), we have a lower bound on the second derivative of $\left(\sum_{j\in\mathbf{J}} d_j \|f_j\|_{\mathcal{F}_j}\right)$. Thus for any element of the ball,

$$F_n(f_{\mathbf{J}}) \geqslant F_n(\bar{f}_{\mathbf{J}}) + \langle \nabla_{f_{\mathbf{J}}} F_n(\bar{f}_{\mathbf{J}}), (f_{\mathbf{J}} - \bar{f}_{\mathbf{J}})\rangle_{\mathcal{F}_{\mathbf{J}}} + C'\mu_n \|f_{\mathbf{J}} - \bar{f}_{\mathbf{J}}\|_{\mathcal{F}_{\mathbf{J}}}^2,$$

where $C' > 0$ is a constant. This implies that the value of $F_n(f_{\mathbf{J}})$ on the edge of the ball is larger than

$$F_n(\bar{f}_{\mathbf{J}}) + \eta_n\mu_n O_p(\mu_n^{1/2} + n^{-1/2}\mu_n^{-1}) + C'\eta_n^2\mu_n,$$

Thus if $\eta_n^2\mu_n \gg \eta_n\mu_n^{3/2}$ and $\eta_n^2\mu_n \gg n^{-1/2}\eta_n$, then we must have all minima inside the ball of radius η_n (because with probability tending to one, the value on the edge is greater than one value inside and the function is convex) which implies that the global minimum of F_n is at most η_n away from $\bar{f}_{\mathbf{J}}$ and thus since $\bar{f}_{\mathbf{J}}$ is $O(\mu_n^{1/2})$ away from $\mathbf{f}_{\mathbf{J}}$, we have the consistency if

$$\eta_n \ll 1 \text{ and } \eta_n \gg \mu_n^{1/2} + n^{-1/2}\mu_n^{-1},$$

which concludes the proof of the lemma. ■

We now prove Theorem 11. Let $\tilde{f}_{\mathbf{J}}$ be defined as in Lemma 20. We extend it by zeros on $\mathbf{J}^c$. We already know the squared norm consistency by Lemma 20. Since by Proposition 14, the solution is unique with probability tending to one, we need to prove that with probability tending to one $\tilde{f}$ is optimal for problem in Eq. (13). We have by the first optimality condition for $\tilde{f}_{\mathbf{J}}$:

$$\hat{\Sigma}_{X_{\mathbf{J}}Y} - \hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}\tilde{f}_{\mathbf{J}} = \mu_n \|\tilde{f}\|_d \operatorname{Diag}(d_j/\|\tilde{f}_j\|)\tilde{f}_{\mathbf{J}},$$

where we use the notation $\|f\|_d = \sum_{j=1}^m d_j\|f_j\|_{\mathcal{F}_j}$ (note the difference with the norm $\|f\|_{\mathcal{F}} = (\sum_{j=1}^m \|f_j\|_{\mathcal{F}_j}^2)^{1/2}$). We thus have by solving for $\tilde{f}_{\mathbf{J}}$ and using $\hat{\Sigma}_{X_{\mathbf{J}}Y} = \hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}\mathbf{f}_{\mathbf{J}} + \hat{\Sigma}_{X_{\mathbf{J}}\varepsilon}$:

$$\tilde{f}_{\mathbf{J}} = \left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_n\right)^{-1}\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}\mathbf{f}_{\mathbf{J}} + \hat{\Sigma}_{X_{\mathbf{J}}\varepsilon}\right),$$

with the notation $D_n = \|\tilde{f}\|_d \operatorname{Diag}(d_j/\|\tilde{f}_j\|_{\mathcal{F}_j})$. We can now put that back into $\hat{\Sigma}_{X_{\mathbf{J}^c}Y} - \hat{\Sigma}_{X_{\mathbf{J}^c}X_{\mathbf{J}}}\tilde{f}_{\mathbf{J}}$ and show that this will have small enough norm with probability tending to one. We have for all $i \in \mathbf{J}^c$:

$$\begin{aligned}
\hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_{\mathbf{J}}}\tilde{f}_{\mathbf{J}} &= \hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_{\mathbf{J}}}\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_n\right)^{-1}\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}\mathbf{f}_{\mathbf{J}} + \hat{\Sigma}_{X_{\mathbf{J}}\varepsilon}\right) \\
&= -\hat{\Sigma}_{X_iX_{\mathbf{J}}}\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_n\right)^{-1}\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}\mathbf{f}_{\mathbf{J}} \\
&\quad + \hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_{\mathbf{J}}}\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_n\right)^{-1}\hat{\Sigma}_{X_{\mathbf{J}}\varepsilon} \\
&= -\hat{\Sigma}_{X_iX_{\mathbf{J}}}\mathbf{f}_{\mathbf{J}} + \hat{\Sigma}_{X_iX_{\mathbf{J}}}\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_n\right)^{-1}\mu_n D_n\mathbf{f}_{\mathbf{J}} \\
&\quad + \hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_{\mathbf{J}}}\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_n\right)^{-1}\hat{\Sigma}_{X_{\mathbf{J}}\varepsilon} \\
&= \hat{\Sigma}_{X_iX_{\mathbf{J}}}\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_n\right)^{-1}\mu_n D_n\mathbf{f}_{\mathbf{J}} \\
&\quad + \hat{\Sigma}_{X_i\varepsilon} - \hat{\Sigma}_{X_iX_{\mathbf{J}}}\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_n\right)^{-1}\hat{\Sigma}_{X_{\mathbf{J}}\varepsilon} \qquad (26) \\
&= A_n + B_n.
\end{aligned}$$

The first term A_n (divided by μ_n) is equal to

$$\frac{A_n}{\mu_n} = \hat{\Sigma}_{X_iX_{\mathbf{J}}}\left(\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_n\right)^{-1} D_n\mathbf{f}_{\mathbf{J}}.$$

We can replace $\hat{\Sigma}_{X_iX_{\mathbf{J}}}$ in $\frac{A_n}{\mu_n}$ by $\Sigma_{X_iX_{\mathbf{J}}}$ at cost $O_p(n^{-1/2}\mu_n^{-1/2})$ because $\langle \mathbf{f}_{\mathbf{J}}, \Sigma^{-1}_{X_{\mathbf{J}}X_{\mathbf{J}}}\mathbf{f}_{\mathbf{J}}\rangle_{\mathcal{F}_{\mathbf{J}}} < \infty$ (by Lemma 24). Also, we can replace $\hat{\Sigma}_{X_{\mathbf{J}}X_{\mathbf{J}}}$ in $\frac{A_n}{\mu_n}$ by $\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}$ at cost $O_p(n^{-1/2}\mu_n^{-1})$ as a consequence of Lemma 23. Those two are $o_p(1)$ by assumptions on μ_n. Thus,

$$\frac{A_n}{\mu_n} = \Sigma_{X_iX_{\mathbf{J}}}\left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D_n\right)^{-1} D_n\mathbf{f}_{\mathbf{J}} + o_p(1).$$

Furthermore, we denote $D = \|\mathbf{f}\|_d \operatorname{Diag}(d_j/\|\mathbf{f}_j\|_{\mathcal{F}_j})$. From Lemma 25, we know that $D_n - D = o_p(1)$. Thus we can replace D_n by D at cost $o_p(1)$ to get:

$$\frac{A_n}{\mu_n} = \Sigma_{X_iX_{\mathbf{J}}}\left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D\right)^{-1} D\mathbf{f}_{\mathbf{J}} + o_p(1) = C_n + o_p(1).$$

We now show that this last deterministic term $C_n \in \mathcal{F}_i$ converges to:

$$C = \Sigma^{1/2}_{X_iX_i} C_{X_iX_{\mathbf{J}}} C^{-1}_{X_{\mathbf{J}}X_{\mathbf{J}}} D\mathbf{g}_{\mathbf{J}},$$

where, from (**A7**), $\forall j \in \mathbf{J}$, $\mathbf{f}_j = \Sigma^{1/2}_{X_jX_j}\mathbf{g}_j$. We have

$$\begin{aligned}
C_n - C &= \Sigma^{1/2}_{X_iX_i} C_{X_iX_{\mathbf{J}}}\left[\operatorname{Diag}(\Sigma^{1/2}_{X_jX_j})\left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D\right)^{-1}\operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}) - C^{-1}_{X_{\mathbf{J}}X_{\mathbf{J}}}\right] D\mathbf{g}_{\mathbf{J}} \\
&= \Sigma^{1/2}_{X_iX_i} C_{X_iX_{\mathbf{J}}} K_n D\mathbf{g}_{\mathbf{J}}.
\end{aligned}$$

where $K_n = \operatorname{Diag}(\Sigma^{1/2}_{X_jX_j})\left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D\right)^{-1}\operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}) - C^{-1}_{X_{\mathbf{J}}X_{\mathbf{J}}}$. In addition, we have:

$$\begin{aligned}
\operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}) C_{X_{\mathbf{J}}X_{\mathbf{J}}} K_n &= \Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}\left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D\right)^{-1}\operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}) - \operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}) \\
&= -\mu_n D\left(\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} + \mu_n D\right)^{-1}\operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}).
\end{aligned}$$

Following Fukumizu et al. (2007), the range of the adjoint operator $\left(\Sigma_{X_iX_i}^{1/2} C_{X_iX_\mathbf{J}}\right)^* = C_{X_\mathbf{J}X_i}\Sigma_{X_iX_i}^{1/2}$ is included in the closure of the range of $\mathrm{Diag}(\Sigma_{X_jX_j})$ (which is equal to the range of $\Sigma_{X_\mathbf{J}X_\mathbf{J}}$ by Lemma 24). For any $v_\mathbf{J} \in \mathcal{F}_\mathbf{J}$ in the intersection of two ranges, we have $v_\mathbf{J} = C_{X_\mathbf{J}X_\mathbf{J}}\mathrm{Diag}(\Sigma_{X_jX_j}^{1/2})u_\mathbf{J}$ (note that $C_{X_\mathbf{J}X_\mathbf{J}}$ is invertible), and thus

$$\begin{aligned}\langle K_n D\mathbf{g}_\mathbf{J}, v_\mathbf{J}\rangle_{\mathcal{F}_\mathbf{J}} &= \langle K_n D\mathbf{g}_\mathbf{J}, C_{X_\mathbf{J}X_\mathbf{J}}\mathrm{Diag}(\Sigma_{X_jX_j}^{1/2})u_\mathbf{J}\rangle_{\mathcal{F}_\mathbf{J}} \\ &= \langle -\mu_n D\left(\Sigma_{X_\mathbf{J}X_\mathbf{J}} + \mu_n D\right)^{-1}\mathrm{Diag}(\Sigma_{X_jX_j}^{1/2})D\mathbf{g}_\mathbf{J}, u_\mathbf{J}\rangle_{\mathcal{F}_\mathbf{J}}\end{aligned}$$

which is $O_p(\mu_n^{1/2})$ and thus tends to zero. Since this holds for all elements in the intersection of the ranges, Lemma 9 by Fukumizu et al. (2007) implies that $\|C_n - C\|_{\mathcal{F}_\mathbf{J}}$ converges to zero.

We now simply need to show that the second term B_n is dominated by μ_n. We have: $\|\hat{\Sigma}_{X_i\varepsilon}\|_{\mathcal{F}_i} = O_p(n^{-1/2})$ and $\|\hat{\Sigma}_{X_iX_\mathbf{J}}\left(\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}} + \mu_n D_n\right)^{-1}\hat{\Sigma}_{X_\mathbf{J}\varepsilon}\|_{\mathcal{F}_i} \leqslant \|\hat{\Sigma}_{X_i\varepsilon}\|_{\mathcal{F}_i}$, thus, since $\mu_n n^{1/2} \to +\infty$, $B_n = o_p(\mu_n)$ and therefore for for each $i \in J^c$,

$$\frac{1}{d_i\mu_n\|\mathbf{f}\|_d}\left(\hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_\mathbf{J}}\tilde{f}_\mathbf{J}\right)$$

converges in probability to $\|C\|_{\mathcal{F}_\mathbf{J}}/d_i\|\mathbf{f}\|_d$ which is strictly smaller than one because Eq. (16) is satisfied. Thus

$$\mathbb{P}\left\{\frac{1}{d_i\mu_n\|\mathbf{f}\|_d}\left\|\hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_\mathbf{J}}\tilde{f}_\mathbf{J}\right\|_{\mathcal{F}_i} \leqslant 1\right\}$$

is tending to 1, which implies the theorem (using the same arguments than in the proof of Theorem 2 in Appendix B.1).

C.3 Proof of Theorem 12

Before proving the analog of the second group Lasso theorem, we need the following additional proposition, which states that consistency of the patterns can only be achieved if $\mu_n n^{1/2} \to \infty$ (even if chosen in a data dependent way).

Proposition 26 *Assume (**A4-7**) and that* $\mathbf{J}$ *is not empty. If* $\hat{f}$ *is converging in probability to* $\mathbf{f}$ *and* $J(\hat{f})$ *converges in probability to* $\mathbf{J}$*, then* $\mu_n n^{1/2} \to \infty$ *in probability.*

Proof We give a proof by contradiction, and we thus assume that there exists $M > 0$ such that $\liminf_{n\to\infty}\mathbb{P}(\mu_n n^{1/2} < M) > 0$. This imposes that there exists a subsequence which is almost surely bounded by M (Durrett, 2004). Thus, we can take a further subsequence which converges to a limit $\mu_0 \in [0,\infty)$. We now consider such a subsequence (and still use the notation of the original sequence for simplicity).

With probability tending to one, we have the optimality condition (15):

$$\hat{\Sigma}_{X_\mathbf{J}\varepsilon} + \hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}}\mathbf{f}_\mathbf{J} = \hat{\Sigma}_{X_\mathbf{J}Y} = \hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}}\hat{f}_\mathbf{J} + \mu_n\|\hat{f}\|_d\,\mathrm{Diag}(d_j/\|\hat{f}_j\|_{\mathcal{F}_j})\hat{f}_\mathbf{J}.$$

If we denote $D_n = n^{1/2}\mu_n\|\hat{f}\|_d\,\mathrm{Diag}(d_j/\|\hat{f}_j\|_{\mathcal{F}_j})$, we get:

$$D_n\mathbf{f}_\mathbf{J} = \left[\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}} + D_n n^{-1/2}\right]n^{1/2}\left[\mathbf{f}_\mathbf{J} - \hat{f}_\mathbf{J}\right] + n^{1/2}\hat{\Sigma}_{X_\mathbf{J}\varepsilon},$$

which can be approximated as follows (we denote $D = \|\mathbf{f}\|_d \operatorname{Diag}(d_j/\|\mathbf{f}_j\|_{\mathcal{F}_j})$):

$$\mu_0 D\mathbf{f}_{\mathbf{J}} + o_p(1) = \Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} n^{1/2}\left[\mathbf{f}_{\mathbf{J}} - \hat{f}_{\mathbf{J}}\right] + o_p(1) + n^{1/2}\hat{\Sigma}_{X_{\mathbf{J}}\varepsilon}.$$

We can now write for $i \in \mathbf{J}^c$:

$$\begin{aligned} n^{1/2}\left(\hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_{\mathbf{J}}}\hat{f}_{\mathbf{J}}\right) &= n^{1/2}\hat{\Sigma}_{X_i\varepsilon} + \hat{\Sigma}_{X_iX_{\mathbf{J}}} n^{1/2}(\mathbf{f}_{\mathbf{J}} - \hat{f}_{\mathbf{J}}) \\ &= n^{1/2}\hat{\Sigma}_{X_i\varepsilon} + \Sigma_{X_iX_{\mathbf{J}}} n^{1/2}(\mathbf{f}_{\mathbf{J}} - \hat{f}_{\mathbf{J}}) + o_p(1). \end{aligned}$$

We now consider an arbitrary vector $w_{\mathbf{J}} \in \mathcal{F}_{\mathbf{J}}$, such that $\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} w_{\mathbf{J}}$ is different from zero (such vector exists because $\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} \neq 0$, as we have assumed in (**A4**) that the variables are not constant). Since the range of $\Sigma_{X_{\mathbf{J}}X_i}$ is included in the range of $\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}}$ (Baker, 1973), there exists $v_i \in \mathcal{F}_i$ such that $\Sigma_{X_{\mathbf{J}}X_i} v_i = \Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} w_{\mathbf{J}}$. Note that since $\Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} w_{\mathbf{J}}$ is different from zero, we must have $\Sigma^{1/2}_{X_iX_i} v_i \neq 0$. We have:

$$\begin{aligned} n^{1/2}\langle v_i, \hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_{\mathbf{J}}}\hat{f}_{\mathbf{J}}\rangle_{\mathcal{F}_i} &= n^{1/2}\langle v_i, \hat{\Sigma}_{X_i\varepsilon}\rangle_{\mathcal{F}_i} + \langle w_{\mathbf{J}}, \Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} n^{1/2}(\mathbf{f}_{\mathbf{J}} - \hat{f}_{\mathbf{J}})\rangle_{\mathcal{F}_{\mathbf{J}}} + o_p(1) \\ &= n^{1/2}\langle v_i, \hat{\Sigma}_{X_i\varepsilon}\rangle_{\mathcal{F}_i} + \langle w_{\mathbf{J}}, \mu_0 D f_{\mathbf{J}} - n^{1/2}\hat{\Sigma}_{X_{\mathbf{J}}\varepsilon}\rangle_{\mathcal{F}_{\mathbf{J}}} + o_p(1) \\ &= \langle w_{\mathbf{J}}, \mu_0 D f_{\mathbf{J}}\rangle_{\mathcal{F}_{\mathbf{J}}} + n^{1/2}\langle v_i, \hat{\Sigma}_{X_i\varepsilon}\rangle_{\mathcal{F}_i} - n^{1/2}\langle w_{\mathbf{J}}, \hat{\Sigma}_{X_{\mathbf{J}}\varepsilon}\rangle_{\mathcal{F}_{\mathbf{J}}} + o_p(1). \end{aligned}$$

The random variable $E_n = n^{1/2}\langle v_i, \hat{\Sigma}_{X_i\varepsilon}\rangle - n^{1/2}\langle w_{\mathbf{J}}, \hat{\Sigma}_{X_{\mathbf{J}}\varepsilon}\rangle$ is a U-statistic with square integrable kernel obtained from i.i.d. random vectors; it is thus asymptotically normal (Van der Vaart, 1998) and we simply need to compute its mean and variance. The mean is zero and a short calculation similar to the one found in the proof of Theorem 3 in Appendix B.2 shows that we have:

$$\begin{aligned} \mathbb{E}E_n^2 &\geqslant (1-1/n)\sigma^2_{\min}\langle v_i, \Sigma_{X_iX_i} v_i\rangle_{\mathcal{F}_i} + \sigma^2_{\min}\langle w_{\mathbf{J}}, \Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} w_{\mathbf{J}}\rangle_{\mathcal{F}_{\mathbf{J}}} - 2\sigma^2_{\min}\langle v_i, \Sigma_{X_iX_{\mathbf{J}}} w_{\mathbf{J}}\rangle_{\mathcal{F}_i} \\ &= (1-1/n)(\sigma^2_{\min}\langle v_i, \Sigma_{X_iX_i} v_i\rangle_{\mathcal{F}_i} - \sigma^2_{\min}\langle v_i, \Sigma_{X_iX_{\mathbf{J}}} w_{\mathbf{J}}\rangle_{\mathcal{F}_i}). \end{aligned}$$

The operator $C^{-1}_{X_{\mathbf{J}}X_{\mathbf{J}}} C_{X_{\mathbf{J}}X_i}$ has the same range as $C_{X_{\mathbf{J}}X_{\mathbf{J}}}$ (because C_{XX} is invertible), and is thus included in the closure of the range of $\operatorname{Diag}(\Sigma^{1/2}_{X_jX_j})$ (Baker, 1973). Thus, for any $u \in \mathcal{F}_i$, $C^{-1}_{X_{\mathbf{J}}X_{\mathbf{J}}} C_{X_{\mathbf{J}}X_i} u$ can be expressed as a limit of terms of the form $\operatorname{Diag}(\Sigma^{1/2}_{X_jX_j})t$ where $t \in \mathcal{F}_{\mathbf{J}}$. We thus have that

$$\langle u, C_{X_iX_{\mathbf{J}}} \operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}) w_{\mathbf{J}}\rangle_{\mathcal{F}_i} = \langle u, C_{X_iX_{\mathbf{J}}} C^{-1}_{X_{\mathbf{J}}X_{\mathbf{J}}} C_{X_{\mathbf{J}}X_{\mathbf{J}}} \operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}) w_{\mathbf{J}}\rangle_{\mathcal{F}_i}$$

can be expressed as a limit of terms of the form

$$\begin{aligned} \langle t, \operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}) C_{X_{\mathbf{J}}X_{\mathbf{J}}} \operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}) w_{\mathbf{J}}\rangle_{\mathcal{F}_{\mathbf{J}}} &= \langle t, \Sigma_{X_{\mathbf{J}}X_{\mathbf{J}}} w_{\mathbf{J}}\rangle_{\mathcal{F}_{\mathbf{J}}} = \langle t, \Sigma_{X_{\mathbf{J}}X_i} v_i\rangle_{\mathcal{F}_{\mathbf{J}}} \\ &= \langle t, \operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}) C_{X_{\mathbf{J}}X_i} \Sigma^{1/2}_{X_iX_i} v_i\rangle_{\mathcal{F}_{\mathbf{J}}} \to \langle u, C_{X_iX_{\mathbf{J}}} C^{-1}_{X_{\mathbf{J}}X_{\mathbf{J}}} C_{X_{\mathbf{J}}X_i} \Sigma^{1/2}_{X_iX_i} v_i\rangle_{\mathcal{F}_i}. \end{aligned}$$

This implies that $C_{X_iX_{\mathbf{J}}} \operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}) w_{\mathbf{J}} = C_{X_iX_{\mathbf{J}}} C^{-1}_{X_{\mathbf{J}}X_{\mathbf{J}}} C_{X_{\mathbf{J}}X_i} \Sigma^{1/2}_{X_iX_i} v_i$, and thus we have:

$$\begin{aligned} \mathbb{E}E_n^2 &\geqslant \sigma^2_{\min}\langle v_i, \Sigma_{X_iX_i} v_i\rangle_{\mathcal{F}_i} - \sigma^2_{\min}\langle v_i, \Sigma^{1/2}_{X_iX_i} C_{X_iX_{\mathbf{J}}} \operatorname{Diag}(\Sigma^{1/2}_{X_jX_j}) w_{\mathbf{J}}\rangle_{\mathcal{F}_i} \\ &= \sigma^2_{\min}\langle v_i, \Sigma_{X_iX_i} v_i\rangle_{\mathcal{F}_i} - \sigma^2_{\min}\langle v_i, \Sigma^{1/2}_{X_iX_i} C_{X_iX_{\mathbf{J}}} C^{-1}_{X_{\mathbf{J}}X_{\mathbf{J}}} C_{X_{\mathbf{J}}X_i} \Sigma^{1/2}_{X_iX_i} v_i\rangle_{\mathcal{F}_i} \\ &= \sigma^2_{\min}\langle \Sigma^{1/2}_{X_iX_i} v_i, (I_{\mathcal{F}_i} - C_{X_iX_{\mathbf{J}}} C^{-1}_{X_{\mathbf{J}}X_{\mathbf{J}}} C_{X_{\mathbf{J}}X_i}) \Sigma^{1/2}_{X_iX_i} v_i\rangle_{\mathcal{F}_i}. \end{aligned}$$

By assumption (**A5**), the operator $I_{\mathcal{F}_i} - C_{X_iX_\mathbf{J}}C_{X_\mathbf{J}X_\mathbf{J}}^{-1}C_{X_\mathbf{J}X_i}$ is lower bounded by a strictly positive constant times the identity matrix, and thus, since $\Sigma_{X_iX_i}^{1/2}v_i \neq 0$, we have $\mathbb{E}E_n^2 > 0$. This implies that $n^{1/2}\langle v_i, \hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_\mathbf{J}}\hat{f}_\mathbf{J}\rangle$ converges to a normal distribution with strictly positive variance. Thus the probability $\mathbb{P}\left(n^{1/2}\langle v_i, \hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_\mathbf{J}}\hat{f}_\mathbf{J}\rangle_{\mathcal{F}_i} \geqslant d_i\|\hat{f}\|_d\|v_i\|_{\mathcal{F}_i} + 1\right)$ converges to a strictly positive limit (note that $\|\hat{f}\|_d$ can be replaced by $\|\mathbf{f}\|_d$ without changing the result). Since $\mu_n n^{1/2} \to \mu_0 < \infty$, this implies that

$$\mathbb{P}\left(\mu_n^{-1}\langle v_i, \hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_\mathbf{J}}\hat{f}_\mathbf{J}\rangle_{\mathcal{F}_i} > d_i\|\hat{f}\|_d\|v_i\|_{\mathcal{F}_i}\right)$$

is asymptotically strictly positive (i.e., has a strictly positive $\liminf$). Thus the optimality condition (14) is not satisfied with non vanishing probability, which is a contradiction and proves the proposition.

■

We now go back to the proof of Theorem 12. We prove by contradiction, by assuming that there exists $i \in \mathbf{J}^c$ such that

$$\frac{1}{d_i}\left\|\Sigma_{X_iX_i}^{1/2}C_{X_iX_\mathbf{J}}C_{X_\mathbf{J}X_\mathbf{J}}^{-1}\operatorname{Diag}(d_j/\|\mathbf{f}_j\|_{\mathcal{F}_j})\mathbf{g}_\mathbf{J}\right\|_{\mathcal{F}_i} > 1.$$

Since with probability tending to one $J(\hat{f}) = \mathbf{J}$, with probability tending to one, we have from optimality condition (15), and the usual line of arguments (see Eq. (26) in Appendix B.2) that for every $i \in \mathbf{J}^c$:

$$\begin{aligned}\hat{\Sigma}_{X_iY} - \hat{\Sigma}_{X_iX_\mathbf{J}}\hat{f}_\mathbf{J} &= \mu_n\hat{\Sigma}_{X_iX_\mathbf{J}}\left(\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}} + \mu_n D_n\right)^{-1}D_n\mathbf{f} \\ &\quad + \hat{\Sigma}_{X_i\varepsilon} - \hat{\Sigma}_{X_iX_\mathbf{J}}\left(\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}} + \mu_n D_n\right)^{-1}\hat{\Sigma}_{X_\mathbf{J}\varepsilon},\end{aligned}$$

where $D_n = \|\hat{f}\|_d \operatorname{Diag}(d_j/\|\hat{f}_j\|)$. Following the same argument as in the proof of Theorem 11, (and because $\mu_n n^{1/2} \to +\infty$ as a consequence of Proposition 26), the first term in the last expression (divided by μ_n) converges to

$$v_i = \Sigma_{X_iX_i}^{1/2}C_{X_iX_\mathbf{J}}C_{X_\mathbf{J}X_\mathbf{J}}^{-1}\|\mathbf{f}\|_d \operatorname{Diag}(d_j/\|\mathbf{f}_j\|_{\mathcal{F}_j})\mathbf{g}_\mathbf{J}$$

By assumption $\|v_i\| > d_i\|\mathbf{f}\|_d$. We have the second term:

$$\begin{aligned}&\hat{\Sigma}_{X_i\varepsilon} - \hat{\Sigma}_{X_iX_\mathbf{J}}\left(\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}} + \mu_n\|\hat{f}\|_d \operatorname{Diag}(d_j/\|\hat{f}_j\|_{\mathcal{F}_j})\right)^{-1}\hat{\Sigma}_{X_\mathbf{J}\varepsilon} \\ = \;& O_p(n^{-1/2}) - \hat{\Sigma}_{X_iX_\mathbf{J}}\left(\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}} + \mu_n\|\mathbf{f}\|_d \operatorname{Diag}(d_j/\|\mathbf{f}_j\|_{\mathcal{F}_j})\right)^{-1}\hat{\Sigma}_{X_\mathbf{J}\varepsilon} + O_p(n^{-1/2}).\end{aligned}$$

The remaining term can be bounded as follows (with $D = \|\mathbf{f}\|_d \operatorname{Diag}(d_j/\|\mathbf{f}_j\|_{\mathcal{F}_j})$):

$$\begin{aligned}&\mathbb{E}\left(\left\|\hat{\Sigma}_{X_iX_\mathbf{J}}\left(\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}} + \mu_n D\right)^{-1}\hat{\Sigma}_{X_\mathbf{J}\varepsilon}\right\|_{\mathcal{F}_i}^2 | \bar{X}\right) \\ \leqslant\;& \frac{\sigma_{\max}^2}{n}\operatorname{tr}\hat{\Sigma}_{X_iX_\mathbf{J}}\left(\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}} + \mu_n D\right)^{-1}\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}}\left(\hat{\Sigma}_{X_\mathbf{J}X_\mathbf{J}} + \mu_n D\right)^{-1}\hat{\Sigma}_{X_\mathbf{J}X_i} \\ \leqslant\;& \frac{\sigma_{\max}^2}{n}\operatorname{tr}\hat{\Sigma}_{X_iX_i},\end{aligned}$$

which implies that the full expectation is $O(n^{-1})$ (because our operators are trace-class, that is, have finite trace). Thus the remaining term is $O_p(n^{-1/2})$ and thus negligible compared to μ_n, therefore $\frac{1}{\mu_n \|\hat{f}\|_d}\left(\hat{\Sigma}_{X_i Y} - \hat{\Sigma}_{X_i X_J}\hat{f}_J\right)$ converges in probability to a limit which is of norm strictly greater than d_i. Thus there is a non vanishing probability of being strictly larger than d_i, which implies that with non vanishing probability, the optimality condition (14) is not satisfied, which is a contradiction. This concludes the proof.

C.4 Proof of Proposition 15

Note that the estimator defined in Eq. (20) is exactly equal to

$$\left\|\hat{\Sigma}_{X_i X_J}(\hat{\Sigma}_{X_J X_J} + \kappa_n I)^{-1}\operatorname{Diag}(d_j/\|(\hat{f}^{LS}_{\kappa_n})_j\|_{\mathcal{F}_j})(\hat{f}^{LS}_{\kappa_n})_J\right\|_{\mathcal{F}_i}.$$

Using Proposition 17 and the arguments from Appendix C.2 by replacing $\tilde{f}$ by $\hat{F}_{LS}$, we get the consistency result.

C.5 Range Condition of Covariance Operators

We denote by $C(q)$ the convolution operator by q on the space of real functions on $\mathbb{R}^p$ and $T(p)$ the pointwise multiplication by $p(x)$. In this appendix, we look at different Hilbertian products of functions on $\mathbb{R}^p$, we use the notations $\langle\cdot,\cdot\rangle_{\mathcal{F}}$ and $\langle\cdot,\cdot\rangle_{L^2(p_X)}$ and $\langle\cdot,\cdot\rangle_{L^2(\mathbb{R}^p)}$ for the dot products in the RKHS $\mathcal{F}$, the space $L^2(p_X)$ of square integrable functions with respect to $p(x)dx$, and the space $L^2(\mathbb{R}^p)$ of square integrable functions with respect to the Lebesgue measure. With our assumptions, for all $\tilde{f},\tilde{g}\in L^2(\mathbb{R}^p)$, we have:

$$\langle\tilde{f},\tilde{g}\rangle_{L^2} = \langle C(q)^{1/2}\tilde{f}, C(q)^{1/2}\tilde{g}\rangle_{\mathcal{F}}.$$

Denote by $\{\lambda_k\}_{k\geq 1}$ and $\{e_k\}_{k\geq 1}$ the positive eigenvalues and the eigenvectors of the covariance operator Σ_{XX}, respectively. Note that since $p_X(x)$ was assumed to be strictly positive, all eigenvalues are strictly positive (the RKHS cannot contain any non zero constant functions on $\mathbb{R}^p$). For $k\geqslant 1$, set $f_k = \lambda_k^{-1/2}(e_k - \int_{\mathbb{R}^p} e_k(x)p_X(x)dx)$. By construction, for any $k,\ell\geqslant 1$,

$$\begin{aligned}\lambda_k\delta_{k,\ell} = \langle e_k, \Sigma e_\ell\rangle_{\mathcal{F}} &= \int_{\mathbb{R}^p} p_X(x)(e_k - \textstyle\int_{\mathbb{R}^p} e_k(x)p_X(x)dx)(e_\ell - \int_{\mathbb{R}^p} e_\ell(x)p_X(x)dx)dx \\ &= \lambda_k^{1/2}\lambda_\ell^{1/2}\int_{\mathbb{R}^p} p_X(x)f_k(x)f_\ell(x)dx = \lambda_k^{1/2}\lambda_\ell^{1/2}\langle f_k, f_\ell\rangle_{L^2(p_X)}.\end{aligned}$$

Thus $\{f_k\}_{k\geqslant 1}$ is an orthonormal sequence in $L^2(p_X)$. Let $f = C(q)g$ for $g\in L^2(\mathbb{R}^p)$ such that $\int_{\mathbb{R}^p} g(x)dx = 0$. Note that f is in the range of $\Sigma_{XX}^{1/2}$ if and only if $\langle f,\Sigma^{-1}f\rangle_{\mathcal{F}}$ is finite. We have:

$$\begin{aligned}\langle f,\Sigma^{-1}f\rangle_{\mathcal{F}} = \sum_{p=1}^{\infty}\lambda_p^{-1}\langle e_p,f\rangle_{\mathcal{F}}^2 = \sum_{p=1}^{\infty}\lambda_p^{-1}\langle e_p,g\rangle_{L^2(\mathbb{R}^p)}^2 &= \sum_{p=1}^{\infty}\lambda_p^{-1}\left(\int_{\mathbb{R}^p} g(x)e_p(x)dx\right)^2 \\ &= \sum_{p=1}^{\infty}\langle p_X^{-1}g, f_p\rangle_{L^2(p_X)}^2 \leqslant \|p_X^{-1}g\|_{L^2(p_X)}^2 = \int_{\mathbb{R}^p}\frac{g^2(x)}{p_X(x)}dx,\end{aligned}$$

because $\{f_k\}_{k\geqslant 1}$ is an orthonormal sequence in $L^2(p_X)$. This concludes the proof.

Appendix D. Proof of Results on Adaptive Group Lasso

In this appendix, we give proofs of the consistency of the adaptive group Lasso procedures.

D.1 Proof of Theorem 16

We define $\tilde{w}$ as the minimizer of the same cost function restricted to $w_{\mathbf{J}^c} = 0$. Because $\hat{w}^{LS}$ is consistent, the norms of $\hat{w}_j^{LS}$ for $j \in \mathbf{J}$ are bounded away from zero, and we get from standard results on M-estimation (Van der Vaart, 1998) the normal limit distribution with given covariance matrix if $\mu_n \ll n^{-1/2}$.

Moreover, the patterns of zeros (which is obvious by construction of $\tilde{w}$) converges in probability. What remains to be shown is that with probability tending to one, $\tilde{w}$ is optimal for the full problem. We just need to show that with probability tending to one, for all $i \in \mathbf{J}^c$,

$$\|\hat{\Sigma}_{X_i\varepsilon} - \hat{\Sigma}_{X_iX_\mathbf{J}}(\tilde{w}_\mathbf{J} - w_\mathbf{J})\| \leqslant \mu_n \|\tilde{w}\|_d \|\hat{w}_i^{LS}\|^{-\gamma}. \tag{27}$$

Note that $\|\tilde{w}\|_d$ converges in probability to $\|\mathbf{w}\|_d > 0$. Moreover, $\|\hat{w}_i^{LS} - \mathbf{w}_i\| = O_p(n^{-1/2})$. Thus, if $i \in \mathbf{J}^c$, that is, if $\mathbf{f}_i = 0$, then $\|\hat{w}_i^{LS}\| = O_p(n^{-1/2})$. The left hand side in Eq. (27) is thus upper bounded by $O_p(n^{-1/2})$ while the right hand side is lower bounded asymptotically by $\mu_n n^{\gamma/2}$. Thus if $n^{-1/2} = o(\mu_n n^{\gamma/2})$, then with probability tending to one we get the correct optimality condition, which concludes the proof.

D.2 Proof of Proposition 17

We have:

$$\hat{f}_{\kappa_n}^{LS} = \left(\hat{\Sigma}_{XX} + \kappa_n I_\mathcal{F}\right)^{-1} \hat{\Sigma}_{XY},$$

and thus:

$$\begin{aligned}
\hat{f}_{\kappa_n}^{LS} - \mathbf{f} &= \left(\hat{\Sigma}_{XX} + \kappa_n I_\mathcal{F}\right)^{-1} \hat{\Sigma}_{XX}\mathbf{f} - \mathbf{f} + \left(\hat{\Sigma}_{XX} + \kappa_n I_\mathcal{F}\right)^{-1} \hat{\Sigma}_{X\varepsilon} \\
&= (\Sigma_{XX} + \kappa_n I)^{-1} \Sigma_{XX}\mathbf{f} - \mathbf{f} + O_p(n^{-1/2}\kappa_n^{-1}) \text{ from Lemma 23} \\
&= -\left(\Sigma_{XX} + \kappa_n I_\mathcal{F}\right)^{-1} \kappa_n \mathbf{f} + O_p(n^{-1/2}\kappa_n^{-1}).
\end{aligned}$$

Since $\mathbf{f} = \Sigma_{XX}^{1/2}\mathbf{g}$, we have $\| - \left(\Sigma_{XX} + \kappa_n I_\mathcal{F}\right)^{-1} \kappa_n \mathbf{f}\|_\mathcal{F}^2 \leqslant C\kappa_n \|\mathbf{g}\|_\mathcal{F}^2$, which concludes the proof.

D.3 Proof of Theorem 18

We define $\tilde{f}$ as the minimizer of the same cost function restricted to $f_{\mathbf{J}^c} = 0$. Because $\hat{f}_{n^{-1/3}}^{LS}$ is consistent, the norms of $(\hat{f}_{n^{-1/3}}^{LS})_j$ for $j \in \mathbf{J}$ are bounded away from zero, and Lemma 25 applies with $\mu_n = \mu_0 n^{-1/3}$, that is, $\tilde{f}$ converges in probability to $\mathbf{f}$ and so are the patterns of zeros (which is obvious by construction of $\tilde{f}$). Moreover, for any $\eta > 0$, from Lemma 25, we have $\|\tilde{f}_\mathbf{J} - f_\mathbf{J}\| = O_p(n^{-1/6+\eta})$ (because $\mu_n^{-1/2} + n^{-1/2}\mu_n^{-1} = O_p(n^{-1/6})$).

What remains to be shown is that with probability tending to one, $\tilde{f}$ is optimal for the full problem. We just need to show that with probability tending to one, for all $i \in \mathbf{J}^c$,

$$\|\hat{\Sigma}_{X_i\varepsilon} - \hat{\Sigma}_{X_iX_\mathbf{J}}(\tilde{f}_\mathbf{J} - f_\mathbf{J})\| \leqslant \mu_n \|\tilde{f}\|_d \|(\hat{f}_{n^{-1/3}}^{LS})_i\|_{\mathcal{F}_i}^{-\gamma}. \tag{28}$$

Note that $\|\tilde{f}\|_d$ converges in probability to $\|\mathbf{f}\|_d > 0$. Moreover, by Proposition 17, $\|(\hat{f}_{n^{-1/3}}^{LS})_i - \mathbf{f}_i\| = O_p(n^{-1/6})$. Thus, if $i \in \mathbf{J}^c$, that is, if $\mathbf{f}_i = 0$, then $\|(\hat{f}_{n^{-1/3}}^{LS})_i\|_{\mathcal{F}_i} = O_p(n^{-1/6})$. The left hand side in

Eq. (28) is thus upper bounded by $O_p(n^{-1/2}+n^{-1/6+\eta})$ while the right hand side is lower bounded asymptotically by $n^{-1/3}n^{\gamma/6}$. Thus if $-1/6+\eta<-1/3+\gamma/6$, then with probability tending to one we get the correct optimality condition. As soon as $\gamma>1$, we can find η small enough and strictly positive, which concludes the proof.

Appendix E. Gaussian Kernels and Gaussian Variables

In this section, we consider $X\in\mathbb{R}^m$ with normal distribution with zero mean and covariance matrix S. We also consider Gaussian kernels $k_j(x_j,x'_j)=\exp(-b_i(x_j-x'_j)^2)$ on each of its component. In this situation, we can find orthonormal basis of the Hilbert spaces $\mathcal{F}_j$ where we can compute the coordinates of all covariance operators. This thus allows to check conditions (16) or (17) without using sampling.

We consider the eigenbasis of the non centered covariance operators on each $\mathcal{F}_j$, $j=1,\ldots,m$, which is equal to (Zhu et al., 1998):

$$e_k^j(x_j)=(\lambda_k^j)^{1/2}\left(\frac{c_j^{1/2}}{a_j^{1/2}2^k k!}\right)^{1/2}e^{-(c_j-a_j)u^2}H_k((2c_j)^{1/2}x_j)$$

with eigenvalues $\lambda_k^j=\left(\frac{2a_j}{A_j}\right)^{1/2}(B_j)^k$, where $a_i=1/4S_{ii}$, $c_j=(a_j^2+2a_jb_j)^{1/2}$, $A_j=a_j+b_j+c_j$ and $B_j=b_j/A_j$, and H_k is the k-th Hermite polynomial.

We can then compute all required expectations as follows (note that by definition we have $\mathbb{E}e_k^j(X_j)^2=\lambda_k^i$):

$$\begin{aligned}\mathbb{E}e_{2k+1}^j(X_j) &= 0\\ \mathbb{E}e_{2k}^j(X_j) &= \left(\lambda_{2k}^j\frac{2a_j^{1/2}c_j^{1/2}}{(a_j+c_j)}\binom{2k}{k}\right)^{1/2}\left(\frac{c_j-a_j}{2(c_j+a_j)}\right)^k\end{aligned}$$

$$\mathbb{E}e_k^j(X_j)e_\ell^i(X_i)=\left(\lambda_{2k}^j\lambda_{2\ell}^i\frac{c_j^{1/2}c_i^{1/2}}{a_j^{1/2}a_i^{1/2}2^k2^\ell k!\ell!}\right)^{1/2}\frac{(S_{ii}S_{jj}-S_{ij}^2)^{-1/2}}{4\pi c_i^{1/2}c_j^{1/2}}D_{k\ell}(Q_{ij}),$$

where $Q_{ij}=\begin{pmatrix}\frac{1}{2}(1-a_i/c_i) & 0\\ 0 & \frac{1}{2}(1-a_j/c_j)\end{pmatrix}+\frac{1}{4}\begin{pmatrix}S_{ii}c_i & S_{ij}c_i^{1/2}c_j^{1/2}\\ S_{ij}c_i^{1/2}c_j^{1/2} & S_{jj}c_j\end{pmatrix}^{-1}$ and

$$D_{k\ell}(Q)=\int_{\mathbb{R}^2}\exp\left[-\begin{pmatrix}u\\ v\end{pmatrix}^\top Q\begin{pmatrix}u\\ v\end{pmatrix}\right]H_k(u)H_\ell(v)dudv,$$

for any positive matrix Q. For any given Q, $D_{k\ell}(Q)$ can be computed exactly by using a singular value decomposition of Q and the appropriate change of variables.[6]

6. Matlab code to compute $D_{k\ell}(Q)$ can be downloaded from the author's webpage.

References

F. R. Bach. Bolasso: model consistent Lasso estimation through the bootstrap. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008a.

F. R. Bach. Consistency of trace norm minimization. *Journal of Machine Learning Research*, 9: 1019–1048, 2008b.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004a.

F. R. Bach, R. Thibaux, and M. I. Jordan. Computing regularization paths for learning multiple kernels. In *Advances in Neural Information Processing Systems 17*, 2004b.

C. Baker. Joint measures and cross-covariance operators. *Transactions of the American Mathematical Society*, 186:273–289, 1973.

A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer Academic Publishers, 2003.

O. Bousquet and D. J. L. Herrmann. On the complexity of learning the kernel matrix. In *Advances in Neural Information Processing Systems 17*, 2003.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, 2003.

P. Brémaud. *Markov Chains, Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer-Verlag, 1999.

H. Brezis. *Analyse Fonctionelle*. Masson, 1980.

A. Caponnetto and E. de Vito. Fast rates for regularized least-squares algorithm. Technical Report 248/AI Memo 2005-013, CBCL, Massachusetts Institute of Technology, 2005.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1), 2002.

R. Durrett. *Probability: Theory and Examples*. Duxbury Press, third edition, 2004.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32: 407, 2004.

W. Fu and K. Knight. Asymptotics for Lasso-type estimators. *Annals of Statistics*, 28(5):1356–1378, 2000.

K. Fukumizu, F. R. Bach, and M. I. Jordan. Dimensionality reduction for supervised learning with reproducing kernel Hilbert spaces. *Journal of Machine Learning Research*, 5:73–99, 2004.

K. Fukumizu, F. R. Bach, and A. Gretton. Statistical convergence of kernel canonical correlation analysis. *Journal of Machine Learning Research*, 8(8), 2007.

A. Gretton, R. Herbrich, A. Smola, O. Bousquet, and B. Schölkopf. Kernel methods for measuring independence. *Journal of Machine Learning Research*, 6:2075–2129, 12 2005.

Z. Harchaoui and F. R. Bach. Image classification with segmentation graph kernels. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models*. Chapman & Hall, 1990.

A. Juditsky and A. Nemirovski. Functional aggregation for nonparametric regression. *Annals of Statistics*, 28(3):681–712, 2000.

G. R. G. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20:2626–2635, 2004a.

G. R. G. Lanckriet, N. Cristianini, L. El Ghaoui, P. Bartlett, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004b.

M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lébret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

J. McAuley, J. Ming, D. Stewart, and P. Hanna. Subband correlation and robust speech recognition. *IEEE Transactions on Speech and Audio Processing*, 13(5):956–964, 2005.

L. Meier, S. van de Geer, and P. Bühlmann. The group Lasso for logistic regression. Technical Report 131, Eidgenöossische Technische Hochschule (ETH), Zürich, Switzerland, 2006.

N. Meinshausen and B. Yu. Lasso-type recovery of sparse representations for high-dimensional data. Technical Report 720, Departement of Statisics, UC Berkeley, 2006.

M. R. Osborne, B. Presnell, and B. A. Turlach. On the lasso and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000.

A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2007.

P. Ravikumar, H. Liu, J. Lafferty, and L. Wasserman. SpAM: Sparse additive models. In *Advances in Neural Information Processing Systems 22*, 2008.

A. Renyi. On Measures of Dependence. *Acta Mathematica Academy Sciences Hungary*, 10:441–451, 1959.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2001.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 07 2006.

I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of The Royal Statistical Society Series B*, 58(1):267–288, 1996.

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems*. V. H. Winston and Sons, 1997.

A. W. Van der Vaart. *Asymptotic Statistics*. Cambridge Univ. Press, 1998.

M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, 2007.

G. Wahba. *Spline Models for Observational Data*. SIAM, 1990.

M. J. Wainwright. Sharp thresholds for noisy and high-dimensional recovery of sparsity using ℓ_1-constrained quadratic programming. Technical Report 709, Department of Statistics, UC Berkeley, 2006.

Q. Wu, Y. Ying, and D.-X. Zhou. Multi-kernel regularized classifiers. *Journal of Complexity*, 23 (1):108–134, 2007.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of The Royal Statistical Society Series B*, 68(1):49–67, 2006.

M. Yuan and Y. Lin. On the non-negative garrotte estimator. *Journal of The Royal Statistical Society Series B*, 69(2):143–161, 2007.

P. Zhao and B. Yu. On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.

D. Zhou and C. J. C. Burges. Spectral clustering and transductive learning with multiple views. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2007.

H. Zhu, C K. I. Williams, R. Rohwer, and M. Morciniec. Gaussian regression and optimal finite dimensional linear models. In *Neural Networks and Machine Learning*. Springer-Verlag, 1998.

H. Zou. The adaptive Lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429, December 2006.

Journal of Machine Learning Research 9 (2008) 1227-1267 Submitted 5/07; Revised 11/07; Published 6/08

Maximal Causes for Non-linear Component Extraction

Jörg Lücke LUCKE@GATSBY.UCL.AC.UK
Maneesh Sahani MANEESH@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit
University College London
17 Queen Square
London WC1N 3AR, UK

Editor: Yoshua Bengio

Abstract

We study a generative model in which hidden causes combine competitively to produce observations. Multiple active causes combine to determine the value of an observed variable through a max function, in the place where algorithms such as sparse coding, independent component analysis, or non-negative matrix factorization would use a sum. This max rule can represent a more realistic model of non-linear interaction between basic components in many settings, including acoustic and image data. While exact maximum-likelihood learning of the parameters of this model proves to be intractable, we show that efficient approximations to expectation-maximization (EM) can be found in the case of sparsely active hidden causes. One of these approximations can be formulated as a neural network model with a generalized softmax activation function and Hebbian learning. Thus, we show that learning in recent softmax-like neural networks may be interpreted as approximate maximization of a data likelihood. We use the bars benchmark test to numerically verify our analytical results and to demonstrate the competitiveness of the resulting algorithms. Finally, we show results of learning model parameters to fit acoustic and visual data sets in which max-like component combinations arise naturally.

Keywords: component extraction, maximum likelihood, approximate EM, competitive learning, neural networks

1. Introduction

In recent years, algorithms such as independent components analysis (ICA; Comon, 1994; Bell and Sejnowski, 1997), sparse coding (SC; Olshausen and Field, 1996), and non-negative matrix factorization (NMF; Lee and Seung, 1999) have been used to describe the statistics of the natural environment, and the components extracted by these methods have been linked to sensory neuronal response properties. Stated in the language of probabilistic generative models (see, e.g., Dayan and Abbott, 2001; Rao et al., 2002) these systems describe sensory data as a linear superposition of learned components. For many types of data, including images, this assumed linear cooperation between generative causes is unrealistic. Alternative, more competitive generative models have also been proposed: for instance, Saund (1995) suggests a model in which hidden causes are combined by a noisy-or rule, while Dayan and Zemel (1995) suggest a yet more competitive scheme. Here, we formulate an extreme case of competition, in which the strongest generative influence on an observed variable (e.g., an image pixel) alone determines its value. Such a rule has the property of selecting, for each observed variable, a single generative cause to determine that variable's value.

This form of combination emerges naturally in the context of spectrotemporal masking in mixed audio signals. For image data, occlusion leads to a different combination rule, but one that shares the selection property in that, under constant lighting conditions, the appearance of each observed pixel is determined by a single object.

In parallel to this development of generative approaches, a number of artificial neural network architectures have been designed to tackle the problem of non-linear component extraction, mostly in artificial data (e.g., Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004; Lücke and Bouecke, 2005; Spratling, 2006), although sometimes in natural images (e.g., Harpur and Prager, 1996; Charles et al., 2002; Lücke, 2007). These models often perform quite well with respect to various benchmark tests. However, the relationship between them and the density models that are implicit or explicit in the generative approach has not, thus far, been made clear. We show here that inference and learning in a restricted form of our novel generative model correspond closely in form to the processing and plasticity rules used in such neural network approaches, thus bringing together these two disparate threads of investigation.

The organization of the remainder of this article is as follows. In Section 2 we define the novel generative model and then proceed to obtain the associated parameter update rules in Section 3. In Section 4 we derive computationally efficient approximations to these update rules, in the context of sparsely active hidden causes—that is, when a small number of hidden causes generally suffices to explain the data. In Section 5 we relate a restricted form of the generative model to neural network learning rules with Hebbian plasticity and divisive normalization. Results of numerical experiments in Section 6 show the component extraction performance of the generative schemes as well as a comparison to other algorithms. Finally, in Section 7, we discuss our analytical and numerical results.

2. A Generative Model with Maximum Non-linearity

We consider a generative model for D observed variables y_d, $(d = 1,\ldots,D)$, in which H hidden binary causes s_h, $(h = 1,\ldots,H)$, each taking the value 0 or 1, *compete* to determine the value of each observation (see Figure 1). Associated with each pair (s_h, y_d), is a weight W_{hd}. Given a set of active causes (i.e., those taking the value 1), the distribution of y_d is determined by the *largest* of the weights associated with the active causes and y_d.

Much of our discussion will apply generally to all models of this causal structure, irrespective of the details of the distributions involved. For concreteness, however, we focus on a particular choice, in which the hidden variables are drawn from a multivariate Bernoulli distribution; and the observed variables are non-negative, integer-valued and, given the causes, conditionally independent and Poisson-distributed. Thus, collecting all the causes into a single binary vector $\vec{s} \in \{0,1\}^H$, and all the observed variables into an integer vector $\vec{y} \in \mathbb{Z}_+^D$ we have:

$$p(\vec{s}\,|\,\vec{\pi}) \;=\; \prod_{h=1}^{H} p(s_h\,|\,\pi_h), \quad p(s_h\,|\,\pi_h) = \pi_h^{s_h}(1-\pi_h)^{1-s_h}, \tag{1}$$

$$p(\vec{y}\,|\,\vec{s},W) \;=\; \prod_{d=1}^{D} p(y_d\,|\,\overline{W}_d(\vec{s},W)), \quad p(y_d\,|\,w) = \frac{w^{y_d}}{y_d!}e^{-w}. \tag{2}$$

Here, $\vec{\pi} \in [0,1]^H$ parameterizes the prior distribution on $\vec{s}$, while the weight matrix $W \in \mathrm{R}^{H\times D}$ parameterizes the influence of the hidden causes on the distribution of $\vec{y}$. It will be convenient to

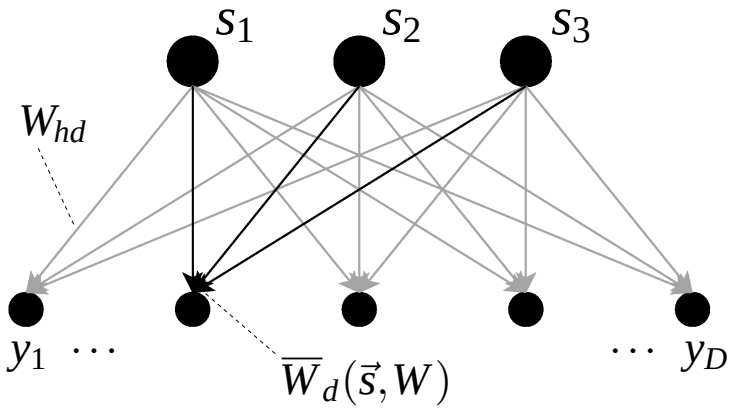


Figure 1: A generative model with $H = 3$ hidden variables and $D = 5$ observed variables. The values y_d of the observed variables are conditionally independent given the values $\vec{s}$ of the hidden variables. The value y_d is drawn from a distribution which is determined by the parameters W_{1d}, W_{2d}, and W_{3d}. For a given binary vector $\vec{s}$ these parameters combine competitively according to the function $\overline{W}_d(\vec{s},W) = \max_h\{s_h W_{hd}\}$.

group these parameters together into $\Theta = (\vec{\pi}, W)$. The function $\overline{W}_d(\vec{s},W)$ in (2) gives the *effective weight* on y_d, resulting from a particular pattern of causes $\vec{s}$. Thus, in the model considered here,

$$\overline{W}_d(\vec{s},W) \quad = \quad \max_h\{s_h W_{hd}\}\,. \tag{3}$$

It is useful to place the model (1)–(3) in context. Models of this general type, in which the observations are conditionally independent of one another given a set of hidden causes, are widespread. They underlie algorithms such as ICA, SC, principal components analysis (PCA), factor analysis (see, e.g., Everitt, 1984), and NMF. In these five cases, and indeed in the majority of such models studied, the effective weights $\overline{W}_d(\vec{s},W)$ are formed by a linear combination of all the weights that link hidden variables to the observation; that is, $\overline{W}_d(\vec{s},W) = \sum_h s_h W_{hd}$. Some other models, notably those of Saund (1995) and Dayan and Zemel (1995), have implemented more competitive combination rules, where larger individual weights dominate the effective combination. The present model takes this competition to an extreme, so that only the single largest weight (amongst those associated with active hidden variables) determines the output distribution. Thus, where ICA, PCA, SC, or NMF use a sum, we use a max. We refer to this new generative model as the Maximal Causes Analysis (MCA) model.

Figure 2 illustrates the difference between linear superposition and competitive combination using (3). Let us suppose that noise-free observations are generated by causes in the form of horizontal and vertical objects with the same gray-value, on a dark (black) background (see Figure 2). If these objects occlude one-another, they may generate an observed image such as that illustrated in Figure 2B. However, if we were to use the actual causes and weights in Figure 2A, but instead combine them linearly, we would obtain the (different) input pattern of Figure 2C. In this case, competitive combination using the max-rule of Equation (3) would result in the correct pattern. This is not, of course, generally true, but for monochrome objects with small variations in their gray-values it

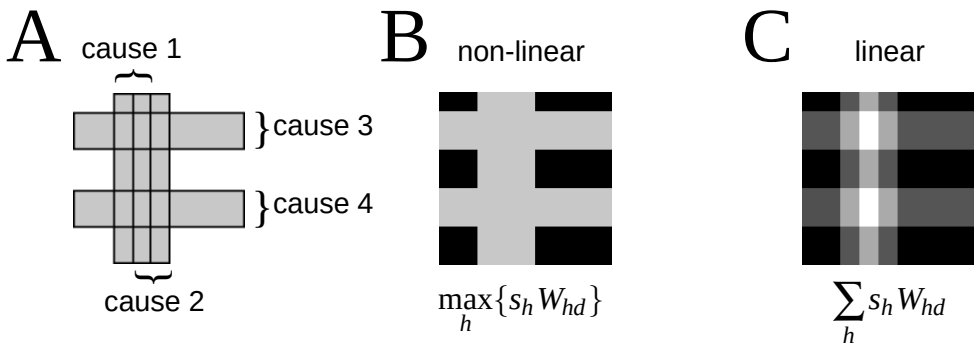


Figure 2: An illustration of non-linear versus linear combination of hidden causes. **A** Four examples of hidden causes with gray-value 200. **B** The input image that may result if sources occlude one another. In this case, the correct function $\overline{W}_d(\vec{s},W)$ (see Figure 1) to combine the hidden causes is the max-operation. **C** The input image that results if the four causes combine linearly (gray-values are scaled to fill the interval [0,255]). For **C**, the correct function $\overline{W}_d(\vec{s},W)$ is linear super-position.

holds approximately. More generally, the maximum combination rule is always closer to the result of occlusion than is the simple sum implied by models such as ICA.

As stated above, although in this paper we focus on the specific distributions given in (1) and (2), much of the analytical treatment is independent of these specific choices. Thus, update rules for learning the weights W from data will be derived in a general form, that can accommodate alternative, non-factored distributions for the binary hidden variables. This general form is also preserved if the Poisson distribution is replaced, for example, by a Gaussian. Poisson variability represents a reasonable choice for the non-negative data considered in this paper, and resembles the cost function introduced by Lee and Seung (1999) for NMF.

3. Maximum Likelihood

Given a set of observed data vectors $Y=\{\vec{y}^{(n)}\}_{n=1,\ldots,N}$, taken to be generated independently from a stationary process, we seek parameter values $\Theta^* = (\vec{\pi}^*, W^*)$ that maximize the likelihood of the data under the generative model of Equations (1) to (3):

$$\Theta^* \;=\; \text{argmax}_\Theta\{\mathcal{L}(\Theta)\} \quad \text{with} \quad \mathcal{L}(\Theta) = \log\Big(p(\vec{y}^{(1)},\ldots,\vec{y}^{(N)}\,|\,\Theta)\Big)\,.$$

We use Expectation-Maximization (EM; Dempster et al. 1977; see also Neal and Hinton 1998, for the formulation that appears here) to maximize the likelihood in this latent variable model. To do so, we introduce the free-energy $\mathcal{F}(\Theta,q)$—a data-dependent function of the parameters Θ and an unknown distribution $q(\vec{s}^{(1)},\ldots,\vec{s}^{(N)})$ over the hidden data or variables—that is always equal to or less than the likelihood evaluated at the same parameter values. For independently generated data vectors $\vec{y}^{(n)}$, the distribution q may be taken (without loss of generality) to factor over the hidden

vectors $q(\vec{s}^{(1)},\ldots,\vec{s}^{(N)}) = \prod_n q_n(\vec{s}^{(n)})$. Then the free-energy is defined as:

$$\mathcal{F}(\Theta,q) = \sum_{n=1}^{N}\left[\sum_{\vec{s}} q_n(\vec{s})\ \left[\log\left(p(\vec{y}^{(n)}\,|\,\vec{s},\Theta)\right)+\log\left(p(\vec{s}\,|\,\Theta)\right)\right]\right]+H(q) \leq \mathcal{L}(\Theta), \tag{4}$$

where $H(q) = \sum_n H(q_n(\vec{s})) = -\sum_n \sum_{\vec{s}} q_n(\vec{s})\log(q_n(\vec{s}))$ is the Shannon entropy of q. The iterations of EM alternately increase $\mathcal{F}$ with respect to the distributions q_n while holding Θ fixed (the E-step), and with respect to Θ while holding the q_n fixed (the M-step). Thus, if we consider a pair of steps beginning from parameters Θ', the E-step first finds new distributions q_n that depend on Θ' and the observations $\vec{y}^{(n)}$, which we write as $q_n(\vec{s};\Theta')$. Ideally, these distributions maximize $\mathcal{F}$ for fixed Θ', in which case it can be shown that $q_n(\vec{s};\Theta') = p(\vec{s}\,|\,\vec{y}^{(n)},\Theta')$ and $\mathcal{F}(\Theta',q_n(\vec{s};\Theta')) = \mathcal{L}(\Theta')$ (Neal and Hinton, 1998). In practice, computation of this exact posterior may be intractable, and it is often replaced by an approximation. After choosing the q_n's in the E-step, we maximize $\mathcal{F}$ with respect to Θ in the M-step while holding the q_n distributions fixed. Thus the free-energy can be re-written in terms of Θ and Θ':

$$\mathcal{F}(\Theta,\Theta') \quad = \quad \sum_{n=1}^{N}\left[\sum_{\vec{s}} \quad q_n(\vec{s};\Theta') \quad \left[\log\left(p(\vec{y}^{(n)}\,|\,\vec{s},\Theta)\right)+\log\left(p(\vec{s}\,|\,\Theta)\right)\right]\right] \ + \ H(\Theta'). \tag{5}$$

where $H(\Theta') = \sum_n H(q_n(\vec{s};\Theta'))$. A necessary condition to achieve this maximum with respect to $W_{id} \in \Theta$, is that (see Appendix A for details):

$$\frac{\partial}{\partial W_{id}}\mathcal{F}(\Theta,\Theta') = \sum_n\sum_{\vec{s}} q_n(\vec{s};\Theta')\left(\frac{\partial}{\partial W_{id}}\overline{W}_d(\vec{s},W)\right)\frac{y_d^{(n)}-\overline{W}_d(\vec{s},W)}{\overline{W}_d(\vec{s},W)} \quad \stackrel{!}{=} \quad 0. \tag{6}$$

Unfortunately, under the max-combination rule of Equation (3), $\overline{W}_d$ is not differentiable. Instead, we define a smooth function $\overline{W}_d^{\rho}$ that converges to $\overline{W}_d$ as ρ approaches infinity:

$$\overline{W}_d^{\rho}(\vec{s},W) \quad := \quad \left(\sum_{h=1}^{H}(s_h W_{hd})^{\rho}\right)^{\frac{1}{\rho}} \ \Rightarrow \ \lim_{\rho\to\infty}\overline{W}_d^{\rho}(\vec{s},W) = \overline{W}_d(\vec{s},W), \tag{7}$$

and replace the derivative of $\overline{W}_d$ by the limiting value of the derivative of $\overline{W}_d^{\rho}$, which we write as $\mathcal{A}_{id}$ (see Appendix A for details):

$$\mathcal{A}_{id}(\vec{s},W) := \lim_{\rho\to\infty}\left(\frac{\partial}{\partial W_{id}}\overline{W}_d^{\rho}(\vec{s},W)\right) = \lim_{\rho\to\infty}\frac{s_i\,(W_{id})^{\rho}}{\sum_h s_h (W_{hd})^{\rho}}. \tag{8}$$

Armed with this definition, a rearrangement of the terms in (6) yields (see Appendix A):

$$W_{id} \ = \ \frac{\sum_n \left\langle \mathcal{A}_{id}(\vec{s},W)\right\rangle_{q_n} y_d^{(n)}}{\sum_n \left\langle \mathcal{A}_{id}(\vec{s},W)\right\rangle_{q_n}}, \tag{9}$$

where $\left\langle \mathcal{A}_{id}(\vec{s},W)\right\rangle_{q_n}$ is the expectation of $\mathcal{A}_{id}(\vec{s},W)$ under the distribution $q_n(\vec{s};\Theta')$:

$$\left\langle \mathcal{A}_{id}(\vec{s},W)\right\rangle_{q_n} = \sum_{\vec{s}} q_n(\vec{s};\Theta')\,\mathcal{A}_{id}(\vec{s},W). \tag{10}$$

Equation (9) represents a set of non-linear equations (one for each W_{id}) that defines the necessary conditions for an optimum of $\mathcal{F}$ with respect to W. The equations do not represent straightforward update rules for W_{id} because the right-hand-side does not depend only on the old values $W' \in \Theta'$. They can, however, be used as fixed-point iteration equations, by simply evaluating the derivatives $\mathcal{A}_{id}$ at W' instead of W. Although there is no guarantee that these iterations converge, if they do converge the corresponding parameters must lie at a stationary point of the free-energy. Numerical experiments described later confirm that this fixed-point approach is, in fact, robust and convergent. Note that the denominator in (9) vanishes only if $q_n(\vec{s};\Theta')\,\mathcal{A}_{id}(\vec{s},W) = 0$ for all $\vec{s}$ and n (assuming positive weights), in which case (6) is already satisfied, and no update of W is required.

Thus far, we have not made explicit reference to the form of prior source distribution, and so the result of Equation (9) is independent of this choice. For our chosen Bernoulli distribution (1), the M-step is obtained by setting the derivative of $\mathcal{F}$ with respect to π_i to zero, giving (after rearrangement):

$$\pi_i \;=\; \frac{1}{N}\sum_n \langle s_i \rangle_{q_n}, \quad \text{with } \langle s_i \rangle_{q_n} = \sum_{\vec{s}} q_n(\vec{s};\Theta')\, s_i\,. \tag{11}$$

Parameter values that satisfy Equations (9) and (11), maximize the free-energy given the distributions $q_n = q_n(\vec{s};\Theta')$. As stated before, the optimum with respect to q (and therefore, exact optimization of the likelihood, since the optimal setting of q forces the free-energy bound to be tight) is obtained by setting the q_n to the posterior distributions:

$$q_n(\vec{s};\Theta') \;=\; p(\vec{s}\,|\,\vec{y}^{(n)},\Theta') \;=\; \frac{p(\vec{s},\vec{y}^{(n)}\,|\,\Theta')}{\sum_{\tilde{\vec{s}}} p(\tilde{\vec{s}},\vec{y}^{(n)}\,|\,\Theta')}, \tag{12}$$

where $p(\vec{s},\vec{y}^{(n)}\,|\,\Theta') = p(\vec{s}\,|\,\vec{\pi}')\,p(\vec{y}^{(n)}\,|\,\vec{s},W')$, and with the latter distributions given by (1) and (2), respectively.

Equations (9) to (12) thus represent a complete set of update rules for maximizing the data likelihood under the generative model. The only approximation made to this point is to use the old values W' on the right-hand-side of the M-step equation in (9). We therefore refer to this set of updates as a pseudo-exact learning rule and call the algorithm they define MCA_{ex}, with the subscript for *exact*. We will see in numerical experiments that MCA_{ex} does indeed maximize the likelihood.

4. E-Step Approximations

The computational cost of finding the exact sufficient statistics $\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n}$, with q_n equal to the posterior probability (12), is intractable in general. It grows exponentially in the smaller of the number of hidden causes H, and the number of observed variables D (see Appendix B for details). A practical learning algorithm, then, must depend on finding a computationally tractable approximation to the true expectation. One approach, a form of variational method (Jordan et al., 1999), would be to optimize the q_n within a constrained class of distributions; for example, distributions that factor over the sources s_h. Unfortunately, this conventional factoring approach provides limited benefit here, as the form of $\mathcal{A}_{id}(\vec{s},W)$ resists straightforward evaluation of the expected value with respect to the individual sources. Instead, we base our approximations on an assumption of sparsity—that only a small number of active hidden sources is needed to explain any one observed

data vector (note that sparsity here refers to the *number* of active hidden sources, rather than to their *proportion*). The resulting expressions relate to those that would be found by a variational optimization constrained to distributions that are sparse in the sense above, but are not identical. The relationship will be explored further in the Discussion.

To develop the sparse approximations, consider grouping the terms in the expected value of Equation (10) according to the number of active sources in the vector $\vec{s}$:

$$\begin{aligned}\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} &= \sum_{\vec{s}} p(\vec{s}|\vec{y}^{(n)},\Theta')\,\mathcal{A}_{id}(\vec{s},W) \qquad (13)\\ &= \sum_a p(\vec{s}_a\,|\vec{y}^{(n)},\Theta')\,\mathcal{A}_{id}(\vec{s}_a,W) + \sum_{\substack{a,b\\a<b}} p(\vec{s}_{ab}\,|\vec{y}^{(n)},\Theta')\,\mathcal{A}_{id}(\vec{s}_{ab},W) + \sum_{\substack{a,b,c\\a<b<c}} \ldots,\end{aligned}$$

where $\vec{s}_a := (0,\ldots,0,1,0,\ldots,0)$ with only $s_a = 1$
$\vec{s}_{ab} := (0,\ldots,0,1,0,\ldots,0,1,0,\ldots,0)$ with only $s_a = 1$, $s_b = 1$, $a \neq b$,
and $\vec{s}_{abc}$ etc. are defined analogously.

Note that $\mathcal{A}_{id}(\vec{0},W) = 0$ because of (7) and (8). Now, each of the conditional probabilities $p(\vec{s}|\vec{y}^{(n)},\Theta')$ implicitly contains a similar sum over $\vec{s}$ for normalization:

$$p(\vec{s}|\vec{y}^{(n)},\Theta') = \frac{1}{\mathcal{Z}}\,p(\vec{s},\vec{y}^{(n)}\,|\,\Theta'), \qquad \mathcal{Z} := \sum_{\vec{s}} p(\vec{s},\vec{y}^{(n)}\,|\,\Theta'), \qquad (14)$$

and the terms of this sum may be grouped in the same way

$$\mathcal{Z} := p(\vec{0},\vec{y}^{(n)}\,|\,\Theta') + \sum_a p(\vec{s}_a,\vec{y}^{(n)}\,|\,\Theta') + \sum_{\substack{a,b\\a<b}} p(\vec{s}_{ab},\vec{y}^{(n)}\,|\,\Theta') + \sum_{\substack{a,b,c\\a<b<c}} p(\vec{s}_{abc},\vec{y}^{(n)}\,|\,\Theta') + \ldots.$$

Combining (13) and (14) yields:

$$\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} = \frac{\sum_a p(\vec{s}_a,\vec{y}^{(n)}\,|\,\Theta')\,\mathcal{A}_{id}(\vec{s}_a,W) + \sum_{\substack{a,b\\a<b}} p(\vec{s}_{ab},\vec{y}^{(n)}\,|\,\Theta')\,\mathcal{A}_{id}(\vec{s}_{ab},W) + \ldots}{p(\vec{0},\vec{y}^{(n)}\,|\,\Theta') + \sum_a p(\vec{s}_a,\vec{y}^{(n)}\,|\,\Theta') + \sum_{\substack{a,b\\a<b}} p(\vec{s}_{ab},\vec{y}^{(n)}\,|\,\Theta') + \ldots}. \qquad (15)$$

A similar grouping of terms is possible for the expectation $\langle s_h\rangle_{q_n}$.

If we now assume that the significant posterior probability mass will concentrate on vectors $\vec{s}$ with only a limited number of non-zero entries, the expanded sums in both numerator and denominator of (15) may be truncated without significant loss. The accuracy of the approximation depends both on the sparsity of the true generative process, and on the distance of the current model parameters (in the current EM iteration) from the true ones. In general, provided that the true process is indeed sparse, a truncated approximation will become more accurate as the estimated parameters approach their maximum likelihood values. The convergence properties and accuracy of algorithms based on this form of approximation will be tested numerically in Section 6.

Different choices of the truncation yield approximate algorithms with different properties. Two of these will be considered here.

4.1 MCA$_3$

In the first approximation, we truncate all but one of the sums that appear in the expansions of $\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n}$ and $\langle s_i\rangle_{q_n}$ after the terms that include three active sources, while truncating the numerator of $\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n}$ after the two-source terms (see Appendix C for details):

$$\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} \approx \frac{\overline{\pi}_i \exp(I_i^{(n)}) + \sum\limits_{c\,(c\neq i)} \overline{\pi}_i\overline{\pi}_c \exp(I_{ic}^{(n)})\,\mathcal{H}(W_{id}-W_{cd})}{1+\sum\limits_h \overline{\pi}_h \exp(I_h^{(n)}) + \frac{1}{2}\sum\limits_{\substack{a,b\\ a\neq b}} \overline{\pi}_a\overline{\pi}_b \exp(I_{ab}^{(n)}) + \frac{1}{6}\sum\limits_{\substack{a,b,c\\ a\neq b\neq c}} \overline{\pi}_a\overline{\pi}_b\overline{\pi}_c \exp(I_{abc}^{(n)})} \tag{16}$$

$$\text{and} \qquad \langle s_i\rangle_{q_n} \approx \frac{\overline{\pi}_i \exp(I_i^{(n)}) + \sum\limits_{c\,(c\neq i)} \overline{\pi}_i\overline{\pi}_c \exp(I_{ic}^{(n)}) + \frac{\alpha}{2}\sum\limits_{b,c\,(b\neq c\neq i)} \overline{\pi}_i\overline{\pi}_b\overline{\pi}_c \exp(I_{ibc}^{(n)})}{1+\sum\limits_h \overline{\pi}_h \exp(I_h^{(n)}) + \frac{1}{2}\sum\limits_{\substack{a,b\\ a\neq b}} \overline{\pi}_a\overline{\pi}_b \exp(I_{ab}^{(n)}) + \frac{1}{6}\sum\limits_{\substack{a,b,c\\ a\neq b\neq c}} \overline{\pi}_a\overline{\pi}_b\overline{\pi}_c \exp(I_{abc}^{(n)})}\,, \tag{17}$$

where

$$\begin{aligned}
&\overline{\pi}_i = \frac{\pi_i}{1-\pi_i}, && I_i^{(n)} = \sum_d \left(\log(W_{id})\, y_d^{(n)} - W_{id}\right),\\
&\tilde{W}_d^{ab} = \max(W_{ad},W_{bd}), && I_{ab}^{(n)} = \sum_d \left(\log(\tilde{W}_d^{ab})\, y_d^{(n)} - \tilde{W}_d^{ab}\right),\\
&\tilde{W}_d^{abc} = \max(W_{ad},W_{bd},W_{cd}), && I_{abc}^{(n)} = \sum_d \left(\log(\tilde{W}_d^{abc})\, y_d^{(n)} - \tilde{W}_d^{abc}\right),
\end{aligned} \tag{18}$$

and where $\mathcal{H}(x) = 1$ for $x > 0$; $\frac{1}{2}$ for $x = 0$; 0 for $x < 0$ is the Heaviside function. The above equations have been simplified by dividing both numerator and denominator by terms that do not depend on $\vec{s}$, for example, by $\prod_{i=1}^{H}(1-\pi_i)$ (see Appendix C). Approximations (16) and (17) are used in the fixed-point updates of Equations (9) and (11), where the parameters that appear on the right-hand-side are held at their current values. Thus all parameters that appear on the right-hand-side of the approximations take values in $\Theta' = (\vec{\pi}', W')$.

The early truncation of the numerator in (16) improves performance in experiments, partly by increasing competition between causes further, and partly by reducing the contribution of more complex data patterns that are better fit, given the current parameter settings, by three active sources than by two. By contrast, the three-source terms are kept in the numerator of (17). In this case, neglecting complex input patterns as in (16) would lead to greater errors in the estimated source activation probabilities π_i. Indeed, even while keeping these terms, π_i tend to be underestimated if the input data include many patterns with more than three active sources. To compensate, we introduce a factor of $\alpha > 1$ multiplying the three-source term in (17) (so that $\alpha = 1$ corresponds to the actual truncated sum), which is updated as described in Appendix C. This scheme yields good estimates of π_i, even if more than three sources are often active in the input data.

The M-step Equations (9) and (11) together with E-step approximations (16) and (17) represent a complete set of update equations for the MCA generative model. The computational cost of one parameter update grows polynomially in the total number of causes, with order H^3. The algorithm that is defined by these updates will therefore be referred to as MCA$_3$.

4.2 R-MCA$_2$

In the second place, we consider a restriction of the generative model in which (i) all s_h are distributed according to the same prior distribution with fixed parameter π; (ii) the weights W_{id} associated with each source variable i are constrained to sum to a constant C:

$$\forall i \in \{1,\ldots,H\}: \quad \pi_i = \pi \quad \text{and} \quad \sum_d W_{id} = C; \tag{19}$$

and (iii) on average, the influence of each hidden source is homogeneously covered by the other sources. This third restriction means that each non-zero generating weight W^{gen}_{id} associated with cause i can be covered by the same number of $W^{\text{gen}}_{cd} \geq W^{\text{gen}}_{id}$:

$$W^{\text{gen}}_{id} > 0 \quad \Rightarrow \quad \sum_{c\neq i} \mathcal{H}(W^{\text{gen}}_{cd} - W^{\text{gen}}_{id}) \approx b_i\,, \tag{20}$$

where $\mathcal{H}$ is the Heaviside function and b_i is the number of causes that can cover cause i. Figure 3 illustrates this condition. Figure 3A,B show weight patterns associated with hidden causes for which the condition is fulfilled; for instance in Figure 3B $b_i = 0$ for all causes with horizontal weight patterns, while $b_i = 1$ for the vertically oriented cause. In Figure 3C the condition is violated. Roughly, these conditions guarantee that all hidden causes have equal average effects on the generated data vectors. They make the development of a more efficient approximate learning algorithm possible but, despite their role in the derivation, the impact of these assumptions is limited in practice, in the sense that the resulting algorithm can perform well even when the input data set violates assumptions (19) and (20). This is demonstrated in a series of numerical experiments detailed below.

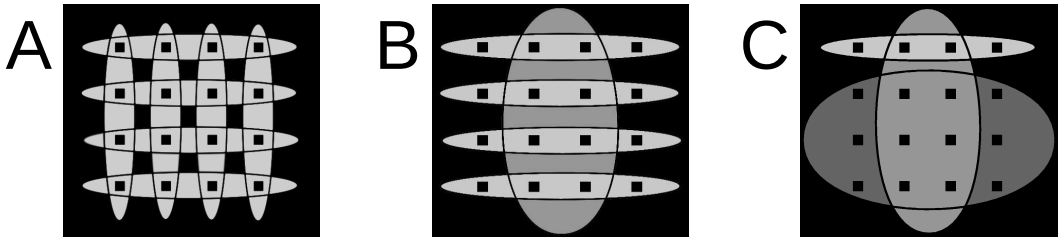


Figure 3: **A** and **B** show patterns of weights that satisfy the uniformity condition (20) whereas weights in **C** violate it. Each hidden cause is symbolized by an ellipse, with the gray-level of the ellipse representing the value W_{id} of each weight within the ellipse. Weights outside the ellipse for each cause are zero (black). The black squares indicate the 4-by-4 grid of observed pixels.

Update rules for the restricted generative model can again be derived by approximate expectation-maximization (see Appendix C). Using both the sum constraint of (19) and the assumption of homogeneous coverage of causes, we obtain the M-step update:

$$W_{id} \quad = \quad C \frac{\sum_n \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} y^{(n)}_d}{\sum_{d'}\sum_n \langle \mathcal{A}_{id'}(\vec{s},W)\rangle_{q_n} y^{(n)}_{d'}}\,. \tag{21}$$

Empirically, we find that the restricted parameter space of this model means that we can approximate the sufficient statistics $\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n}$ by a more severe truncation than before, now keeping two-source terms in the denominator, but only single-source terms in the numerator, of the expansion (15). This approximation, combined with the fact that any zero-valued observed patterns (i.e., those with $\sum_d y_d^{(n)} = 0$) do not affect the update rule (21) and so can be neglected, yields the expression (see Appendix C):

$$\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} \approx \frac{\exp(I_i^{(n)})}{\sum_h \exp(I_h^{(n)}) + \frac{\overline{\pi}}{2}\sum_{\substack{a,b\\ a\neq b}} \exp(I_{ab}^{(n)})}\,, \quad \overline{\pi} := \frac{\pi}{1-\pi}\,, \tag{22}$$

with abbreviations given in (18). Equations (21) and (22) are update rules for the MCA generative model, subject to the conditions (19) and (20). They define an algorithm that we will refer to as R-MCA$_2$ with R for *restricted* and with *2* indicating a computational cost that grows quadratically with H.

5. Relation to Neural Networks

We now relate component extraction as learned within the MCA framework to that achieved by a family of artificial neural networks. Consider the network of Figure 4 which consists of D input variables (or *units*) with values $y_1,\ldots,y_D$ and H hidden units with values $g_1,\ldots,g_H$. An observation $\vec{y}$ is represented by the values (or *activities*) of the input units, which act through *connections* parameterized by $(\mathcal{W}_{id})$ to determine the activities of the hidden units through an *activation function* $g_i = g_i(\vec{y},\mathcal{W})$. These parameters $(\mathcal{W}_{id})$ are known as the network (or synaptic) *weights*.

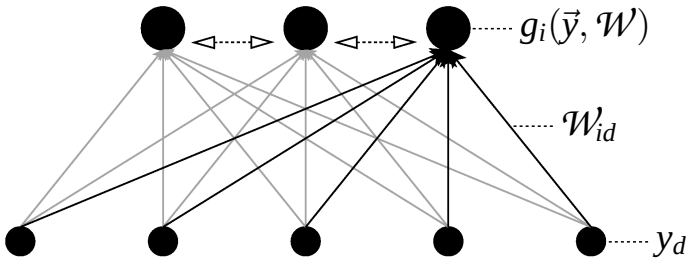


Figure 4: Architecture of a two layer neural network. Input is represented by values y_1 to y_D of D input units (small black circles). These values combine with synaptic weights $\mathcal{W}$ to determine the activities of the hidden units g_1 to g_H (big black circles). The dotted horizontal arrows symbolize lateral information exchange that may be required to compute the functions g_1 to g_H. After the g_i are computed the parameters $(\mathcal{W}_{id})$ are modified using a Δ-rule.

Learning in such a neural network involves adjusting the weights $\mathcal{W}$ in response to a series of input patterns, using a rule that is heuristically designed to extract some form of structure from these

inputs. A standard choice is the Hebbian Δ-rule with divisive normalization:

$$\Delta \mathcal{W}_{id} \;=\; \varepsilon g_i(\vec{y}, \mathcal{W}) y_d \quad \text{and} \quad \mathcal{W}_{id}^{\text{new}} = C \frac{\mathcal{W}_{id} + \Delta \mathcal{W}_{id}}{\sum_{d'} (\mathcal{W}_{id'} + \Delta \mathcal{W}_{id'})}, \tag{23}$$

The normalization step is needed to prevent weights from growing without bound, and the divisive form used here is most common. Here, the constant C defines the value at which $\sum_d \mathcal{W}_{id}$ is held constant; it will be related below to the C appearing in Equation 19. Many neural networks with the structure depicted in Figure 4, and that use a learning rule identical or similar to (23), have been shown to converge to weight values that identify clusters in, or extract useful components from, a set of input patterns (O'Reilly, 2001; Spratling and Johnson, 2002; Yuille and Geiger, 2003; Lücke and von der Malsburg, 2004; Lücke, 2004; Lücke and Bouecke, 2005; Spratling, 2006).

The update rule (23) depends on only one input pattern, and is usually applied *online*, with the weights being changed in response to each pattern in turn. If, instead, we consider the effect of presenting a group of patterns $\{\vec{y}^{(n)}\}$, the net change is approximately (see Appendix D):

$$\mathcal{W}_{id}^{\text{new}} \approx C \frac{\sum_n g_i(\vec{y}^{(n)}, \mathcal{W}) y_d^{(n)}}{\sum_{d'} \sum_n g_i(\vec{y}^{(n)}, \mathcal{W}) y_{d'}^{(n)}}. \tag{24}$$

Now, comparing (24) to (21), we see that if the activation function of a neural network were chosen so that $g_i(\vec{y}^{(n)}, \mathcal{W}) = \langle \mathcal{A}_{id}(\vec{s}, W) \rangle_{q_n}$, then the network would optimize the parameters of the restricted MCA generative model, with $W = \mathcal{W}$ (we drop the distinction between $\mathcal{W}$ and W from now on). Unfortunately, the expectation $\langle \mathcal{A}_{id}(\vec{s}, W) \rangle_{q_n}$ depends on d, and thus exact optimization in the general case would require a modified Hebbian rule. However, the truncated approximation of (22) is the same for all d, and so the changes in each weight depend only on the activities of the corresponding pre- and post-synaptic units. Thus, the Hebbian Δ-rule,

$$\Delta W_{id} \;=\; \varepsilon g_i y_d \quad \text{with} \quad g_i = \frac{\exp(I_i)}{\sum_h \exp(I_h) + \frac{\overline{\pi}}{2} \sum_{\substack{a,b \\ a \neq b}} \exp(I_{ab})} \tag{25}$$

(where I_h, I_{ab}, and $\overline{\pi}$ are the abbreviations introduced in Equations 18 and 22), when combined with divisive normalization, implements an online version of the R-MCA$_2$ algorithm. We refer to this online weight update rule as R-MCA$_{\text{NN}}$ (for Neural Network).

Note that the function g_i in (25) resembles the softmax function (see, e.g., Yuille and Geiger, 2003), but contains an additional term in the denominator. This added term reduces the change in weights when an input pattern results in more than one hidden unit with significant activity. That is, the system tries to explain a given input pattern using the current state of its model parameters W. If one hidden unit explains the input better than any combination of two units, that unit is modified. If the input is better explained by a combination of two units, the total learning rate is reduced.

Soft winner-take-all (WTA) activation functions, such as the softmax, are found in many networks that serve to both cluster *and* extract components from inputs, as appropriate. For clustering, the relationship between WTA-like competition and maximum-likelihood methods is well known (Nowlan, 1990). The connection drawn here offers a probabilistic account of the effectiveness of similar rules for component identification. If the probability of more than one cause being active is small (i.e., π is small), our activation rule for g_i (25) reduces to the standard softmax, suggesting that neural networks with activation and learning functions that resemble Equations (25) may perform well at both component extraction and clustering.

6. Experiments

The MCA generative model, along with the associated learning algorithms that have been introduced here, are designed to extract component features from non-linear mixtures. To study their performance, we employ numerical experiments, using artificial as well as more realistic data. The artificial data sets are based on a widely-used benchmark for non-linear component extraction, while the more realistic data are taken from acoustic recordings in one case and from natural images in the other. The goals of these experiments are (1) to establish whether the approximate algorithms do indeed increase the likelihood of the model parameters; (2) to test convergence and asymptotic accuracy of the algorithms; (3) to compare component extraction using MCA to other component-extraction algorithms; and (4) to demonstrate the applicability of the model and algorithms to more realistic data where non-linear component combinations arise naturally.

6.1 The Bars Test

The data sets used in experiments on artificial data were drawn from variants of the "bars test" introduced by Földiák (1990). Each data vector represents a grayscale image, with a non-linear combination of randomly chosen horizontal and vertical light-colored bars, each extending all the way across a black background. Most commonly, the intensity of the bars is uniform and equal, and the combination rule is such that overlapping regions remain at the same intensity. This type of data is a benchmark for the study of component extraction with non-linear interactions between hidden causes. Many component-extraction algorithms have been applied to a version of the bars test, including some with probabilistic generative semantics (Saund, 1995; Dayan and Zemel, 1995; Hinton et al., 1995; Hinton and Ghahramani, 1997), as well as many with non-generative objective functions (Harpur and Prager, 1996; Hochreiter and Schmidhuber, 1999; Lee and Seung, 2001; Hoyer, 2004) a substantial group of which have been neurally inspired (Földiák, 1990; Fyfe, 1997; O'Reilly, 2001; Charles et al., 2002; Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004; Lücke and Bouecke, 2005; Spratling, 2006; Butko and Triesch, 2007).

In most of the experiments described here the input data were 25-dimensional vectors, representing a 5-by-5 grid of pixels; that is, $D = 5 \times 5$. There were b possible single bars, some of which were superimposed to create each image. On the 5-by-5 grid there are 5 possible horizontal, and 5 vertical, bar positions, so that $b = 10$. Each bar appears independently with a probability π, with areas of overlap retaining the same value as the individual bars. Figure 5A shows an example set of noisy data vectors constructed in this way.

6.2 Annealing

The likelihood surface for the MCA generative model is potentially multimodal. Thus, hill-climbing algorithms based on EM may converge to local optima in the likelihood, which may well be considerably poorer than the global optimum. This tendency to find sub-optimal fixed points can be reduced by incorporating a deterministic annealing, or relaxation, procedure (Ueda and Nakano, 1998; Sahani, 1999), whereby the entropy of the posterior distribution in the free energy (4) is artificially inflated in early iterations, with this inflation progressively reduced in later iterations, under the control of a temperature parameter. All of the experiments discussed here incorporated deterministic annealing, the details of which are given in Appendix E.

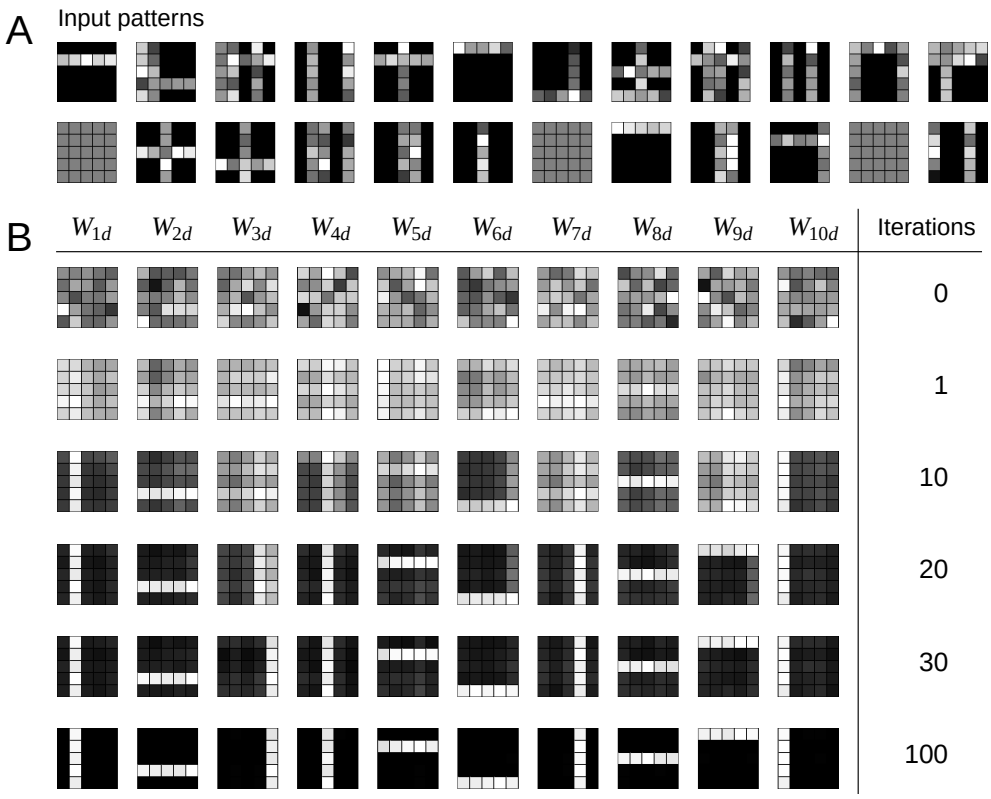


Figure 5: Bars test data with $b = 10$ bars on $D = 5 \times 5$ pixels and a bar appearance probability of $\pi = \frac{2}{10}$. **A** 24 patterns from the set of $N = 500$ input patterns that were generated according to the generative model with Poisson noise. **B** Change of the parameters W if MCA_3 is used for parameter update. Learning stopped automatically after 108 iterations in this trial (see Appendix E).

6.3 Convergence

From a theoretical standpoint, none of the four algorithms MCA_{ex}, MCA_3, R-MCA_2, or R-MCA_{NN}, can be guaranteed to maximize the likelihood of the MCA generative model. All of them update the parameters in the M-step using a fixed-point iteration, rather than either maximization or a gradient step. All but MCA_{ex} also approximate the posterior sufficient statistics (10). Thus, our first numerical experiments are designed to verify that the algorithms do, in fact, increase parameter likelihood in practice, and that they do converge. For this purpose, it is appropriate to use a version of the bars test in which observations are generated by the MCA model.

Thus, we selected MCA parameters that generated noisy bar-like images. There were 10 hidden sources in the generating model, one corresponding to each bar. The associated matrix of generating weights, W^{gen}, was 10×25, with each row representing a horizontal or vertical bar in a 5-by-5 pixel grid. The weights W^{gen}_{id} that correspond to the pixels of the bar were set to 10, the others to 0, so that $\sum_d W^{\text{gen}}_{id} = 50$. Each source was active with probability $\pi^{\text{gen}}_i = \frac{2}{10}$, leading to an average of two

bars appearing in each image. We generated $N = 500$ input patterns (each with 25 elements) using Equations (1) to (3); a subset of the resulting patterns is displayed in Figure 5A.

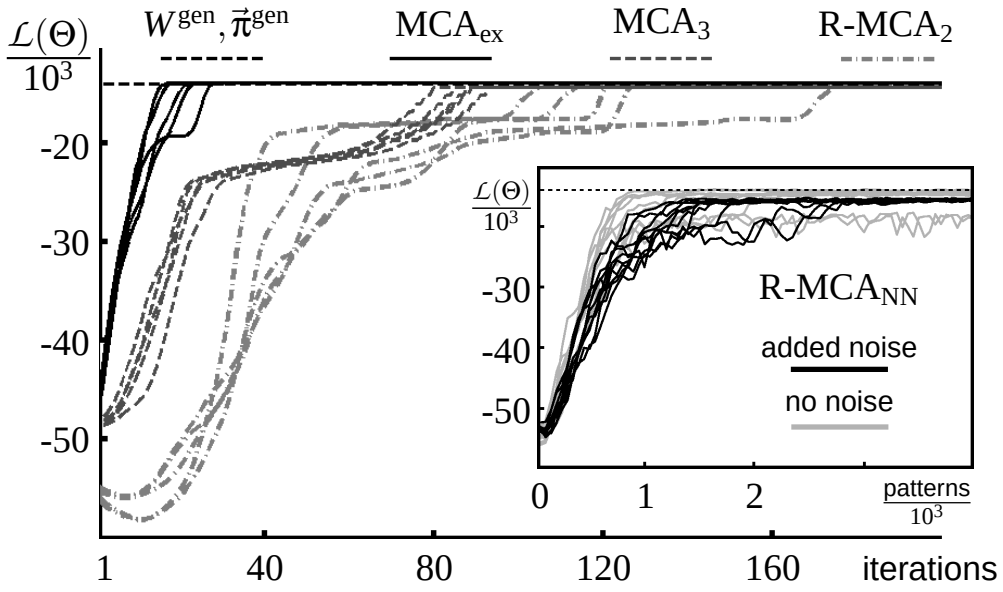


Figure 6: Change of the MCA parameter likelihood under different MCA learning algorithms. Data were generated as in Figure 5. To allow for comparison, the same set of $N = 500$ input patterns was used for all experiments shown. The likelihood of the generating parameters (W^{gen},$\vec{\pi}^{\text{gen}}$) is shown by the dotted horizontal line. The main axes show likelihood values of the batch-mode algorithms MCA_{ex}, MCA_3, and R-MCA_2 as a function of EM iteration. The inset axes shows likelihood values of the online algorithm R-MCA_{NN} as a function of number of input pattern presentations. Patterns were randomly selected from the set of $N = 500$ inputs, and the parameters were updated for each pattern.

Figure 6 shows the evolution of parameter likelihoods, as a function of iteration, for each of the MCA algorithms, with 5 different choices of initial parameters for each. With the exception of the first few iterations of R-MCA_2, the likelihood of the parameters under the batch mode algorithms increased at almost every iteration. The online R-MCA_{NN} showed greater fluctuations as updates based on individual data vectors inevitably perturbed the parameter estimates.

As might be expected, given the observation of increasing likelihoods and the fact that the likelihoods are bounded, each algorithm eventually converged from each initial value used in Figure 6. Furthermore, in each case, the likelihood of the solution found was close to the likelihood of the actual weights used in generation (the dashed horizontal lines). The final likelihood values for MCA_{ex} were slightly higher than the likelihoods of (W^{gen}, $\vec{\pi}^{\text{gen}}$), as is expected for an exact maximum-likelihood algorithm in noisy data; whereas the values achieved by the approximations MCA_3 and R-MCA_2 were slightly lower. In fact, in 100 further experiments, the annealing and parameter initialization schemes described in Appendix E, brought the likelihood close to that of the generating weights in 98 of 100 runs using R-MCA_2 and in 90 of 100 runs using MCA_3. We did not run

these more extensive tests for MCA_{ex} due to its long running time (it is also omitted from similar quantitative analyses below).

The two basic observations, that likelihoods generally increased at each iteration and that the batch-mode algorithms all reliably converged, held true for all of the experiments described here and below, even where data were not generated from a version of the MCA model. Thus, we conclude that these algorithms are generally robust in practice, despite the absence of any theoretical guarantees.

6.4 Parameter Recovery

Figure 5B shows the evolution of parameters W, under the approximate MCA_3 algorithm, showing that the estimated W did indeed converge to values close to the generating parameters W^{gen}, as was suggested by the convergence of the likelihood to values close to that of the generative parameters. While not shown, the convergence of W under MCA_{ex}, R-MCA_2 or R-MCA_{NN} was qualitatively similar to this sequence.

Clearly, if MCA_{ex} finds the global optimum, we would expect the parameters found to be close to those used for generation. The same is not necessarily true of the approximate algorithms. However, both MCA_3 and R-MCA_2 did in fact find weights W that were very close to the generating values whenever an obviously poor local optimum was avoided.

In MCA_3 the average pixel intensity of a bar was estimated to be 10.0 ± 0.5 (standard deviation), taken across all bar pixels in 90 trials where the likelihood increased to a high value. Using R-MCA_2 this value was estimated to be 10.0 ± 0.8 (across all bar pixels on 98 high-likelihood trials). Note that the Poisson distribution (2) results in a considerable variance of bar pixel intensities around the mean of 10.0 (compare Figure 5A) which explains the high standard deviation around the relatively precise mean value. The background pixels (original value zero) are estimated to have an intensity of 0.05 ± 0.02 in MCA_3 and are all virtually zero (all are smaller than 10^{-56}) in R-MCA_2. MCA_3 also estimates the parameters $\vec{\pi}$. Because of the finite number of patterns ($N = 500$) we compared the estimates with the actual frequency of occurrence of each bar i: $\pi'_i = (\text{numb of bars } i \text{ in input})/N$. The mean absolute difference between the estimate π_i and the actual probability π'_i was 0.006 (across the 90 trials with high likelihood), which demonstrates the relative accuracy of the solutions, despite the approximation made in Equation (17).

For the neural network algorithm R-MCA_{NN} given by (25) we observed virtually the same behavior as for R-MCA_2 when using a small learning rate (e.g., $\varepsilon = 0.1$) and the same cooling schedule in both cases (see Lücke and Sahani, 2007). The additional noise introduced by the online updates of R-MCA_{NN} had only a negligible effect. For larger learning rates the situation was different, however. For later comparison to noisy neural network algorithms, we used a version of R-MCA_{NN} with a relatively high learning rate of $\varepsilon = 1.0$. Furthermore, instead of a cooling schedule, we used a fixed temperature $T = 16$ and added Gaussian noise ($\sigma = 0.02$) at each parameter update: $\Delta W_{id} = \varepsilon g_i y_d + \sigma\eta$. With these learning parameters, R-MCA_{NN} learned very rapidly, requiring fewer than 1000 pattern presentations in the majority of trials. Ten plots of likelihoods against number of presented patterns are shown for R-MCA_{NN} in Figure 6 (inset figure, black lines) for the same $N = 500$ patterns as used for the batch-mode algorithms. Because of the additional noise in W, the final likelihood values were somewhat lower than those of the generating weights. Using R-MCA_{NN} with the same parameters but without added noise ($\sigma = 0$), final likelihood values were often higher (inset axes, gray lines) but the algorithm also converged to local optima more often. In

Reliability		
Model	noisy	no noise
MCA_3	90%	81%
R-MCA_2	98%	96%
R-MCA_{NN}	>99%	>99%

Reliability		
Model	no noise	reference
noisy-or	27%	Saund, 1995
competitive	69%*	Dayan/Zemel, 1995
LOCOCODE	96%	Hochreiter/Schmidhuber, 1999

Table 1: Comparison of MCA algorithms with other systems in the standard bars test with $b = 10$ bars ($D = 5 \times 5$, $\pi = \frac{2}{10}$, $N = 500$). For the MCA algorithms reliability values are computed on the basis of 100 trials. Values for these algorithms are also given for the same bars test with Poisson noise. Reliability values for the other systems are taken from the literature. For instance, the model of Hochreiter and Schmidhuber (1999) is reported to fail to extract all bars in one of 25 trials. Two systems, back-propagation (BP) and GeneRec, that are described by O'Reilly (2001) have also been applied to this bars test. In their standard versions, BP and GeneRec achieve 10% and 60% reliability, respectively. Hochreiter and Schmidhuber (1999) report that ICA and PCA extract only subsets of all bars. *Trained without bar overlap.

contrast, R-MCA_{NN} with noise avoided local optima in all 100 trials. In the following, R-MCA_{NN} will therefore refer to the noisy version with $\sigma = 0.02$ unless otherwise stated.

6.5 Comparison to Other Algorithms—Noiseless Bars

To compare the component extraction results of MCA to that of other algorithms reported in the literature, we used a standard version of the bars benchmark test, in which the bars appear with no noise. The competing algorithms do not necessarily employ probabilistic semantics, and may not be explicitly generative; thus, we cannot compare performance in terms of likelihoods, nor in terms of the accuracy with which generative parameters are recovered. Instead, we adopt a commonly used measure, which asks how *reliably* all the different bars are identified (see, e.g., Hochreiter and Schmidhuber, 1999; O'Reilly, 2001; Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004; Spratling, 2006). For each model, an internal variable (say the activities of the hidden units, or the posterior probabilities of each source being active) is identified as the response to an image. The responses evoked in the learned model by each of the possible single-bar images are then considered, and the most active unit or most probable source corresponding to each bar is identified. If the mapping from single-bar images to the most active internal variable is injective—that is, for each single bar a different hidden unit or source is the most active—then this instance of the model is said to have represented all of the bars. The reliability is the frequency with which each model represents all possible bars, when started from random initial conditions, and given a random set of images generated with the same parameter settings. For the MCA algorithms, the responses are defined to be the approximated posterior values for each possible source vector with only one active source, evaluated at the final parameter values after learning: $q(\vec{s}_h;\Theta) \approx p(\vec{s}_h\,|\,\vec{y}^{\,\mathrm{bar}},W)$.

The reliabilities of MCA_3, R-MCA_2, and R-MCA_{NN} as well as some other published component-extraction algorithms are shown in Table 1. These experiments used a configuration of the bars test much as above ($D = 5 \times 5$, $b = 10$, and $\pi^{\mathrm{gen}} = \frac{2}{10}$) which is perhaps the most commonly used

in the literature, (e.g., Saund, 1995; Dayan and Zemel, 1995; Hochreiter and Schmidhuber, 1999; O'Reilly, 2001). The bars have a fixed and equal gray-value. We generated $N = 500$ patterns according to these settings and normalized the input patterns $\vec{y}^{(n)}$ to lie in the interval $[0, 10]$ (i.e., bar pixels have a value of 10 and the background is 0). We considered both the case with Poisson noise (which has been discussed above) and the standard noiseless case. Experiments were run starting from 100 different randomly initialized parameters W. The same algorithms and the same cooling schedule were used (the same fixed T in the case of R-MCA$_{\text{NN}}$) to fit patterns with and without noise.

Without noise, MCA$_3$ with $H = 10$ hidden variables found all 10 bars in 81 of 100 experiments. R-MCA$_2$ with $H = 10$ found all bars in 96 of 100 experiments. Using the criterion of reliability, R-MCA$_{\text{NN}}$ performed best and found all bars in all 100 of 100 experiments. This seems likely to result from the fact that the added Gaussian noise, as well as noise introduced by the online updates, combined to drive the system out of shallow optima. Furthermore, R-MCA$_{\text{NN}}$ was, on average, faster than MCA$_3$ and R-MCA$_2$ in terms of required pattern presentations. It took fewer than 1000 pattern presentations to find all bars in the majority of 100 experiments,[1] although in a few trials learning did take much longer.

On the other hand, MCA$_3$ and R-MCA$_2$ achieved better likelihoods and recovered generative parameters closer to the true values. These algorithms also have the advantage of a well defined stopping criterion. MCA$_3$ learns the parameters of the prior distribution whereas R-MCA$_2$ uses a fixed value. R-MCA$_2$ does, however, remain highly reliable, even when the fixed parameter π differs significantly from the true value π^{gen}.

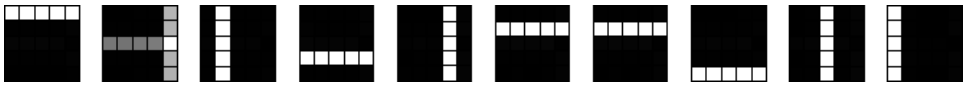


Figure 7: A common local optimum found by MCA$_3$ in the standard bars test. Two weight patterns reflect the same hidden cause, while another represents the superposition of two causes.

As was the case for the noisy bars, the R-MCA algorithms avoided local optima more often. This may well be a result of the smaller parameter space associated with the restricted model. A common local optimum for MCA$_3$ is displayed in Figure 7, where the weights associated with two sources generate the same horizontal bar, while a third source generates a weaker combination of two bars. This local solution is suboptimal, but the fact that MCA$_3$ has parameters to represent varying probabilities for each cause being present, means that it can adjust the corresponding rates to match the data. The fixed setting of π for R-MCA would introduce a further likelihood penalty for this solution.

Many component-extraction algorithms—particularly those based on artificial neural networks—use models with more hidden elements than there are distinct causes in the input data (e.g., Charles et al., 2002; Lücke and von der Malsburg, 2004; Spratling, 2006). If we use $H = 12$ hidden variables, then all the MCA-algorithms (MCA$_3$, R-MCA$_2$, and R-MCA$_{\text{NN}}$) found all of the bars in all of 100 trials.

1. Note that, according to the definition above, all bars are often already represented at intermediate likelihood values.

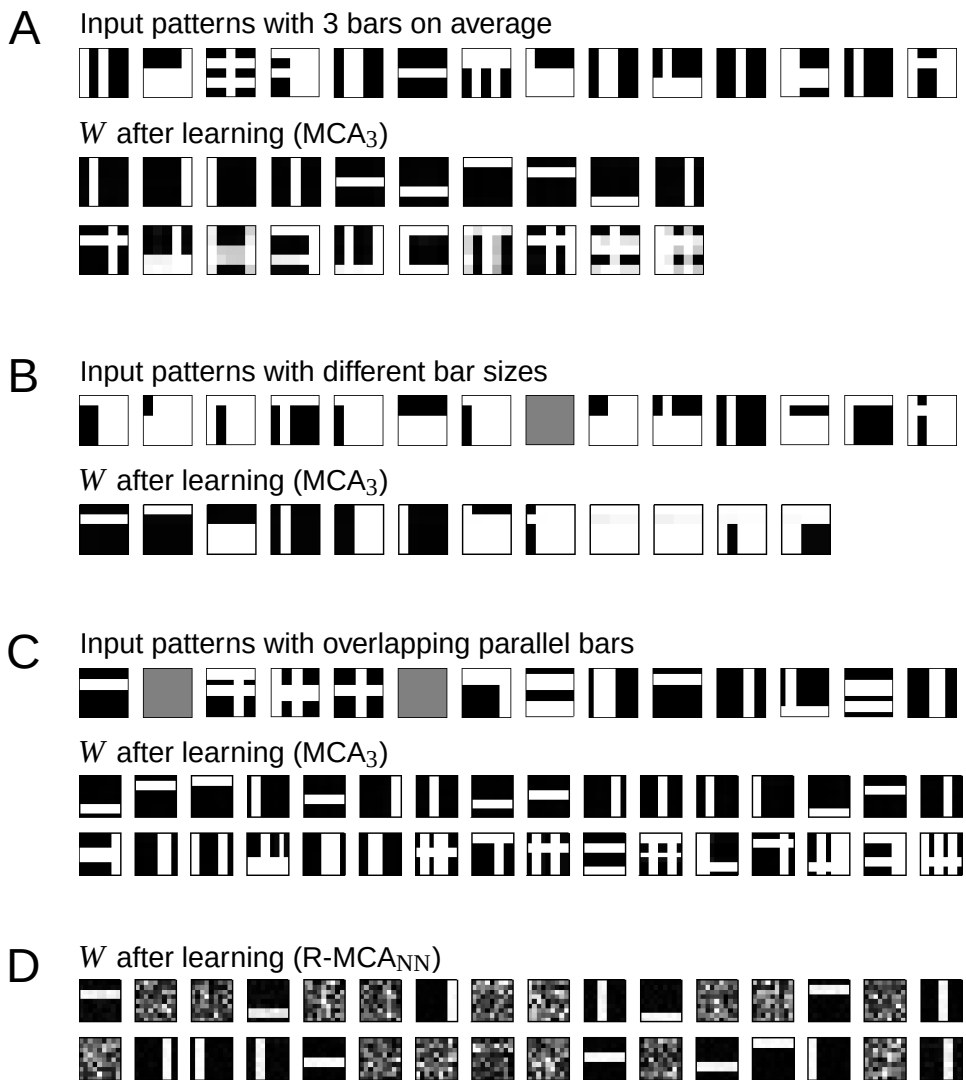


Figure 8: Experiments with increased bar overlap. In **A** bar overlap is increased by increasing the bar appearance probability to $\pi^{\text{gen}} = \frac{3}{10}$ (an average of three bars per pattern). In **B** bar overlap is varied using different bar widths (two one-pixel-wide bars and one three-pixel-wide bar for each orientation). In the bars test in **C** there are 8 (two-pixel-wide) horizontal bars and 8 (two-pixel-wide) vertical bars on a $D = 9 \times 9$ pixel grid. Each bar appears with probability $\pi^{\text{gen}} = \frac{2}{16}$ (two bars per input pattern on average). Each experimental data set is illustrated by 14 typical input patterns. For **A** to **C** the parameters W of a typical trial are shown if MCA$_3$ is used for learning. The vectors $\vec{W}_i = (W_{i1}, \ldots, W_{iD})$ appear in order of decreasing learned appearance probability π_i. In **D** the parameters W for a typical trial using R-MCA$_{\text{NN}}$ are shown.

6.6 Comparison to Other Algorithms—Bar Overlap

For most component-extraction algorithms that have been tested against the bars benchmark, it is difficult to know how specialized they are to the form of this test. The algorithms might, for example, depend on the fact that all bars appear with the same probability, or that they have the same width. Different versions of the bars test have therefore been introduced to probe how generally the different algorithms might succeed. In particular, there has been considerable recent interest in studying robustness to varying degrees of overlap between bars (see, e.g., Lücke and von der Malsburg, 2004; Lücke, 2004; Spratling, 2006). This is because it is the non-linear combination within the regions of overlap that most distinguishes the bars test images from linear superpositions of sources. In three different experiments we varied the degree of overlap in three different ways. Following Spratling (2006), in all experiments the MCA model had twice as many possible sources as there were bars in the generative input. In all experiments we used the same algorithms, initial conditions, and cooling schedules as described above and in Appendix E. Again, each trial used a newly generated set of training patterns and a different randomly generated matrix W. In the following, reliability values are computed on the basis of 25 trials each.

The most straightforward way to increase the degree of bar overlap is to use the standard bars test with an average of three instead of two bars per image, that is, take $\pi = \frac{3}{10}$ for an otherwise unchanged bars test with $b = 10$ bars on $D = 5 \times 5$ pixels (see Figure 8A for some examples). When using $H = 20$ hidden variables, MCA$_3$ extracted all bars in 92% of 25 experiments. Thus the algorithm works well even for relatively high degrees of superposition. The values of W found in a typical trial are shown in Figure 8A. The parameters $\vec{W}_i = (W_{i1}, \ldots, W_{iD})$ that are associated with a hidden variable or unit are sorted according to the learned appearance probabilities π_i. Like MCA$_3$, both R-MCA$_2$ and R-MCA$_{\text{NN}}$ were run without changing any parameters. In the restricted case, this meant that the assumed value for the source probability ($\pi = \frac{2}{10}$) was different from the generating value ($\pi^{\text{gen}} = \frac{3}{10}$). Nevertheless, the performance of both algorithms remained better than that of MCA$_3$, with R-MCA$_2$ and R-MCA$_{\text{NN}}$ finding all 10 bars in 96% and 100% of 25 trials, respectively.

We can also choose unequal bar appearance probabilities (cf., Lücke and von der Malsburg, 2004). For example, half the bars appeared with probability $\pi_h^{\text{gen}} = (1+\gamma)\,\frac{2}{10}$ and the other half[2] appeared with probability $\pi_h^{\text{gen}} = (1-\gamma)\,\frac{2}{10}$, MCA$_3$ extracted all bars in all of 25 experiments for $\gamma = 0.5$. For $\gamma = 0.6$ (when half the bars appeared 4 times more often than the other half) all bars were extracted in 88% of 25 experiments. For $\gamma = 0.6$ R-MCA$_2$ and R-MCA$_{\text{NN}}$ found all bars in 96% and 100% of 25 experiments respectively. Reliability values for R-MCA$_{\text{NN}}$ started to decrease for $\gamma = 0.7$ (92% reliability).

As suggested by Lücke and von der Malsburg (2004), we also varied the bar overlap in a second experiment by choosing bars of different widths. For each orientation we used two one-pixel wide bars and one three-pixel-wide bar. Thus, for this data set, $b = 6$ and $D = 5 \times 5$. The bar appearance probability was $\pi = \frac{2}{6}$, so that an input contained, as usual, two bars on average. Figure 8B shows some examples. MCA$_3$ extracted all bars in 84% of 25 experiments for this test. Reliability values decreased for more extreme differences in the bar sizes. R-MCA$_2$ and R-MCA$_{\text{NN}}$ both found all bars in all 25 trials each. Thus, although the unequal bar sizes violated the assumption $\sum_d W_{id} = C$ that was made in the derivation of R-MCA$_2$ and R-MCA$_{\text{NN}}$, the algorithms' performance in terms of reliability seemed unaffected.

2. If bars are numbered $h = 1$ to 5 for the horizontal and $h = 6$ to 10 for the vertical, we chose the ones with even numbers to appear with the higher probability.

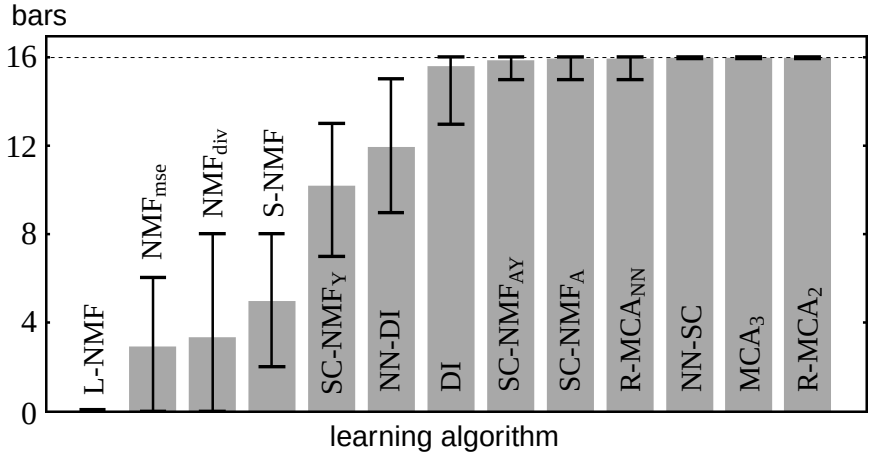


Figure 9: Comparison of MCA$_3$, R-MCA$_2$, and R-MCA$_{NN}$ with other systems in the bars test with increased occlusion (compare Figure 8C and Figure 2). Bars test parameters are $D = 9 \times 9$, $b = 16$, $\pi = \frac{2}{16}$, and $N = 400$. Data for the non-MCA algorithms are taken from Spratling (2006). The bar heights represent the average numbers of extracted bars in 25 trials. Error bars indicate the largest and the lowest number of bars found in a trial. The algorithms NN-DI and DI are feed-forward neural networks of the type depicted in Figure 4. All other (non-MCA) algorithms are versions of NMF with different objectives and constraints (see Appendix E and Spratling, 2006, for details).

In the third experiment we changed the degree of bar overlap more substantially, using a bars test that included overlapping parallel bars as introduced by Lücke (2004). We used eight horizontal and eight vertical bars, each two pixels wide, on a 9-by-9 grid. Thus, two parallel neighboring bars had substantial overlap. Figure 8C shows some typical input patterns. Note that the introductory example of Figure 2A,B is also of this type. To allow for a detailed comparison with other systems we adopted the exact settings used by Spratling (2006), that is, we considered 25 runs of a system with $H = 32$ hidden variables using bars test parameters $D = 9 \times 9$, $\pi^{\text{gen}} = \frac{2}{16}$, and $N = 400$. For these data, MCA$_3$ found all 16 bars in all of 25 experiments. The same is true for R-MCA$_2$ whereas R-MCA$_{NN}$ missed one bar in one of the 25 trials. Figure 9 shows a quantitative comparison with other algorithms that have been applied to this version of the bars test. Of the ten algorithms studied by Spratling (2006) just one, namely non-negative sparse coding (NN-SC; Hoyer, 2002, with sparseness parameter $\lambda = 1$), is as reliable as MCA$_3$ and R-MCA$_2$. The other systems, including forms of NMF both with and without a sparseness constraint, fail partly or entirely in extracting the actual hidden causes. For a typical trial using MCA$_3$ the final parameters W are displayed in Figure 8C. Again the $\vec{W}_i$'s associated with the different hidden variables are sorted according to their learned parameters π_i. A qualitatively different set of $\vec{W}_i$'s was obtained when R-MCA$_{NN}$ was used for learning. Figure 8D shows a typical outcome ($\vec{W}_i$'s are not sorted). In this case, only the actual causes are clearly represented whereas the $\vec{W}_i$'s of the supernumerary units remain unspecialized. The same feature is reported by Spratling (2006) for the algorithms NN-DI and DI used in this same test. Convergence to a representation that contains just the true hidden causes and leaves super-

numerary units unspecialized can improve the interpretability of the result. When using a higher fixed temperature for R-MCA$_{NN}$ all the hidden units represented bars, with some bars represented by more than one unit. However, hidden units that represented more composite inputs, as seen for MCA$_3$, were rarely observed. On the other hand, the parameters found by MCA$_3$ provide an indication of significance of each weight pattern in the appearance probabilities π_i. Thus, in Figure 8C the appearance probabilities for the first 16 sources are much higher than for the others. The later sources may be interpreted as capturing some of the higher-order structure that results from a finite set of input patterns. In contrast to R-MCA, such higher-order representations need not adversely affect the data likelihood because the corresponding appearance probabilities can be relatively small.

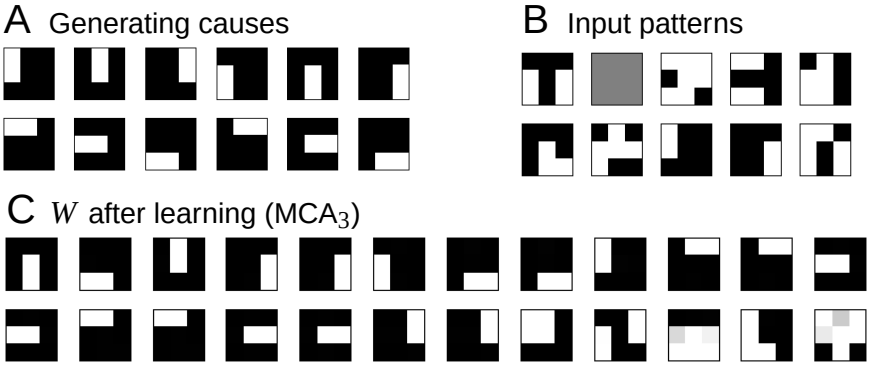


Figure 10: Experiments with more causes and hidden variables than observed variables. **A** The 12 patterns used to generate the data. Each is a 1-by-2 pixel bar on a 3-by-3 grid ($D = 9$). **B** Ten examples of the 500 input patterns generated using the causes shown in **A**. **C** Parameters W found in a typical run of MCA$_3$ with $H = 24$. The vectors $\vec{W}_i = (W_{i1}, \ldots, W_{iD})$ appear in order of decreasing learned appearance probability π_i.

6.7 More Causes than Observed Variables

In the experiments described above, the number of hidden causes was always smaller than the number of observed variables. We next briefly studied the "over-complete" case where data were generated, and models were fit, using more hidden causes than observed variables. We generated $N = 500$ patterns on a 3-by-3 grid ($D = 9$), using sparse combinations of 12 hidden causes corresponding to 6 horizontal and 6 vertical bars, each 1-by-2 pixels in size and thus extending across only a portion of the image (Figure 10A). As in the bars tests above, black was assigned to a value of 0 and white to 10. Patterns were generated without noise, with an average of two bars appearing in each ($\pi = \frac{2}{12}$). Ten such patterns are shown in Figure 10B.

Figure 10C shows the weights learned during a typical run using MCA$_3$ with the same parameter settings as above and twice as many hidden variables than observed ones ($H = 24$). Weights are sorted in order of decreasing inferred appearance probabilities π_i. All 12 causes were identified, with many represented more than once. A few hidden variables, with lower inferred probabilities of

appearance, were associated with more composite patterns. MCA_3 extracted all causes in all of 25 trials. R-MCA_2 also extracted all causes in all of 25 trials, and never represented composite patterns. R-MCA_{NN} only extracted all causes when run at fixed temperatures that were lower than those used for the bars tests above (e.g., $T = 3$), in which case it did so in all of 25 trials. This requirement for a lower temperature was consistent with the observation that a lower data dimension D leads to a decrease in the critical temperatures associated with the algorithms (see Appendix E). For larger values of T (e.g., $T = 16$) R-MCA_{NN} did not extract single causes.

6.8 Violations of Model Assumptions

To optimize the likelihood of the data under the MCA generative model, each of the approximate learning algorithms relies on the fact that, under the Bernoulli prior (1), some number of the observed data vectors will be generated by only a small number of active sources. To highlight this point we explicitly removed such sparse data vectors from a standard bars test, thereby violating the Bernoulli prior assumption of the generative model. We used bars tests as described above, with $b = 10$ or $b = 16$ bars and $\pi = \frac{2}{b}$, generating $N = 500$ (or more) patterns, in each case by first drawing causes from the Bernoulli distribution (1) and then rejecting patterns in which fewer than m causes were active. As might be expected, when m was 3 or greater the approximate algorithms all failed to learn the weights associated with single causes. However, when only patterns with fewer than 2 bars had been removed, MCA_3 was still able to identify all the bars in many of the runs. More precisely, using data generated as above with $b = 10$, $m = 2$ and $N = 500$, MCA_3 with $H = 10$ hidden variables found all causes in 69 of 100 trials with noisy observations and in 37 of 100 trials without noise (the parameters for MCA_3 and the associated annealing schedule were unchanged). Note that in these experiments the average number of active causes per input vector is increased by the removal of sparse data vectors. An increase in reliability in the noisy case is consistent with our other experiments. The relatively low reliability seen for noiseless bars in this experiment may be due to the combined violation of both the assumed prior and noise distributions.

As long as the data set did contain some vectors generated by few sources, the learning algorithms could all relatively robustly identify the causes given sufficient data, even when the average observation contained many active sources. For instance, in a standard noiseless bars test with $b = 16$ bars on an 8×8 grid, and $N = 1000$ patterns with an average of four active causes in each ($\pi = \frac{4}{16}$), all three algorithms still achieved high reliability values, using twice as many hidden variables as actual bars ($H = 32$), and using the same parameters as for the standard bars test above. MCA_3 found all causes in 20 of 25 trials in these data (80% reliability). Reliabilities of R-MCA_2 and R-MCA_{NN} (25 trials each) were 76% and 100%, respectively. The reliabilities of all algorithms fell when the data set contained fewer patterns, or when the average number of bars per pattern was larger.

6.9 Applications to More Realistic Data

We study two examples of component extraction in more realistic settings, applying the MCA algorithms to both acoustic and image data.

Acoustic data. Sound waveforms from multiple different sources combine linearly, and so are conventionally unmixed using algorithms such as ICA applied to simultaneous recordings from multiple microphones. The situation is different, however, for *spectrogram* representations of natural

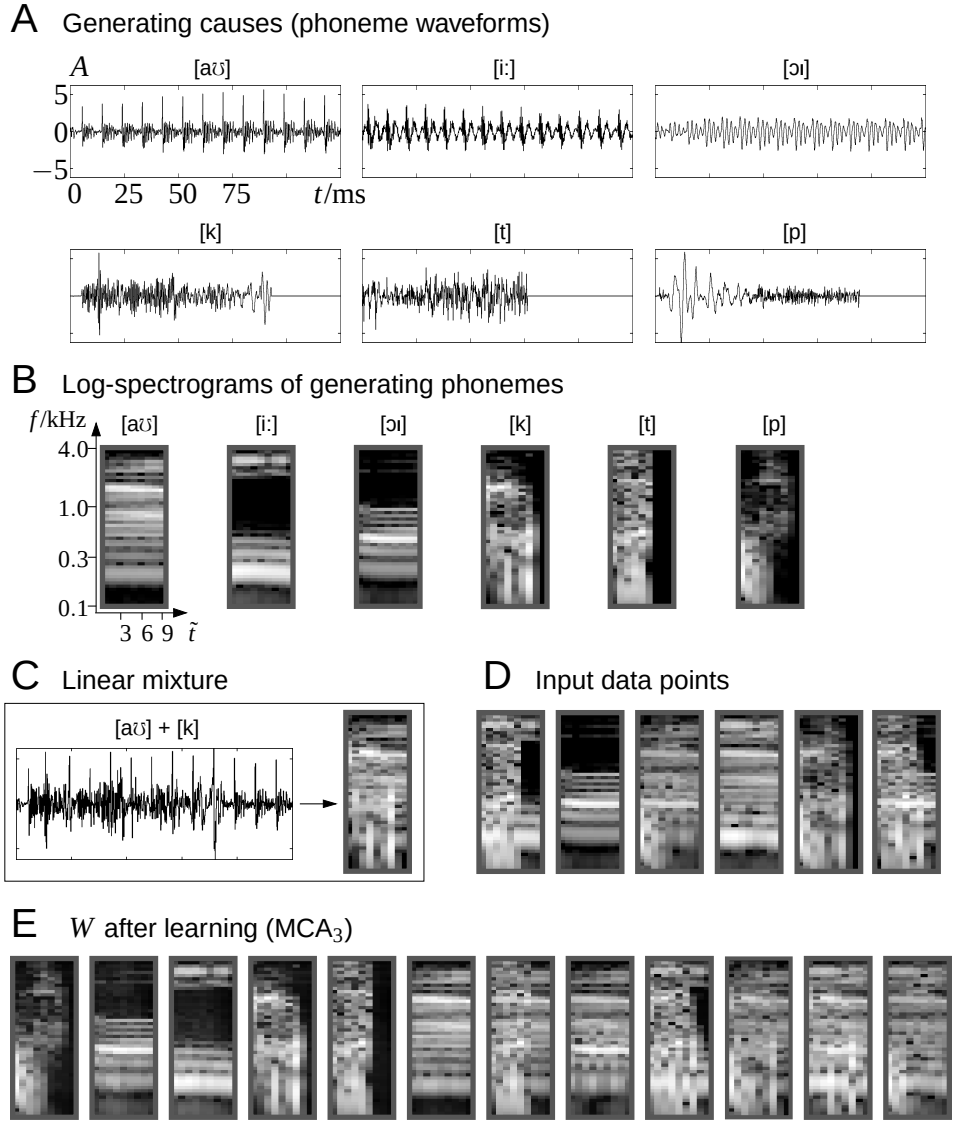


Figure 11: Application to acoustic data. **A** Pressure waveforms of six phonemes spoken by a male voice. Axes here, and for the waveform in **C**, are as shown for [aʊ] (A is a normalized amplitude). **B** The log-spectrograms of the phonemes in **A**. We use 50 frequency channels and nine time windows ($\tilde{t} = 1, \ldots, 9$). Axes of all log-spectrograms in the figure are as shown for [aʊ]. **C** Waveform of the linear mixture of phonemes [aʊ] and [k], and the log-spectrogram of this linear mixture. **D** Six examples of the $N = 500$ data vectors that were used for the experiments. Each data vector is the log-spectrogram of a linear mixture of the phoneme waveforms in **A**. The data sets for the experiments used an average of two waveforms per data vector. **E** Parameters W found by MCA_3 with $H = 12$, using 500 mixture log-spectrograms. The parameter vectors $\vec{W}_i = (W_{i1}, \ldots, W_{iD})$ appear in order of decreasing learned appearance probability π_i and are linearly scaled to fill the gray scale.

sound. The power of natural sounds in individual time-frequency bins varies over many orders of magnitude, and so is typically measured logarithmically and expressed in units of decibels, giving a representation that is closely aligned with the response of the cochlea to the corresponding sound. In this representation, the combination of log-spectrograms of the different sources may be well approximated by the max rule (R. K. Moore, 1983, quoted by Roweis, 2003). In particular, the logarithmic power distribution, as well as the sub-linear power summation due to phase misalignment, both lead to the total power in a time-frequency bin being dominated by the single largest contribution to that bin (see Discussion).

To study the extraction of components from mixtures of sound by MCA, we based the following experiment on six recordings of phonemes spoken by a male voice (see Figure 11A). The phoneme waveforms were mixed *linearly* to generate $N = 500$ superpositions, with each phoneme appearing in each mixture with probability $\pi = \frac{2}{6}$. Thus each mixture comprised two phonemes on average, with a combination rule that resembled the MCA max-rule in the approximate sense described above.

We applied the MCA algorithms to the log-spectrograms of these mixtures. Figure 11B shows the log-spectrograms of the individual phonemes and Figure 11C shows the log-spectrogram of an example phoneme mixture. We used 50 frequency channels and 9 time bins to construct the log-spectrograms. The resulting values were thresholded and then rescaled linearly so that power-levels across all phonemes filled the interval $[0, 10]$, as in the standard bars test. For more details see Appendix E.

The MCA algorithms were used with the same parameter settings as in the bars tests above, except that annealing began at a lower initial temperature (see Appendix E). As in the bars tests with increased overlap, we used twice as many hidden variables ($H = 12$) as there were causes in the input. Figure 11E shows the parameters W learned in one run using MCA_3. The parameter vectors $\vec{W}_i = (W_{i1}, \ldots, W_{iD})$ are displayed in decreasing order of the corresponding learned value of π_i. As can be seen, the first six such vectors converged to spectrogram representations similar to those of the six original phonemes. The six hidden variables associated with lower values of π_i, converged to weight vectors that represented more composite spectrograms. This result is representative of those found with MCA_3. R-MCA_2 also converged to single spectrogram representations, but tended to represent those single spectrograms multiple times rather than representing more composite patterns with the additional components. Results for R-MCA_{NN} were very similar to those for R-MCA_2 when we used a high fixed temperature (see Appendix E for details). For intermediate fixed temperatures, results for R-MCA_{NN} were similar to those of the bars test in Figure 8D in that each cause was represented just once, with additional hidden units displaying little structure in their weights. For lower fixed temperatures (starting from $T \approx 40$) R-MCA_{NN} failed to represent all causes.

In general, the reliability values of all three algorithms were high. These were measured as described for the bars tests above, by checking whether, after learning, inference based on each individual phoneme log-spectrogram led to a different hidden cause being most probable. MCA_3 found all causes in 21 of 25 trials (84% reliability), R-MCA_2 found all causes in all of 25 trials; as did R-MCA_{NN} (with fixed $T = 70$). Reliability for MCA_3 improved to 96% with a slower cooling procedure ($\theta_{\Delta W} = 0.25 \times 10^{-3}$; see Appendix E).

Visual data. Finally, we consider a data set for which the exact hidden sources and their mixing rule are unknown. The data were taken from a single 250-by-250 pixel gray-level image of grass taken

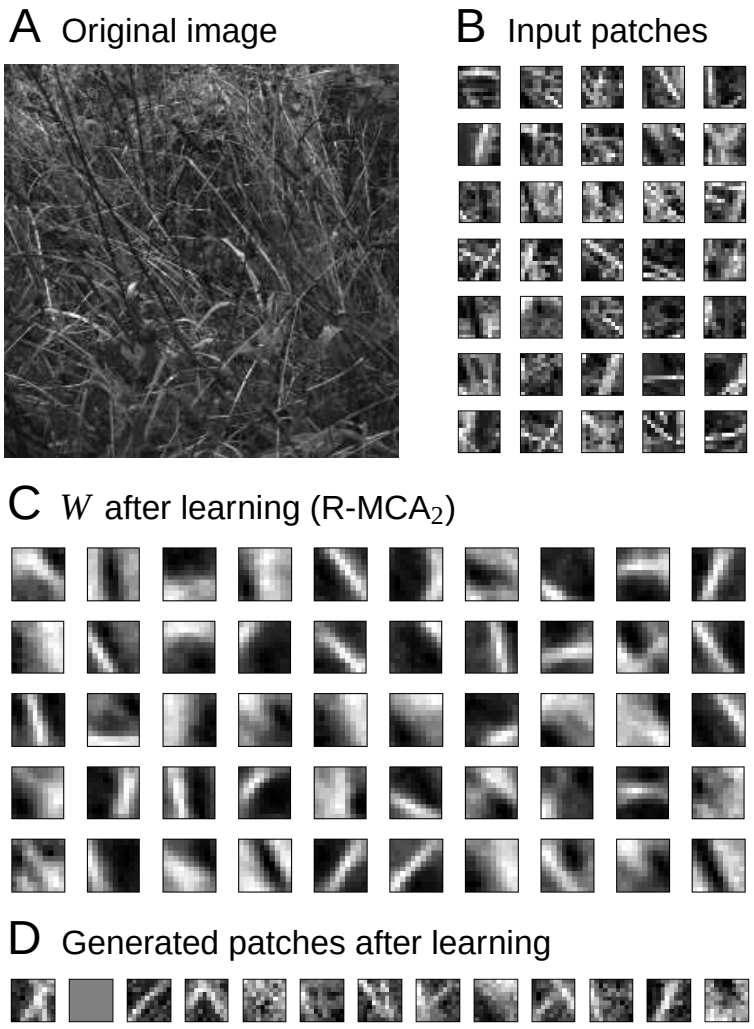


Figure 12: Application to visual data. **A** The 250-by-250 pixel image used as basis for the experiments. The image is taken from the van Hateren database of natural images (see Appendix E). For visualization we have brightened the image (we let values in the lower half of the light intensity range fill the range of gray values from zero to 255 and clamped values in the upper half to value 255). Without brightening, the image would appear unnaturally dark on a finite gray scale because of a small number of pixels with very high values. **B** 35 examples taken from the 5000 10-by-10 pixel patches that were used for numerical experiments. The patches represent light intensities linearly. For visualization, each patch has been scaled to fill the range of gray values. **C** Parameters W resulting from a typical run of R-MCA$_2$ with $H = 50$ hidden variables and $N = 5000$ image patches. For visualization, each parameter vector $\vec{W}_i = (W_{i1}, \ldots, W_{iD})$ has been linearly scaled to fill the range of gray values. **D** Patches generated using the restricted generative model and weights as in **C** (patches have been scaled as in **B** and **C**).

from the van Hateren database (Figure 12A) and linearly rescaled so that pixel intensities filled the interval $[0,10]$. Each data vector was a 10-by-10 pixel patch drawn from a random position in the image (see Figure 12B for some examples).

The image comprised stems and blades of grass which occluded each other. As discussed above (see, e.g., Figure 2), the combination rule for such objects may be well approximated by the max rule of the MCA generative model (at least for the lighting conditions that appear to prevail in Figure 12A). Thus, the MCA learning algorithms may be expected to converge to parameters W that represent intensity images of 'grass'-like object parts. However, each blade of grass might appear at many different positions within the image patches, rather than at a fixed set of possible locations as in the bars test. Thus to recover these grass-like elements in the MCA causal weights requires the use of models with large numbers of hidden variables (and, correspondingly, many data vectors). For the number of patches and hidden variables required, the cubic cost of MCA_3 led to impractically long execution times. In experiments with smaller patch sizes and small H (e.g., $H = 10$ or $H = 20$) some weight patterns did converge to represent 'grass'-like objects, but many converged to less structured configurations.

The computational cost of R-MCA_2 is smaller and we evaluated trials using $H = 50$ hidden variables and $N = 5000$ 10-by-10 patches. R-MCA_2 was used with the same parameter setting as for the bars tests above, except for lower initial and final temperatures for annealing (see Appendix E). Figure 12C shows a typical outcome obtained when cooling from $T = 4.0$ to $T = 1.0$. A large number of weight vectors have converged to represent 'grass'-like object parts, whereas others represent more extensive causes that might be interpreted as capturing background noise. Many of the weight patterns have an orientation similar to the dominant orientation in the original image. Figure 12D shows a selection of patches generated using the learned weights. We used a higher value of C during generation than during learning (the parameter is not learned with R-MCA_2), thus globally rescaling the learned weights, so as to reduce the apparent noise level. In experiments where annealing was terminated at $T = 1.5$ (as in the bars test), the resulting weights were generally similar to the ones in Figure 12C, but with a larger proportion of weight vectors showing little structure. Learning with slower annealing did not result in significantly different weights. With fewer than $N = 5000$ patches for training, the weight patterns were less smooth, presumably reflecting overfitting to the subset of data used.

In experiments applying the online algorithm R-MCA_{NN} to a set of 5000 10-by-10 patches as above, we found that it would converge to 'grass'-like weight patterns provided the learning rate (ε in Equation 25) was set to a much lower value than had been used in the bars tests. A lower learning rate corresponds to effectively averaging over a much larger set of input patterns. With $\varepsilon = 0.02$ (instead of 1.0 as above), and with noise on the weights (σ) scaled down by the same factor, R-MCA_{NN} converged to weights similar to those shown in Figure 12C (for R-MCA_2), although a larger number of hidden units showed relatively uniform weight structure. For R-MCA_{NN} we used a fixed temperature of $T = 2.0$.

7. Discussion

We discuss the applicability of the MCA learning algorithms, and the generality of the MCA framework, before relating the new algorithms to previous methods and neural network systems.

7.1 Applicability of the Model

The MCA generative model and associated learning algorithms are designed to extract causal components from input data in which the components combine non-linearly. More precisely, the generative model assumes that the single active cause with the strongest influence on a particular observed variable alone determines its observed value—something we have referred to here as the max-rule for combination. This stands in contrast to other feature extraction models such as PCA, ICA, NMF, or SC, in which the influences of the different causes are summed.

One context in which data with a superposition property very close to the max-rule arise naturally is the psychoacoustic combination of sounds. The perception of sound is largely driven by the logarithm of the time-varying intensity within each of a bank of narrow-band frequency channels. The narrow-band, short-time intensity of natural sounds may vary over many orders of magnitude. Further, sounds from different sources may have unrelated phases, and so intensities within each channel will generally add sub-linearly. Thus, even though sound *waveforms* from different sources combine linearly, the time- and frequency-local *intensities*, expressed logarithmically (in decibels), are dominated by the loudest of the sounds within each time-frequency bin. Indeed, even if two sounds are of equal loudness, the intensity of the sum is greater than each of them by at most 3 dB. Here, then, the max-rule is a very good approximation to the true generative combination. This observation motivated our use of acoustic data in the experiments shown in Figure 11.

In the image domain, the max-rule's relevance comes from the fact that it matches the true occlusive combination rule more closely than does the more commonly used sum. This is true both quantitatively (see Figure 2 and the discussion thereof), and also qualitatively, in the sense that both occlusion and the max-rule share a property of exclusiveness—that is, only one of the hidden causes determines the value of each pixel. Numerical experiments on raw image data (Figure 12) demonstrate that plausible generative causes are extracted using the MCA approach. The weight patterns associated with the extracted causes resemble images of the single object parts (blades and stems of grass in our example) that combine non-linearly to generate the image. The MCA approach also holds some potential for component extraction in more low-level image processing, for example, if we assume that each input pixel is generated exclusively by one edge instead of a whole object or object part. The application of MCA might, however, be less straight-forward in this case and presumably requires image preprocessing and perhaps a different noise model.

7.2 Generality of the Framework

Many of the details of the algorithms presented here, as well as many of the experiments, have been based on a specific model in which the hidden variables are drawn from a multivariate Bernoulli distribution (1), and the observations are then Poisson, conditioned on these values (2). These choices are natural ones for non-negative data generated from binary sources (cf., NMF; Lee and Seung, 1999, 2001). However, while the details have largely been omitted for brevity, it is straightforward to incorporate alternative generative distributions within the same framework, and with the same approximations.

Thus, the equations that define the M-step (9), as well as the expansion (15) used to approximate the E-step, would hold for any well-behaved prior over binary variables. In particular, the sources need not be marginally independent. This generality contrasts with the key assumption of independence that underlies many linear combination models. It suggests that an extension to a

hierarchical model, in which higher-order statistics in the source distribution were captured by a further parametric model, might be straightforward.

The general formalism also remains unchanged for different noise distributions. Thus, if the conditional distribution of observations given sources, here Poisson, were instead Gaussian, all of the derivations and approximations would essentially remain unchanged, with one exception being the definitions of $I_i^{(n)}$, $I_{ab}^{(n)}$, and $I_{abc}^{(n)}$ in Equation (18). Approximate learning for the additional variance parameter of a Gaussian distribution would also be straightforward, largely following the arguments developed for the parameters W and $\vec{\pi}$. If suited to the data set under consideration, distributions other than Poisson and Gaussian may also be used within this same framework, and combined with different dependent or independent prior distributions.

7.3 Relationship to Variational Approximations

A now standard approach to approximate learning in intractable models is to replace the true posterior distribution of equation (12) by an approximate q_n that is obtained by minimizing the Kullback-Leibler divergence $\mathsf{KL}[q_n \| p(\vec{s}|\vec{y}^{(n)}, \Theta')]$ within a constrained class of functions. This provides a form of variational learning (Jordan et al., 1999), which provably increases a lower bound on the likelihood at every iteration. A common choice of a constrained family might be one which factors over the latent variables. Unfortunately, this common choice is of little benefit in the MCA generative model, as the costs of evaluation of the expected values of the derivatives $\mathcal{A}_{id}(\vec{s}, W)$, given in (8), grows exponentially even under factored distributions.

An alternative approach would be to constrain q_n to place mass only on source configurations where a limited number of causes are active. The minimum Kullback-Leibler divergence under this constraint would then be achieved when the probability of such sparse configurations under q_n was proportional to the corresponding true posterior values. Revisiting the argument leading to equation (15), it is clear that such an approximation would correspond to truncating the sums in both numerator and denominator of (15), as well as the corresponding expression for $\langle s_h \rangle_{q_n}$, at the same point. Our experience has been that the algorithms described here, in which fewer terms are kept in the numerator of (15) than in the denominator, always perform better than this strict variational approach.

7.4 The Different MCA Algorithms

The computational cost of exact expectation-maximization learning (i.e., MCA_{ex}) in the MCA generative model grows exponentially in the smaller of the number of observation dimensions and the number of hidden variables ($\min(D, H)$), and is thus generally intractable. We have introduced three approximations, all based on early truncation of the expanded sums in Equation (15). One of these, MCA_3, with cubic computational complexity, learns all the parameters of the full generative model, including the prior source probabilities. However, if the sums over source distributions are truncated further, to yield an algorithm with quadratic complexity, experimental performance of an otherwise unconstrained algorithm suffers. This difficulty is avoided in the restricted version of the generative model, in which the prior probabilities are held fixed and equal, and the weights associated with the sources satisfy a homogeneous coverage property. In this restricted model, the quadratic-cost algorithm becomes effective, and we have studied both a batch-mode algorithm, R-MCA_2, and an online version R-MCA_{NN}. Experiments showed that these restricted algorithms remained effective in terms of identifying generative weight vectors, even when the data were generated with prior

source probabilities that were substantially different from that assumed by the algorithms (see, e.g., the experiments of Figure 8A or the discussion of violations of model assumptions in Experiments). Furthermore, the R-MCA algorithms were also robust to violations of the assumption of homogeneously distributed hidden causes (20). In some situations, the R-MCA algorithms succeeded in extracting the true causes where MCA_3 did not. We have observed that this is particularly true for bars with large differences in intensity. For this type of data, R-MCA_2 appears to be the most robust of the algorithms (see Lücke and Sahani, 2007). The R-MCA algorithms may also be more robust to greater differences in bar widths than those that we have studied here. Overall, in terms of the reliability with which hidden sources are recovered, R-MCA_2 and R-MCA_{NN} may outperform MCA_3 even in experiments in which the assumptions used to derive them are violated. These results suggest that constrained optimization can improve measures such as reliability or learning time (in terms of pattern presentations), even when the constraint is not exactly valid. Approximately valid constraints may make it easier to avoid local optima, and to learn from fewer examples. However, the more severe approximation of R-MCA_2 and R-MCA_{NN} can affect the likelihood of the parameters found. In this sense, MCA_3 is the more successful algorithm. One approach to increasing the speed of convergence might be to use R-MCA_{NN} to provide initial values to MCA_3, thus reducing the number of cubic-complexity iterations required for final convergence. Such a hybrid algorithm would provide a learning system with relatively short learning times, high final likelihoods and high reliability.

7.5 Relationship to Previous Algorithms

It is helpful to divide the algorithms that have previously been proposed for component extraction into three groups: generative models with linear superposition, competitive generative models, and neural network models in which assumptions about the data are implicit in the network structure and learning algorithm.

7.5.1 LINEAR SUPERPOSITION MODELS

The functional difference between linear superposition and the max-rule has been discussed above. Despite the mismatch in the generative process, linear superposition models have been used within non-linear component extraction contexts, with some success. In particular, the non-negativity constraints of NMF have helped to identify constructively combined features (Lee and Seung, 1999).

For non-negative data, the specific algorithms developed here (MCA_3, R-MCA_2, and R-MCA_{NN}) can all be regarded as explicitly non-linear alternatives to the different versions of NMF. In particular, the Poisson noise distribution matches one of the cost functions often used with NMF (Lee and Seung, 1999). The basic methodology of our MCA development is, however, independent of the assumption of non-negativity.

It is worth noting that non-negativity may be better suited to finding featural sub-parts (cf., Lee and Seung, 1999; Wersing and Körner, 2003) of generative components (as was, in fact, originally proposed) than the entire components. In the bars test (with $b = 16$) Spratling (2006) showed that at least some NMF algorithms succeed in extracting all of the bars. However, if the bar overlap is increased (as in the test depicted in Figure 2 and Figure 8C), most NMF algorithms fail. For such input data, NMF only succeeds if its objective function is extended by an additional term that enforces a form of sparseness. MCA_3 and R-MCA_2 perform better on these data than all other algorithms tested (see Figure 9 for results) except for one sparse-NMF version (NN-SC; Hoyer,

2002) that performs equally well. However, for this and other sparse versions of NMF the sparseness parameter (or parameters) must be chosen either based on prior knowledge about the input, or by trial-and-error (see Spratling, 2006, for a critical discussion).

In contrast, the *generative model* underlying MCA does not assume sparseness. Instead, the notion of sparseness was introduced in the discussion of learning, to justify the tractable approximate learning algorithms MCA_3, R-MCA_2, and R-MCA_{NN}. The truncated approximations that underlie these algorithms are more accurate when the input causes are sparsely active; but this does not incorporate an explicit prior for sparsity in the way that SC, ICA, or sparse-NMF do, and does not enforce a pre-specified degree of sparseness in the learned generative model. Indeed, the MCA algorithms were found to robustly optimize the data likelihood, even for input causes that were not sparsely active on average. It is possible, however, that if data were generated by a process that was substantially different from that assumed by MCA, the approximate algorithms might well introduce a bias towards a sparser solution. These differences in approach to sparsity between MCA and models such as SC, ICA, and sparse-NMF, suggest that MCA might provide a good basis from which to study the relationship between non-linear component combinations and sparsity assumed in learning algorithms.

7.5.2 COMPETITIVE GENERATIVE MODELS

Models that use an explicitly non-linear generative combination rule include those of Saund (1995), which uses a noisy-or rule for binary observations, and of Dayan and Zemel (1995), where the combination scheme is more competitive. The MCA model may be viewed as taking this competition to an extreme, by selecting just one hidden variable to be responsible for each observed one.

Competitive generative models have proven challenging from a learning standpoint, in that published algorithms often converge to local optima. In the bars test ($b = 10$, $N = 500$) the noisy-or algorithm (Saund, 1995) finds all bars in just 27% of trials. The more competitive scheme (Dayan and Zemel, 1995) only extracts all bars if bar overlap is excluded for training, that is, if training patterns only contain parallel bars. In this simplified case the system achieves 69% reliability.

For comparison, the MCA learning algorithms MCA_3, R-MCA_2, and R-MCA_{NN} all show significantly higher values of reliability in the same bars test (see Table 1), at least when combined with an annealing procedure. The reliability can be boosted further in two ways—either by adding Poisson noise to the input (Table 1), or by adding more hidden variables or units to the model (in which case all three MCA algorithms find all 10 bars in all of our experiments).

7.5.3 NEURAL NETWORK MODELS

High reliability in component extraction in the bars test is not a feature exclusive to the new algorithms presented. Other highly reliable systems include some that optimize a non-probabilistic objective function (e.g., Charles and Fyfe, 1998; Hochreiter and Schmidhuber, 1999; Charles et al., 2002) as well as neural network models (Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004; Lücke, 2004; Lücke and Bouecke, 2005; Spratling, 2006; Butko and Triesch, 2007). While the probabilistic generative approach has the advantage of a principled framework, which makes clear the assumptions being made about the data, it has been criticized (Hochreiter and Schmidhuber, 1999; Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004) for not working reliably—that is, for often failing to extract the true causes.

The models and algorithms introduced here show that a generative approach can indeed be made robust. The R-MCA$_{NN}$ algorithm shows that generative and neural network approaches can come together in the form of a competitive neural network model that is both reliable and probabilistically interpretable. Using a high learning rate and additional noise, the network model R-MCA$_{NN}$ avoids local optima and needs few pattern presentations for learning (less than 1000 in the majority of trials). The same is reported for other network models (Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004; Spratling, 2006) which fit into the framework of Equation (23) and Figure 4. The appropriate activation rule for a network to optimize the data likelihood under our generative model turns out to be a generalization of the softmax rule (see Equation 25). For input generated by very sparsely active causes, this generalization reduces to the usual softmax, which is commonly used for clustering (see, e.g., McLachlan and Peel, 2000). The generalized rule (25) therefore offers an explanation for why some networks (Spratling and Johnson, 2002; Lücke and von der Malsburg, 2004; Lücke, 2004) can also be successfully applied to clustering tasks. However, R-MCA$_{NN}$ and standard neural network algorithms can differ in the details of their behavior. On the one hand, for data involving substantial overlap between components (e.g., Figure 9), R-MCA$_{NN}$ seems to be more robust than the DI and NN-DI networks discussed by Spratling (2006). On the other hand, DI and the network of Lücke and von der Malsburg (2004) seem to be more robust to larger differences in component sizes.

A distinguishing feature of our model is the use of the max function. In neural modeling this function has also been used in other contexts and for other purposes. Among other models (e.g., Grzywacz and Yuille, 1990) it has been used as an activation function for hidden units in a feedforward model for visual object recognition (Riesenhuber and Poggio, 1999). However this use in the recognition model should not be confused with our use of the max function in the generative process. Indeed, inference within the MCA model shows that the appropriate activation function of hidden units, for example, (16) or (22), is necessarily more complex. The extraction of input components, for example, in the bars test, fails if a simple max is used for inference instead. However, for input without superposition (and for recognition after learning) a max function as used by Riesenhuber and Poggio (1999) may be interpreted as a further approximation of the generalized softmax in the neural network approximation of R-MCA$_{NN}$.

7.6 Conclusion

To conclude, we have formulated a novel class of generative models that competitively combines hidden causes. In place of the linear superposition of prominent models like PCA, ICA, SC, and NMF, we use the max-operation. We have shown how a new technique for posterior approximation in such models can provide efficient parameter update rules if the input causes are sparsely active. Making specific choices for prior and noise distributions, we obtain efficient algorithms that performed well on artificial and natural non-linear mixtures, and are found to be competitive with the best current performance on standard non-linear benchmarks.

Acknowledgments

We thank Richard Turner for helpful discussions and for assistance with the acoustic data. This work was funded by the Gatsby Charitable Foundation.

Appendix A. Maximum Likelihood

To maximize $\mathcal{F}(\Theta,\Theta')$ in (5) with respect to W_{id} we require that:

$$\begin{aligned}
& \frac{\partial}{\partial W_{id}}\mathcal{F}(\Theta,\Theta') \;\stackrel{!}{=}\; 0 \\
\Rightarrow\quad & \sum_n \sum_{\vec{s}} q_n(\vec{s};\Theta') \sum_{d'} \left(\frac{\partial}{\partial W_{id}} \log\left(p(y_{d'}^{(n)} \,|\, \overline{W}_{d'}(\vec{s},W)) \right) \right) \;\stackrel{!}{=}\; 0 \\
\Rightarrow\quad & \sum_n \sum_{\vec{s}} q_n(\vec{s};\Theta') \left(\frac{\partial}{\partial W_{id}} \overline{W}_d(\vec{s},W) \right) f(y_d^{(n)}, \overline{W}_d(\vec{s},W)) \;\stackrel{!}{=}\; 0\,, \qquad (26) \\
& \text{where} \qquad f(y,w) = \frac{\partial}{\partial w}\log(p(y\,|\,w))\,, \qquad (27)
\end{aligned}$$

with $p(y|w)$ given in (2). Now, for any well-behaved function g, and large ρ:

$$\mathcal{A}_{id}^{\rho}(\vec{s},W)\, g(\overline{W}_d(\vec{s},W)) \approx \mathcal{A}_{id}^{\rho}(\vec{s},W)\, g(W_{id})\,, \text{ where } \mathcal{A}_{id}^{\rho}(\vec{s},W) := \frac{\partial}{\partial W_{id}} \overline{W}_d^{\rho}(\vec{s},W). \qquad (28)$$

Equation (28) holds because $\mathcal{A}_{id}^{\rho}(\vec{s},W) \approx 0$ whenever $\overline{W}_d(\vec{s},W) \neq W_{id}$. Hence it follows from (26) that:

$$\begin{aligned}
& \sum_n \sum_{\vec{s}} q_n(\vec{s};\Theta')\, \mathcal{A}_{id}^{\rho}(\vec{s},W)\; f(y_d^{(n)}, W_{id}) \;\stackrel{!}{\approx}\; 0\,, \qquad (29) \\
\Rightarrow\quad & \sum_n \sum_{\vec{s}} q_n(\vec{s};\Theta')\, \mathcal{A}_{id}^{\rho}(\vec{s},W) \left(y_d^{(n)} - W_{id} \right) \;\stackrel{!}{\approx}\; 0\,. \qquad (30)
\end{aligned}$$

Equation (9) is obtained in the limit of large ρ. To be more precise, we might have used $\mathcal{A}_{id}^{\rho}(\vec{s},W)$ instead of $\mathcal{A}_{id}(\vec{s},W)$ in the main text. However, we abstained from doing so for the sake of readability, and because only the limit $\rho \to \infty$ is needed to derive the learning algorithms. The expression for this limit given in Equation (8) is found from (7), with the derivative of $\overline{W}_d^{\rho}$ given by:

$$\frac{\partial}{\partial W_{id}} \overline{W}_d^{\rho}(\vec{s},W) = \left(\frac{s_i\,(W_{id})^{\rho}}{\sum_h s_h (W_{hd})^{\rho}} \right) \frac{\left(\sum_h s_h (W_{hd})^{\rho}\right)^{\frac{1}{\rho}}}{s_i\, W_{id}}\,. \qquad (31)$$

In the limit $\rho \to \infty$ this reduces to (8) because the second factor on the right-hand-side converges to 1 whenever the first term is nonzero.

Note that (29) is true for any type of conditionally independent noise distribution. Only in the final step, from Equation (29) to (30), is the specific form of the Poisson distribution needed, where it appears in the derivative (27).

Appendix B. Intractability of Sufficient Statistics

To compute the exact sufficient statistics in (10) requires the evaluation of sums over all possible hidden states $\vec{s}$, suggesting a computational complexity of 2^H. In some cases, however, there may be multiple different source configurations, all of which result in the same effective weights $\overline{W}_d(\vec{s},W)$ for all dimensions $d \in \{1,\ldots,D\}$. In such cases, it might be possible to group these equal terms in each of the sums together, thereby reducing the complexity of the sum to scale with the number of

such groups, rather than with the number of source configurations. In fact, we show below that the complexity of computing these expected values in general scales at least as fast as $2^{\min(H,D)}$.

We examine three cases individually:

$H = D$ Consider parameters W for which, corresponding to each observed node d, there is a hidden node i such that $W_{id} > W_{jd}$ for all $j \neq i$ (beyond this restriction, the entries in W may have arbitrary values). For $H = D$ this condition can be satisfied, and if it is, then for any two hidden vectors $\vec{s}$ and $\vec{s}'$ there is a d such that $\overline{W}_d(\vec{s},W) \neq \overline{W}_d(\vec{s}',W)$. In other words: any change of the hidden vector $\vec{s}$ results in a change of the pre-noise output vector $(\overline{W}_1(\vec{s},W),\ldots,\overline{W}_D(\vec{s},W))$. Hence, each summand in the partition function in (12) contributes a potentially different value, and they must be evaluated one-by-one. Thus, in this case the computational cost scales as 2^H.

$H < D$ Consider a subset of H of the D observed nodes and apply the argument above. Thus, the computational cost scales as 2^H. Note that for fixed H and random parameters W the existence of H (or approximately H) hidden nodes for which the above condition is fulfilled becomes increasingly likely with increasing D.

$H > D$ On the one hand, if we just consider D of the H hidden nodes and apply the argument above, we can infer that the computational complexity grows with at least 2^D. On the other hand, we can obtain at most $H^D + 1$ different vectors $(\overline{W}_1(\vec{s},W),\ldots,\overline{W}_D(\vec{s},W))$ and thus at most $H^D + 1$ groups to sum over. The computational complexity thus lies between 2^D and $H^D + 1$ in this case.

Appendix C. MCA$_3$ and R-MCA$_2$—Details of the Derivations

The update rules that define the algorithms MCA$_3$ and R-MCA$_2$ follow directly from (12) and (15) using the distributions (1) and (2). Note that in (15) the joint probability $p(\vec{s},\vec{y}^{(n)}\,|\,\Theta')$ can be replaced by any function F satisfying $F(\vec{s},\vec{y}^{(n)},\Theta') = \frac{p(\vec{s},\vec{y}^{(n)}\,|\,\Theta')}{A(\vec{y}^{(n)},\Theta')}$, where A is any well-behaved function not depending on $\vec{s}$. For the update rules of MCA$_3$ and R-MCA$_2$ we have used:

$$F(\vec{s},\vec{y}^{(n)},\Theta) = \left(\prod_i \overline{\pi}_i^{s_i}\right) \exp(I^{(n)}), \quad I^{(n)} = \sum_d \left(\log(\overline{W}_d(\vec{s},W))\, y_d^{(n)} - \overline{W}_d(\vec{s},W)\right).$$

C.1 MCA$_3$

The first term in the sum over states $\vec{s}$ in the denominator of (16) and (17) only contributes significantly if $\vec{y}^{(n)} = \vec{0}$, that is, if $F(\vec{0},\vec{y}^{(n)},\Theta) = 1$ (given Poisson noise). In all other cases its contribution is negligible. To derive the numerator of (16) we have used the property:

$$\mathcal{A}_{id}(\vec{s}_h,W) = \delta_{ih}, \qquad \mathcal{A}_{id}(\vec{s}_{ab},W) = \delta_{ia}\,\mathcal{H}(W_{id} - W_{bd}) + \delta_{ib}\,\mathcal{H}(W_{id} - W_{ad}), \tag{32}$$

where $\mathcal{H}$ is the Heaviside function. Note that instead of (32) we also could have used $\mathcal{A}_{id}(\vec{s}_h,W)$ and $\mathcal{A}_{id}(\vec{s}_{ab},W)$ directly or the corresponding expressions of $\mathcal{A}^{\rho}_{id}(\vec{s},W)$ in (31) with high ρ. By using (32) we can simplify the expression of the numerator of (16), however.

The derivation of the numerator of (17) (with $\alpha = 1$) is straightforward. For less sparse input we have to correct for neglecting input patterns which were generated by four or more hidden causes.

We do so by updating α using a consistency argument. On the one hand, using the same arguments as for the derivation of (16) and (17), we can estimate the total number of input patterns generated by less than three causes:

$$\mathcal{N}^{\leq 2}(Y,\Theta) \approx \sum_n \frac{\mathcal{H}(\frac{1}{2}-\sum_d y_d^{(n)}) + \sum_i \overline{\pi}_i \exp(I_i^{(n)}) + \frac{1}{2}\sum_{a,b\,(a\neq b)} \overline{\pi}_a\overline{\pi}_b \exp(I_{ab}^{(n)})}{1+\sum_h \overline{\pi}_h \exp(I_h^{(n)}) + \frac{1}{2}\sum_{\substack{a,b\\a\neq b}} \overline{\pi}_a\overline{\pi}_b \exp(I_{ab}^{(n)}) + \frac{1}{6}\sum_{\substack{a,b,c\\a\neq b\neq c}} \overline{\pi}_a\overline{\pi}_b\overline{\pi}_c \exp(I_{abc}^{(n)})}\,. \tag{33}$$

On the other hand, the same number can be estimated using the prior distributions alone:

$$\tilde{\mathcal{N}}^{\leq 2}(\vec{\pi}) \;=\; N\left(\prod_i (1-\pi_i)\right)\left(1+\sum_h \overline{\pi}_h + \frac{1}{2}\sum_{a,b\,(a\neq b)} \overline{\pi}_a\overline{\pi}_b\right). \tag{34}$$

If the parameters π_i are underestimated using approximation (17), the estimate (33) is smaller than the estimate (34). We change α after each EM iteration until both estimates are consistent ($\mathcal{N}^{\leq 2}(Y,\Theta) \approx \tilde{\mathcal{N}}^{\leq 2}(\vec{\pi})$):

$$\alpha \;=\; \alpha^{\text{old}} + \frac{\varepsilon_\alpha}{N}\left(\tilde{\mathcal{N}}^{\leq 2}(\vec{\pi}) - \mathcal{N}^{\leq 2}(Y,\Theta)\right).$$

Note that the additional computational cost to infer α is small. Computations in (34) scale quadratically with H, and the terms in (33) have to be computed for (17) anyway. In experiments we use $\varepsilon_\alpha = 1$.

C.2 R-MCA$_2$

If we optimize (5) under the constraint $\sum_d W_{id} = C$ in (19), we obtain:

$$\sum_n \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} \frac{y_d^{(n)} - W_{id}}{W_{id}} + \mu_i = 0\,.$$

The elimination of the Lagrange multipliers μ_i results in:

$$W_{id} \;=\; \frac{\sum_n \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} y_d^{(n)}}{\frac{1}{C}\sum_{n,d'} \langle \mathcal{A}_{id'}(\vec{s},W)\rangle_{q_n} y_{d'}^{(n)} - \sum_n \left((\sum_{d'} \langle \mathcal{A}_{id'}(\vec{s},W)\rangle_{q_n} \frac{W_{id'}}{C}) - \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n}\right)}\,.$$

If the model parameters W fulfill condition (20), which can be expected at least close to the maximum likelihood solution, we obtain after the rearrangement of terms, the approximate M-step of Equation (21). To derive the sufficient statistics (22) note that given the update rule (21) we have:

$$W_{id} = C\frac{\sum_n \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} y_d^{(n)}}{\sum_{d'}\sum_n \langle \mathcal{A}_{id'}(\vec{s},W)\rangle_{q_n} y_{d'}^{(n)}} = C\frac{\sum_{n\in N^{>0}} \langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n} y_d^{(n)}}{\sum_{d'}\sum_{n\in N^{>0}} \langle \mathcal{A}_{id'}(\vec{s},W)\rangle_{q_n} y_{d'}^{(n)}}\,,$$

where $N^{>0}$ is the set of all non-zero input patterns. To approximate $\langle \mathcal{A}_{id}(\vec{s},W)\rangle_{q_n}$ we can therefore assume input statistics that follow from Equations (1) to (3), but in which all inputs exactly equal to zero are omitted. For such modified input statistics, Equation (15) remains unchanged except for the partition function $\mathcal{Z}$ in (14) whose term $p(\vec{0},\vec{y}^{(n)}\,|\,\Theta')$ now equals zero. If we truncate the sum in (15) after terms of order two, we obtain Equation (22).

Appendix D. Neural Network Details

Here we show that the online neural network update rule (23) approaches the batch rule (24) for large data sets and small learning rates. Let $\mathcal{W}^{(n)}$ be the weight matrix at the nth update, and for convenience define the Hebbian correlation signal to be $G_{id}^{(n)} = g_i(\vec{y}^{(n)}, \mathcal{W}^{(n)}) y_d^{(n)}$, and the weight renormalization term to be $z_i^{(n)} = C^{-1} \sum_{d'} \left(\mathcal{W}_{id'}^{(n)} + \varepsilon G_{id'}^{(n)} \right) = (1 + \frac{\varepsilon}{C} \sum_{d'} G_{id'}^{(n)})$ (where we have used the fact that $\sum_{d'} \mathcal{W}_{id'}^{(n)} = C$). Then we can rewrite (23) as

$$\mathcal{W}_{id}^{(n)} = \frac{\mathcal{W}_{id}^{(n-1)} + \varepsilon G_{id}^{(n-1)}}{z_i^{(n-1)}},$$

and by applying N updates starting from initial step t find that

$$\mathcal{W}_{id}^{(t+N)} = \frac{\mathcal{W}_{id}^{(t)} + \varepsilon \sum_{n=1}^{N} G_{id}^{(t+N-n)} \prod_{k=n+1}^{N} z_i^{(t+N-k)}}{\prod_{k=1}^{N} z_i^{(t+N-k)}},$$

where the empty product at $n = N$ is taken equal to 1.

We now make approximations based on the assumptions that N is large, ε is small, and that $\mathcal{W}^{(t)}$ is drawn from the equilibrium distribution over weights. First, as each $z_i^{(n)}$ is (slightly) larger than 0, the sum will be dominated by the leading terms (where n is small). The coefficients of these terms can be approximated, assuming that ε is small, by a logarithmic transform and the weak law of large numbers: $\prod_{k=n+1}^{N} z_i^{(t+N-k)} \approx \exp\left((N-n) \frac{\varepsilon}{C} \sum_{d'} \overline{G}_{id'} \right)$, where $\overline{G}_{id}$ is the expected value of $G_{id}^{(n)}$, which is taken to be stationary by the equilibrium assumption.

Inserting this expression into (35), taking the expected value of the right-hand-side, and summing the resulting geometric series to infinity, we obtain:

$$\mathcal{W}_{id}^{(t+N)} \approx e^{-N\varepsilon \sum_{d'} \overline{G}_{id'}/C} \mathcal{W}_{id}^{(t)} + \varepsilon \overline{G}_{id} \left(\frac{1}{1 - e^{-\varepsilon \sum_{d'} \overline{G}_{id'}/C}} - 1 \right).$$

Finally, assuming $N\varepsilon$ to be large enough for the first term to be negligible, expanding the second, and keeping only terms that do not scale with ε we obtain

$$\mathcal{W}_{id}^{(t+N)} \approx C \frac{\overline{G}_{id}}{\sum_{d'} \overline{G}_{id'}} \approx C \frac{\sum_{n=1}^{N} g_i(\vec{y}^{(n)}, \mathcal{W}^{(t)}) y_d^{(n)}}{\sum_{d'} \sum_{n=1}^{N} g_i(\vec{y}^{(n)}, \mathcal{W}^{(t)}) y_{d'}^{(n)}}.$$

This equivalence of the online and batch rules at equilibrium shows that the average fixed points of R-MCA$_{\text{NN}}$ equal fixed points of R-MCA$_2$. The equivalence becomes inexact away from equilibrium, although our experiments suggest that the behaviour during convergence may nonetheless be similar (Lücke and Sahani, 2007).

Appendix E. Experimental Details

This appendix gives details of the training procedures used for the MCA algorithms, provides more information about the other algorithms used for comparison, and gives particulars of the acoustic and visual data used.

E.1 Initialization

We initialized the parameters W by drawing each W_{id} from a Gaussian distribution with unit mean and standard deviation of $\frac{1}{3}$. Thereafter we normalized such that the average over all W_{id} was W^{init}. For MCA$_3$ we used $W^{\text{init}} = 4$ and for the R-MCA algorithms we used $W^{\text{init}} = 2$ (for this choice the sum, $\sum_d W_{id} = 50$, corresponds to the sum of the parameters used to generate the data). The parameters of the prior distribution were initialized to be $\pi_i = \frac{1}{H}$ for MCA$_3$, that is, half the value of the generating $\pi_i^{\text{gen}} = \frac{2}{H}$ in the standard bars test. For R-MCA$_2$ and R-MCA$_{\text{NN}}$ we initialized with values $\pi_i = \pi = \frac{2}{H}$. The reliability of both R-MCA algorithms is only marginally affected by the exact choices of π and W^{init}. Reliability values remained about the same even if the assumed values of π differed significantly from the generating values π_i^{gen} (see, e.g., the experiments of Figure 8 or the paragraph on 'Violations of model assumptions' in Experiments).

E.2 Annealing

In Equations (16) and (18) we make the following substitutions:

$$\mathcal{H}(x) \rightarrow \mathcal{S}_T(x) = \left(1 + \exp\left(-\tfrac{\lambda}{T-1}x\right)\right)^{-1}, \tag{35}$$

$$\overline{\pi}_i \rightarrow (\overline{\pi}_i)^{\beta}, \;\; I_i^{(n)} \rightarrow \beta I_i^{(n)}, \;\; I_{ab}^{(n)} \rightarrow \beta I_{ab}^{(n)}, \;\; I_{abc}^{(n)} \rightarrow \beta I_{abc}^{(n)}, \;\; \text{with} \;\; \beta = \tfrac{1}{T}. \tag{36}$$

while making only the substitutions of (36) in Equation (33). Here, T plays the role of a 'temperature'. In the limit of $T = 1$, β is equal to one and the sigmoidal function $\mathcal{S}_T(x)$ converges to the Heaviside function, that is, we recover the original Equations (16) and (17). A temperature $T > 1$ has the effect of leveling the differences between the parameter updates to a certain extent. For a high temperature $T \gg 1$, the differences between the parameters associated with different hidden variables vanish after a few iterations.

Smoothing the Heaviside function in (35) is a technique frequently used, for example, in the context of perceptrons, and the substitutions in (36) correspond to the standard annealing procedure for EM (Ueda and Nakano, 1998; Sahani, 1999). The slope of the sigmoidal function $\mathcal{S}_T(x)$ at $x = 0$ is parameterized by λ whose value is set to $\lambda = 0.2$.

In experiments for MCA$_3$ and R-MCA$_2$ we started learning at a relatively high temperature $T_1 > 1$ and cooled to a value T_o close to one. A final temperature $T_o > 1$ makes the system more robust and counteracts over-fitting (Weiss, 1998). Experimental results on artificial data remained essentially the same when we used $T_o = 1$ but the cooling procedure needed to be slower to avoid numerical instabilities in this case. If not otherwise stated, we used $T_o = 1.5$. For MCA$_3$ and R-MCA$_2$ we cooled from T_1 to T_o in steps of $\Delta T = \frac{T_1 - T_o}{50}$ after each iteration. However, we did not change the temperature if the parameters W still changed significantly. More precisely, we only decreased the temperature if the change in W_{id} fell below a threshold $\theta_{\Delta W}$ for all $i = 1, \ldots, H$. In formulas:

$$(\forall i: \;\; \Delta W_i \; < \; \theta_{\Delta W}) \quad \Rightarrow \quad T^{\text{new}} = T^{\text{old}} - \Delta T\,, \tag{37}$$

$$\text{where} \quad \Delta W_i = \frac{\sqrt{\sum_d (W_{id}^{\text{old}} - W_{id}^{\text{new}})^2}}{\sum_d W_{id}^{\text{old}}}\,. \tag{38}$$

Cooling conditioned on small parameter changes in this way allows the use of larger cooling steps ΔT and thus leads to learning in fewer iterations. For all trials we used a threshold of

$\theta_{\Delta W} = 2.5 \times 10^{-3}$, except for the application to more realistic data for which we ran additional trials with $\theta_{\Delta W} = 0.25 \times 10^{-3}$.

In experiments it was observed that a given system had a critical temperature T_c above which the weights did not specialize to different patterns after random initialization. Instead the parameters converged to about the same values for all hidden variables (compare Sahani, 1999; Lücke, 2004). A natural choice of an initial temperature $T_1 > 1$ is therefore a value close to this critical temperature. Experiments on different versions of the bars test showed a roughly linear dependence between the critical temperature T_c and the number of input dimensions D. Thus, in all versions of the bars test we used $T_1 = 0.4D + 1$ and $T_1 = 0.7D + 1$ for MCA$_3$ and R-MCA$_2$, respectively. In the experiments on acoustic and visual data, the critical temperatures are lower than those measured in bars tests with same D, presumably due to more homogeneous distributions of input values in those cases (generating weights in the bars test were all either 0 or 10, whereas generating weights in the naturally-derived data could take on any value in $[0, 10]$). Thus, experiments on phoneme data started at an initial temperature of $T_1 = 70$ for MCA$_3$ and $T_1 = 100$ for R-MCA$_2$; and those on visual data started at $T_1 = 2$ for MCA$_3$ (8-by-8 patches, $H = 20$) and $T_1 = 4$ for R-MCA$_2$ (10-by-10 patches, $H = 50$). In all experiments the temperature was maintained at T_1 during the first ten iterations. After the system had cooled to T_o using (37) and (38) learning was terminated once all ΔW_i remained smaller than $\theta_{\Delta W}$ for 20 iterations.

For R-MCA$_{\text{NN}}$ we used a fixed temperature of $T = 16$ if not otherwise stated, and stopped after all single causes were represented by the same hidden variables for 4000 pattern presentations. In a given trial, the first pattern presentation after which the representation did not change was taken as the learning time of R-MCA$_{\text{NN}}$. For the acoustic data set we used $T = 70$ and additionally report results for $T = 50$ and values $T \leq 40$. For the visual data we used $T = 2$.

For MCA$_{\text{ex}}$ in Figure 6 we have used a relatively fast and fixed cooling schedule.

E.3 Algorithms for the Comparison in Figure 9

For the comparison in Figure 9 we have reproduced data reported by Spratling (2006). While we have adopted the same abbreviations as were used there, we repeat them in Table 2 for the convenience of the reader.

Algorithm	Description
NN-SC	non-negative sparse coding ($\lambda = 1$)
SC-NMF$_{\text{A}}$	NMF with a sparseness constraint of 0.5 on the basis vectors
SC-NMF$_{\text{AY}}$	NMF with a sparseness constraint of 0.5 on the basis vectors and 0.7 on the activations
DI	dendritic inhibition network
NN-DI	dendritic inhibition network with non-negative weights
SC-NMF$_{\text{Y}}$	NMF with a sparseness constraint of 0.7 on the activations
S-NMF	sparse-NMF ($\alpha = 1$)
NMF$_{\text{div}}$	NMF with divergence objective
NMF$_{\text{mse}}$	NMF with Euclidean objective
L-NMF	local NMF

Table 2: Description of the algorithms used for the comparison in Figure 9.

E.4 Acoustic Data

The data used to study component extraction in the acoustic domain were generated from recordings of three vowels (two of them diphthongs), [aʊ], [i:], and [ɔɪ] and three consonants [k], [t], [p]. The phonemes were spoken by a male voice and recorded at 8000Hz. The data were taken from a publicly accessible data base (Sunsite, 1997). To construct spectrograms we used 1000 samples for each of the phonemes, which required truncation in three cases and padding with zeros in the other three cases. The waveforms $z(t)$ were normalized in power such that $\frac{1}{T}\sum_t (z(t))^2 = 1$. The waveforms were then linearly mixed ($z_{\text{mix}}(t) = z(t) + z'(t) + \ldots$) to produce $N = 500$ observed spectrograms. The probability of a phoneme of appearing in a mixed waveform was set to $\frac{2}{6}$. The spectrograms of these mixtures were computed using short-time Fourier transforms with 50 frequency channels ranging from 100 to 4000 Hz, with logarithmic scaling of center frequencies (see Figure 11D). We used 9 Hamming windows of 200 samples each, with successive windows overlapping by 100 samples. We then took the logarithms of the magnitudes of the 50-by-9 spectrogram entries, and linearly rescaled the top 42.8 dB of dynamic range to lie between 0 and 10, with magnitudes more than 42.8 dB below the highest intensity being clipped to 0.

E.5 Visual Data

The image that was used for the experiments on visual data has been taken from the publicly available image database of the van Hateren group at `hlab.phys.rug.nl/imlib/`. Images of the database represent light intensities linearly, which results in most images appearing relatively dark if displayed using a finite gray scale (see van Hateren and van der Schaaf, 1998, for details). We have used image number 2338 (deblurred), cut out a segment of 500-by-500 pixels in the lower left corner and scaled it down to a resolution of 250-by-250 pixels.

References

A. J. Bell and T. J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

N. J. Butko and J. Triesch. Learning sensory representations with intrinsic plasticity. *Neurocomputing*, 70(7-9):1130–1138, 2007.

D. Charles and C. Fyfe. Modelling multiple-cause structure using rectification constraints. *Network: Computation in Neural Systems*, 9:167–182, 1998.

D. Charles, C. Fyfe, D. MacDonald, and J. Koetsier. Unsupervised neural networks for the identification of minimum overcomplete basis in visual data. *Neurocomputing*, 47(1-4):119–143, 2002.

P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

P. Dayan and L. F. Abbott. *Theoretical Neuroscience*. MIT Press, Cambridge, 2001.

P. Dayan and R. S. Zemel. Competition and multiple cause models. *Neural Computation*, 7:565–579, 1995.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

B. S. Everitt. *An Introduction to Latent Variable Models*. Chapman and Hall, 1984.

P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64:165–170, 1990.

C. Fyfe. A neural network for PCA and beyond. *Neural Processing Letters*, 6:33–41, 1997.

N. M. Grzywacz and A. L. Yuille. A model for the estimate of local image velocity by cells in the visual cortex. *Proceedings of the Royal Society of London B*, 239:129–161, 1990.

G. F. Harpur and R. W. Prager. Development of low entropy coding in a recurrent network. *Network: Computation in Neural Systems*, 7:277–284, 1996.

G. E. Hinton and Z. Ghahramani. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society B*, 352:1177–1190, 1997.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The 'wake-sleep' algorithm for unsupervised neural networks. *Science*, 268:1158–1161, 1995.

S. Hochreiter and J. Schmidhuber. Feature extraction through LOCOCODE. *Neural Computation*, 11:679–714, 1999.

P. O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing XII: Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565. 2002.

P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*, pages 556–562, 2001.

J. Lücke. Hierarchical self-organization of minicolumnar receptive fields. *Neural Networks*, 17/8–9: 1377–1389, 2004.

J. Lücke. A dynamical model for receptive field self-organization in V1 cortical columns. In *Proceedings of the International Conference on Artificial Neural Networks*, LNCS 4669, pages 389–398. Springer, 2007.

J. Lücke and J. D. Bouecke. Dynamics of cortical columns – self-organization of receptive fields. In *Proceedings of the International Conference on Artificial Neural Networks*, LNCS 3696, pages 31–37. Springer, 2005.

J. Lücke and M. Sahani. Generalized softmax networks for non-linear component extraction. In *Proceedings of the International Conference on Artificial Neural Networks*, LNCS 4668, pages 657–667. Springer, 2007.

J. Lücke and C. von der Malsburg. Rapid processing and unsupervised learning in a model of the cortical macrocolumn. *Neural Computation*, 16:501–533, 2004.

G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000.

R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

S. J. Nowlan. Maximum likelihood competitive learning. In *Advances in Neural Information Processing Systems 2*, pages 574–582, 1990.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

R. C. O'Reilly. Generalization in interactive networks: The benefits of inhibitory competition and Hebbian learning. *Neural Computation*, 13:1199–1241, 2001.

R. P. N. Rao, B. A. Olshausen, and M. S. Lewicki, editors. *Probabilistic Models of the Brain: Perception and Neural Function*. Neural Information Processing. The MIT Press, Cambridge, MA, 2002.

M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.

S. T. Roweis. Factorial models and refiltering for speech separation and denoising. In *Proceedings Eurospeech*, volume 7, pages 1009–1012, 2003.

M. Sahani. *Latent Variable Models for Neural Data Analysis*. PhD thesis, California Institute of Technology, Pasadena, California, 1999. URL `http://www.gatsby.ucl.ac.uk/~maneesh/thesis/`.

E. Saund. A multiple cause mixture model for unsupervised learning. *Neural Computation*, 7: 51–71, 1995.

M. W. Spratling. Learning image components for object recognition. *Journal of Machine Learning Research*, 7:793–815, 2006.

M. W. Spratling and M. H. Johnson. Preintegration lateral inhibition enhances unsupervised learning. *Neural Computation*, 14:2157–2179, 2002.

Sunsite. Sun-sounds: Phonemes. Data retrieved in 2007 through `ibiblio.org` from `ftp://sunsite.unc.edu/pub/multimedia/sun-sounds/phonemes/`, 1997.

N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282, 1998.

J. H. van Hateren and A. van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society of London B*, 265: 359–366, 1998.

Y. Weiss. Phase transitions and the perceptual organization of video sequences. In *Advances in Neural Information Processing Systems 10*, pages 850–856, 1998.

H. Wersing and E. Körner. Learning optimized features for hierarchical models of invariant object recognition. *Neural Computation*, 15(7):1559–1588, 2003.

A. L. Yuille and D. Geiger. Winner-take-all networks. In M.A. Arbib, editor, *The handbook of brain theory and neural networks*, pages 1228–1231. MIT Press, 2003.

Journal of Machine Learning Research 9 (2008) 1269-1294 Submitted 11/07; Revised 5/08; Published 7/08

Optimal Solutions for Sparse Principal Component Analysis

Alexandre d'Aspremont ASPREMON@PRINCETON.EDU
ORFE, Princeton University
Princeton, NJ 08544, USA

Francis 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

Laurent El Ghaoui ELGHAOUI@EECS.BERKELEY.EDU
EECS Department, U.C. Berkeley
Berkeley, CA 94720, USA

Editor: Aapo Hyvarinen

Abstract

Given a sample covariance matrix, we examine the problem of maximizing the variance explained by a linear combination of the input variables while constraining the number of nonzero coefficients in this combination. This is known as sparse principal component analysis and has a wide array of applications in machine learning and engineering. We formulate a new semidefinite relaxation to this problem and derive a greedy algorithm that computes a *full set* of good solutions for all target numbers of non zero coefficients, with total complexity $O(n^3)$, where n is the number of variables. We then use the same relaxation to derive sufficient conditions for global optimality of a solution, which can be tested in $O(n^3)$ per pattern. We discuss applications in subset selection and sparse recovery and show on artificial examples and biological data that our algorithm does provide globally optimal solutions in many cases.

Keywords: PCA, subset selection, sparse eigenvalues, sparse recovery, lasso

1. Introduction

Principal component analysis (PCA) is a classic tool for data analysis, visualization or compression and has a wide range of applications throughout science and engineering. Starting from a multivariate data set, PCA finds linear combinations of the variables called *principal components*, corresponding to orthogonal directions maximizing variance in the data. Numerically, a full PCA involves a singular value decomposition of the data matrix.

One of the key shortcomings of PCA is that the factors are linear combinations of *all* original variables; that is, most of factor coefficients (or loadings) are non-zero. This means that while PCA facilitates model interpretation and visualization by concentrating the information in a few factors, the factors themselves are still constructed using all variables, hence are often hard to interpret.

In many applications, the coordinate axes involved in the factors have a direct physical interpretation. In financial or biological applications, each axis might correspond to a specific asset or gene. In problems such as these, it is natural to seek a trade-off between the two goals of *statistical fidelity* (explaining most of the variance in the data) and *interpretability* (making sure that the

factors involve only a few coordinate axes). Solutions that have only a few nonzero coefficients in the principal components are usually easier to interpret. Moreover, in some applications, nonzero coefficients have a direct cost (e.g., transaction costs in finance) hence there may be a direct trade-off between statistical fidelity and practicality. Our aim here is to efficiently derive *sparse principal components*, that is, a set of sparse vectors that explain a maximum amount of variance. Our belief is that in many applications, the decrease in statistical fidelity required to obtain sparse factors is small and relatively benign.

In what follows, we will focus on the problem of finding sparse factors which explain a maximum amount of variance, which can be written:

$$\max_{\|z\|\leq 1} z^T \Sigma z - \rho \, \mathbf{Card}(z) \tag{1}$$

in the variable $z \in \mathbf{R}^n$, where $\Sigma \in \mathbf{S}_n$ is the (symmetric positive semi-definite) sample covariance matrix, ρ is a parameter controlling sparsity, and $\mathbf{Card}(z)$ denotes the cardinal (or ℓ_0 norm) of z, that is, the number of non zero coefficients of z.

While PCA is numerically easy, each factor requires computing a leading eigenvector, which can be done in $O(n^2)$, sparse PCA is a hard combinatorial problem. In fact, Moghaddam et al. (2006b) show that the subset selection problem for ordinary least squares, which is NP-hard (Natarajan, 1995), can be reduced to a sparse generalized eigenvalue problem, of which sparse PCA is a particular intance. Sometimes factor rotation techniques are used to post-process the results from PCA and improve interpretability (see QUARTIMAX by Neuhaus and Wrigley 1954, VARIMAX by Kaiser 1958 or Jolliffe 1995 for a discussion). Another simple solution is to *threshold* the loadings with small absolute value to zero (Cadima and Jolliffe, 1995). A more systematic approach to the problem arose in recent years, with various researchers proposing nonconvex algorithms (e.g., SCoTLASS by Jolliffe et al. 2003, SLRA by Zhang et al. 2002 or D.C. based methods such as (Sriperumbudur et al., 2007) which find modified principal components with zero loadings). The SPCA algorithm, which is based on the representation of PCA as a regression-type optimization problem (Zou et al., 2006), allows the application of the LASSO (Tibshirani, 1996), a penalization technique based on the ℓ_1 norm. With the exception of simple thresholding, all the algorithms above require solving non convex problems. Recently also, d'Aspremont et al. (2007b) derived an ℓ_1 based semidefinite relaxation for the sparse PCA problem (1) with a complexity of $O(n^4\sqrt{\log n})$ for a given ρ. Finally, Moghaddam et al. (2006a) used greedy search and branch-and-bound methods to solve small instances of problem (1) exactly and get good solutions for larger ones. Each step of this greedy algorithm has complexity $O(n^3)$, leading to a total complexity of $O(n^4)$ for a full set of solutions. Moghaddam et al. (2007) improve this bound in the regression/discrimination case.

Our contribution here is twofold. We first derive a greedy algorithm for computing a *full set* of good solutions (one for each target sparsity between 1 and n) at a total numerical cost of $O(n^3)$ based on the convexity of the of the largest eigenvalue of a symmetric matrix. We then derive *tractable* sufficient conditions for a vector z to be a *global* optimum of (1). This means in practice that, given a vector z with support I, we can test if z is a globally optimal solution to problem (1) by performing a few binary search iterations to solve a one dimensional convex minimization problem. In fact, we can take any sparsity pattern candidate from any algorithm and test its optimality. This paper builds on the earlier conference version (d'Aspremont et al., 2007a), providing new and simpler conditions for optimality and describing applications to subset selection and sparse recovery.

While there is certainly a case to be made for ℓ_1 penalized maximum eigenvalues (à la d'Aspremont et al., 2007b), we strictly focus here on the ℓ_0 formulation. However, it was shown recently (see

Candès and Tao 2005, Donoho and Tanner 2005 or Meinshausen and Yu 2006 among others) that there is in fact a deep connection between ℓ_0 constrained extremal eigenvalues and LASSO type variable selection algorithms. Sufficient conditions based on sparse eigenvalues (also called restricted isometry constants in Candès and Tao 2005) guarantee consistent variable selection (in the LASSO case) or sparse recovery (in the decoding problem). The results we derive here produce upper bounds on sparse extremal eigenvalues and can thus be used to prove consistency in LASSO estimation, prove perfect recovery in sparse recovery problems, or prove that a particular solution of the subset selection problem is optimal. Of course, our conditions are only sufficient, not necessary and the duality bounds we produce on sparse extremal eigenvalues cannot always be tight, but we observe that the duality gap is often small.

The paper is organized as follows. We begin by formulating the sparse PCA problem in Section 2. In Section 3, we write an efficient algorithm for computing a full set of candidate solutions to problem (1) with total complexity $O(n^3)$. In Section 4 we then formulate a convex relaxation for the sparse PCA problem, which we use in Section 5 to derive tractable sufficient conditions for the global optimality of a particular sparsity pattern. In Section 6 we detail applications to subset selection, sparse recovery and variable selection. Finally, in Section 7, we test the numerical performance of these results.

1.1 Notation

For a vector $z \in \mathbf{R}$, we let $\|z\|_1 = \sum_{i=1}^n |z_i|$ and $\|z\| = \left(\sum_{i=1}^n z_i^2\right)^{1/2}$, $\mathbf{Card}(z)$ is the cardinality of z, that is, the number of nonzero coefficients of z, while the support I of z is the set $\{i : z_i \neq 0\}$ and we use I^c to denote its complement. For $\beta \in \mathbf{R}$, we write $\beta_+ = \max\{\beta, 0\}$ and for $X \in \mathbf{S}_n$ (the set of symmetric matrix of size $n \times n$) with eigenvalues λ_i, $\mathbf{Tr}(X)_+ = \sum_{i=1}^n \max\{\lambda_i, 0\}$. The vector of all ones is written $\mathbf{1}$, while the identity matrix is written $\mathbf{I}$. The diagonal matrix with the vector u on the diagonal is written $\mathbf{diag}(u)$.

2. Sparse PCA

Let $\Sigma \in \mathbf{S}_n$ be a symmetric matrix. We consider the following sparse PCA problem:

$$\phi(\rho) \equiv \max_{\|z\| \leq 1} z^T \Sigma z - \rho\, \mathbf{Card}(z) \tag{2}$$

in the variable $z \in \mathbf{R}^n$ where $\rho > 0$ is a parameter controlling sparsity. We assume without loss of generality that $\Sigma \in \mathbf{S}_n$ is positive semidefinite and that the n variables are ordered by decreasing marginal variances, that is, that $\Sigma_{11} \geq \ldots \geq \Sigma_{nn}$. We also assume that we are given a square root A of the matrix Σ with $\Sigma = A^T A$, where $A \in \mathbf{R}^{n \times n}$ and we denote by $a_1, \ldots, a_n \in \mathbf{R}^n$ the columns of A. Note that the problem and our algorithms are invariant by permutations of Σ and by the choice of square root A. In practice, we are very often given the data matrix A instead of the covariance Σ.

A problem that is directly related to (2) is that of computing a cardinality constrained maximum eigenvalue, by solving:

$$\begin{array}{ll} \text{maximize} & z^T \Sigma z \\ \text{subject to} & \mathbf{Card}(z) \leq k \\ & \|z\| = 1, \end{array} \tag{3}$$

in the variable $z \in \mathbf{R}^n$. Of course, this problem and (2) are related. By duality, an upper bound on the optimal value of (3) is given by:

$$\inf_{\rho \in P} \phi(\rho) + \rho k.$$

where P is the set of penalty values for which $\phi(\rho)$ has been computed. This means in particular that if a point z is provably optimal for (2), it is also globally optimum for (3) with $k = \mathbf{Card}(z)$.

We now begin by reformulating (2) as a relatively simple convex maximization problem. Suppose that $\rho \geq \Sigma_{11}$. Since $z^T \Sigma z \leq \Sigma_{11}(\sum_{i=1}^n |z_i|)^2$ and $(\sum_{i=1}^n |z_i|)^2 \leq \|z\|^2 \mathbf{Card}(z)$ for all $z \in \mathbf{R}^n$, we have:

$$\begin{array}{rl} \phi(\rho) & = \max_{\|z\| \leq 1} z^T \Sigma z - \rho \, \mathbf{Card}(z) \\ & \leq (\Sigma_{11} - \rho) \, \mathbf{Card}(z) \\ & \leq 0, \end{array}$$

hence the optimal solution to (2) when $\rho \geq \Sigma_{11}$ is $z = 0$. From now on, we assume $\rho \leq \Sigma_{11}$ in which case the inequality $\|z\| \leq 1$ is tight. We can represent the sparsity pattern of a vector z by a vector $u \in \{0,1\}^n$ and rewrite (2) in the equivalent form:

$$\begin{array}{rl} \phi(\rho) & = \max\limits_{u \in \{0,1\}^n} \lambda_{\max}(\mathbf{diag}(u) \Sigma \, \mathbf{diag}(u)) - \rho \mathbf{1}^T u \\ & = \max\limits_{u \in \{0,1\}^n} \lambda_{\max}(\mathbf{diag}(u) A^T A \, \mathbf{diag}(u)) - \rho \mathbf{1}^T u \\ & = \max\limits_{u \in \{0,1\}^n} \lambda_{\max}(A \, \mathbf{diag}(u) A^T) - \rho \mathbf{1}^T u, \end{array}$$

using the fact that $\mathbf{diag}(u)^2 = \mathbf{diag}(u)$ for all variables $u \in \{0,1\}^n$ and that for any matrix B, $\lambda_{\max}(B^T B) = \lambda_{\max}(B B^T)$. We then have:

$$\begin{array}{rl} \phi(\rho) & = \max\limits_{u \in \{0,1\}^n} \lambda_{\max}(A \, \mathbf{diag}(u) A^T) - \rho \mathbf{1}^T u \\ & = \max\limits_{\|x\|=1} \max\limits_{u \in \{0,1\}^n} x^T A \, \mathbf{diag}(u) A^T x - \rho \mathbf{1}^T u \\ & = \max\limits_{\|x\|=1} \max\limits_{u \in \{0,1\}^n} \sum\limits_{i=1}^n u_i ((a_i^T x)^2 - \rho). \end{array}$$

Hence we finally get, after maximizing in u (and using $\max_{v \in \{0,1\}} \beta v = \beta_+$):

$$\phi(\rho) = \max_{\|x\|=1} \sum_{i=1}^n ((a_i^T x)^2 - \rho)_+, \tag{4}$$

which is a nonconvex problem in the variable $x \in \mathbf{R}^n$. We then select variables i such that $(a_i^T x)^2 - \rho > 0$. Note that if $\Sigma_{ii} = a_i^T a_i < \rho$, we must have $(a_i^T x)^2 \leq \|a_i\|^2 \|x\|^2 < \rho$ hence variable i will never be part of the optimal subset and we can remove it.

3. Greedy Solutions

In this section, we focus on finding a good solution to problem (2) using greedy methods. We first present very simple preprocessing solutions with complexity $O(n \log n)$ and $O(n^2)$. We then recall a simple greedy algorithm with complexity $O(n^4)$. Finally, our first contribution in this section is to derive an approximate greedy algorithm that computes a full set of (approximate) solutions for problem (2), with total complexity $O(n^3)$.

3.1 Sorting and Thresholding

The simplest ranking algorithm is to sort the diagonal of the matrix Σ and rank the variables by variance. This works intuitively because the diagonal is a rough proxy for the eigenvalues: the Schur-Horn theorem states that the diagonal of a matrix majorizes its eigenvalues (Horn and Johnson, 1985); sorting costs $O(n \log n)$. Another quick solution is to compute the leading eigenvector of Σ and form a sparse vector by thresholding to zero the coefficients whose magnitude is smaller than a certain level. This can be done with cost $O(n^2)$.

3.2 Full Greedy Solution

Following Moghaddam et al. (2006a), starting from an initial solution of cardinality one at $\rho = \Sigma_{11}$, we can update an increasing sequence of index sets $I_k \subseteq [1,n]$, scanning all the remaining variables to find the index with maximum variance contribution.

Greedy Search Algorithm.

- **Input**: $\Sigma \in \mathbf{R}^{n \times n}$
- **Algorithm**:
  1. Preprocessing: sort variables by decreasing diagonal elements and permute elements of Σ accordingly. Compute the Cholesky decomposition $\Sigma = A^T A$.
  2. Initialization: $I_1 = \{1\}$, $x_1 = a_1 / \|a_1\|$.
  3. Compute $i_k = \operatorname{argmax}_{i \notin I_k} \lambda_{\max} \left(\sum_{j \in I_k \cup \{i\}} a_j a_j^T \right)$.
  4. Set $I_{k+1} = I_k \cup \{i_k\}$ and compute x_{k+1} as the leading eigenvector of $\sum_{j \in I_{k+1}} a_j a_j^T$.
  5. Set $k = k+1$. If $k < n$ go back to step 3.
- **Output**: sparsity patterns I_k.

At every step, I_k represents the set of nonzero elements (or sparsity pattern) of the current point and we can define z_k as the solution to problem (2) given I_k, which is:

$$z_k = \operatorname*{argmax}_{\{z_{I_k^c}=0,\ \|z\|=1\}} z^T \Sigma z - \rho k,$$

which means that z_k is formed by padding zeros to the leading eigenvector of the submatrix Σ_{I_k,I_k}. Note that the entire algorithm can be written in terms of a factorization $\Sigma = A^T A$ of the matrix Σ, which means significant computational savings when Σ is given as a Gram matrix. The matrices Σ_{I_k,I_k} and $\sum_{i \in I_k} a_i a_i^T$ have the same eigenvalues and their eigenvectors are transformed of each other through the matrix A, that is, if z is an eigenvector of Σ_{I_k,I_k}, then $A_{I_k} z / \|A_{I_k} z\|$ is an eigenvector of $A_{I_k} A_{I_k}^T$.

3.3 Approximate Greedy Solution

Computing $n-k$ eigenvalues at each iteration is costly and we can use the fact that uu^T is a subgradient of $\lambda_{\max}$ at X if u is a leading eigenvector of X (Boyd and Vandenberghe, 2004), to get:

$$\lambda_{\max} \left(\sum_{j \in I_k \cup \{i\}} a_j a_j^T \right) \geq \lambda_{\max} \left(\sum_{j \in I_k} a_j a_j^T \right) + (x_k^T a_i)^2,$$

which means that the variance is increasing by at least $(x_k^T a_i)^2$ when variable i is added to I_k. This provides a lower bound on the objective which does not require finding $n-k$ eigenvalues at each iteration. We then derive the following algorithm:

Approximate Greedy Search Algorithm.

- **Input**: $\Sigma \in \mathbf{R}^{n\times n}$
- **Algorithm**:
  1. Preprocessing. Sort variables by decreasing diagonal elements and permute elements of Σ accordingly. Compute the Cholesky decomposition $\Sigma = A^T A$.
  2. Initialization: $I_1 = \{1\}$, $x_1 = a_1/\|a_1\|$.
  3. Compute $i_k = \mathrm{argmax}_{i\notin I_k} (x_k^T a_i)^2$
  4. Set $I_{k+1} = I_k \cup \{i_k\}$ and compute x_{k+1} as the leading eigenvector of $\sum_{j\in I_{k+1}} a_j a_j^T$.
  5. Set $k = k+1$. If $k < n$ go back to step 3.
- **Output**: sparsity patterns I_k.

Again, at every step, I_k represents the set of nonzero elements (or sparsity pattern) of the current point and we can define z_k as the solution to problem (2) given I_k, which is:

$$z_k = \underset{\{z_{I_k^c}=0,\ \|z\|=1\}}{\mathrm{argmax}} \ z^T \Sigma z - \rho k,$$

which means that z_k is formed by padding zeros to the leading eigenvector of the submatrix Σ_{I_k,I_k}. Better points can be found by testing the variables corresponding to the p largest values of $(x_k^T a_i)^2$ instead of picking only the best one.

3.4 Computational Complexity

The complexity of computing a greedy regularization path using the classic greedy algorithm in Section 3.2 is $O(n^4)$: at each step k, it computes $(n-k)$ maximum eigenvalue of matrices with size k. The approximate algorithm in Section 3.3 computes a full path in $O(n^3)$: the first Cholesky decomposition is $O(n^3)$, while the complexity of the k-th iteration is $O(k^2)$ for the maximum eigenvalue problem and $O(n^2)$ for computing all products $(x^T a_j)$. Also, when the matrix Σ is directly given as a Gram matrix $A^T A$ with $A \in \mathbf{R}^{q\times n}$ with $q < n$, it is advantageous to use A directly as the square root of Σ and the total complexity of getting the path up to cardinality p is then reduced to $O(p^3 + p^2 n)$ (which is $O(p^3)$ for the eigenvalue problems and $O(p^2 n)$ for computing the vector products).

4. Convex Relaxation

In Section 2, we showed that the original sparse PCA problem (2) could also be written as in (4):

$$\phi(\rho) = \max_{\|x\|=1} \sum_{i=1}^{n} ((a_i^T x)^2 - \rho)_+.$$

Because the variable x appears solely through $X = xx^T$, we can reformulate the problem in terms of X only, using the fact that when $\|x\| = 1$, $X = xx^T$ is equivalent to $\mathbf{Tr}(X) = 1$, $X \succeq 0$ and $\mathbf{Rank}(X) = 1$. We thus rewrite (4) as:

$$\begin{array}{rll} \phi(\rho) = & \max. & \sum_{i=1}^n (a_i^T X a_i - \rho)_+ \\ & \text{s.t.} & \mathbf{Tr}(X) = 1, \ \mathbf{Rank}(X) = 1 \\ & & X \succeq 0. \end{array}$$

Note that because we are maximizing a convex function over the convex set (spectahedron) $\Delta_n = \{X \in \mathbf{S}_n : \ \mathbf{Tr}(X) = 1, \ X \succeq 0\}$, the solution must be an extreme point of Δ_n (i.e., a rank one matrix), hence we can drop the rank constraint here. Unfortunately, $X \mapsto (a_i^T X a_i - \rho)_+$, the function we are *maximizing*, is convex in X and not concave, which means that the above problem is still hard. However, we show below that on rank one elements of Δ_n, it is also equal to a concave function of X, and we use this to produce a semidefinite relaxation of problem (2).

Proposition 1 *Let $A \in \mathbf{R}^{n\times n}$, $\rho \geq 0$ and denote by $a_1, \ldots, a_n \in \mathbf{R}^n$ the columns of A, an upper bound on:*

$$\begin{array}{rll} \phi(\rho) = & \textit{max.} & \sum_{i=1}^n (a_i^T X a_i - \rho)_+ \\ & \textit{s.t.} & \mathbf{Tr}(X) = 1, \ X \succeq 0, \ \mathbf{Rank}(X) = 1 \end{array}$$

can be computed by solving

$$\begin{array}{rll} \psi(\rho) = & \textit{max.} & \sum_{i=1}^n \mathbf{Tr}(X^{1/2} B_i X^{1/2})_+ \\ & \textit{s.t.} & \mathbf{Tr}(X) = 1, \ X \succeq 0. \end{array} \tag{5}$$

in the variables $X \in \mathbf{S}_n$, where $B_i = a_i a_i^T - \rho \mathbf{I}$, or also:

$$\begin{array}{rll} \psi(\rho) = & \textit{max.} & \sum_{i=1}^n \mathbf{Tr}(P_i B_i) \\ & \textit{s.t.} & \mathbf{Tr}(X) = 1, \ X \succeq 0, \ X \succeq P_i \succeq 0, \end{array} \tag{6}$$

which is a semidefinite program in the variables $X \in \mathbf{S}_n, \ P_i \in \mathbf{S}_n$.

Proof We let $X^{1/2}$ denote the positive square root (i.e., with nonnegative eigenvalues) of a symmetric positive semi-definite matrix X. In particular, if $X = xx^T$ with $\|x\| = 1$, then $X^{1/2} = X = xx^T$, and for all $\beta \in \mathbf{R}$, βxx^T has one eigenvalue equal to β and $n-1$ equal to 0, which implies $\mathbf{Tr}(\beta xx^T)_+ = \beta_+$. We thus get:

$$\begin{array}{rcl} (a_i^T X a_i - \rho)_+ & = & \mathbf{Tr}((a_i^T x x^T a_i - \rho) x x^T)_+ \\ & = & \mathbf{Tr}(x(x^T a_i a_i^T x - \rho) x^T)_+ \\ & = & \mathbf{Tr}(X^{1/2} a_i a_i^T X^{1/2} - \rho X)_+ = \mathbf{Tr}(X^{1/2}(a_i a_i^T - \rho \mathbf{I}) X^{1/2})_+. \end{array}$$

For any symmetric matrix B, the function $X \mapsto \mathbf{Tr}(X^{1/2} B X^{1/2})_+$ is concave on the set of symmetric positive semidefinite matrices, because we can write it as:

$$\begin{array}{rcl} \mathbf{Tr}(X^{1/2} B X^{1/2})_+ & = & \max\limits_{\{0 \preceq P \preceq X\}} \mathbf{Tr}(PB) \\ & = & \min\limits_{\{Y \succeq B, \ Y \succeq 0\}} \mathbf{Tr}(YX), \end{array}$$

where this last expression is a concave function of X as a pointwise minimum of affine functions. We can now relax the original problem into a convex optimization problem by simply dropping the rank constraint, to get:

$$\begin{array}{rll} \psi(\rho) \equiv & \max. & \sum_{i=1}^n \mathbf{Tr}(X^{1/2} a_i a_i^T X^{1/2} - \rho X)_+ \\ & \text{s.t.} & \mathbf{Tr}(X) = 1,\ X \succeq 0, \end{array}$$

which is a convex program in $X \in \mathbf{S}_n$. Note that because B_i has at most one nonnegative eigenvalue, we can replace $\mathbf{Tr}(X^{1/2} a_i a_i^T X^{1/2} - \rho X)_+$ by $\lambda_{\max}(X^{1/2} a_i a_i^T X^{1/2} - \rho X)_+$ in the above program. Using the representation of $\mathbf{Tr}(X^{1/2} B X^{1/2})_+$ detailed above, problem (5) can be written as a semidefinite program:

$$\begin{array}{rll} \psi(\rho) = & \max. & \sum_{i=1}^n \mathbf{Tr}(P_i B_i) \\ & \text{s.t.} & \mathbf{Tr}(X) = 1,\ X \succeq 0,\ X \succeq P_i \succeq 0, \end{array}$$

in the variables $X \in \mathbf{S}_n,\ P_i \in \mathbf{S}_n$, which is the desired result. ■

Note that we always have $\psi(\rho) \geq \phi(\rho)$ and when the solution to the above semidefinite program has rank one, $\psi(\rho) = \phi(\rho)$ and the semidefinite relaxation (6) is *tight*. This simple fact allows us to derive sufficient global optimality conditions for the original sparse PCA problem.

5. Optimality Conditions

In this section, we derive necessary and sufficient conditions to test the optimality of solutions to the relaxations obtained in Sections 3, as well as sufficient condition for the tightness of the semidefinite relaxation in (6).

5.1 Dual Problem and Optimality Conditions

We first derive the dual problem to (6) as well as the Karush-Kuhn-Tucker (KKT) optimality conditions:

Lemma 2 *Let $A \in \mathbf{R}^{n\times n}$, $\rho \geq 0$ and denote by $a_1, \ldots, a_n \in \mathbf{R}^n$ the columns of A. The dual of problem (6):*

$$\begin{array}{rll} \psi(\rho) = & \textit{max.} & \sum_{i=1}^n \mathbf{Tr}(P_i B_i) \\ & \textit{s.t.} & \mathbf{Tr}(X) = 1,\ X \succeq 0,\ X \succeq P_i \succeq 0, \end{array}$$

in the variables $X \in \mathbf{S}_n,\ P_i \in \mathbf{S}_n$, is given by:

$$\begin{array}{ll} \textit{min.} & \lambda_{\max}\left(\sum_{i=1}^n Y_i\right) \\ \textit{s.t.} & Y_i \succeq B_i,\ Y_i \succeq 0, \quad i = 1, \ldots, n. \end{array} \tag{7}$$

in the variables $Y_i \in \mathbf{S}_n$. Furthermore, the KKT optimality conditions for this pair of semidefinite programs are given by:

$$\left\{ \begin{array}{l} \left(\sum_{i=1}^n Y_i\right) X = \lambda_{\max}\left(\sum_{i=1}^n Y_i\right) X \\ (X - P_i) Y_i = 0,\ P_i B_i = P_i Y_i \\ Y_i \succeq B_i,\ Y_i, X, P_i \succeq 0,\ X \succeq P_i,\ \mathbf{Tr}\, X = 1. \end{array} \right. \tag{8}$$

Proof Starting from:

$$\begin{array}{ll} \max. & \sum_{i=1}^n \mathbf{Tr}(P_i B_i) \\ \text{s.t.} & 0 \preceq P_i \preceq X \\ & \mathbf{Tr}(X) = 1,\ X \succeq 0, \end{array}$$

we can form the Lagrangian as:

$$L(X, P_i, Y_i) = \sum_{i=1}^n \mathbf{Tr}(P_i B_i) + \mathbf{Tr}(Y_i(X - P_i))$$

in the variables $X, P_i, Y_i \in \mathbf{S}_n$, with $X, P_i, Y_i \succeq 0$ and $\mathbf{Tr}(X) = 1$. Maximizing $L(X, P_i, Y_i)$ in the primal variables X and P_i leads to problem (7). The KKT conditions for this primal-dual pair of SDP can be derived from Boyd and Vandenberghe (2004, p.267).

■

5.2 Optimality Conditions for Rank One Solutions

We now derive the KKT conditions for problem (6) for the particular case where we are given a rank one candidate solution $X = xx^T$ and need to test its optimality. These necessary and sufficient conditions for the optimality of $X = xx^T$ for the convex relaxation then provide sufficient conditions for *global* optimality for the non-convex problem (2).

Lemma 3 *Let $A \in \mathbf{R}^{n\times n}$, $\rho \geq 0$ and denote by $a_1, \ldots, a_n \in \mathbf{R}^n$ the columns of A. The rank one matrix $X = xx^T$ is an optimal solution of (6) if and only if there are matrices $Y_i \in \mathbf{S}_n,\ i = 1, \ldots, n$ such that:*

$$\begin{cases} \lambda_{\max}\left(\sum_{i=1}^n Y_i\right) = \sum_{i\in I}((a_i^T x)^2 - \rho) \\ x^T Y_i x = \begin{cases} (a_i^T x)^2 - \rho \text{ if } i \in I \\ 0 \text{ if } i \in I^c \end{cases} \\ Y_i \succeq B_i,\ Y_i \succeq 0. \end{cases}$$

where $B_i = a_i a_i^T - \rho \mathbf{I},\ i = 1, \ldots, n$ and I^c is the complement of the set I defined by:

$$\max_{i\notin I}(a_i^T x)^2 \leq \rho \leq \min_{i\in I}(a_i^T x)^2.$$

Furthermore, x must be a leading eigenvector of both $\sum_{i\in I} a_i a_i^T$ and $\sum_{i=1}^n Y_i$.

Proof We apply Lemma 2 given $X = xx^T$. The condition $0 \preceq P_i \preceq xx^T$ is equivalent to $P_i = \alpha_i xx^T$ and $\alpha_i \in [0,1]$. The equation $P_i B_i = X Y_i$ is then equivalent to $\alpha_i(x^T B_i x - x^T Y_i x) = 0$, with $x^T B_i x = (a_i^T x)^2 - \rho$ and the condition $(X - P_i)Y_i = 0$ becomes $x^T Y_i x(1 - \alpha_i) = 0$. This means that $x^T Y_i x = ((a_i^T x)^2 - \rho)_+$ and the first-order condition in (8) becomes $\lambda_{\max}\left(\sum_{i=1}^n Y_i\right) = x^T\left(\sum_{i=1}^n Y_i\right)x$. Finally, we recall from Section 2 that:

$$\begin{aligned} \sum_{i\in I}((a_i^T x)^2 - \rho) &= \max_{\|x\|=1} \max_{u\in\{0,1\}^n} \sum_{i=1}^n u_i((a_i^T x)^2 - \rho) \\ &= \max_{u\in\{0,1\}^n} \lambda_{\max}(A\,\mathbf{diag}(u)A^T) - \rho \mathbf{1}^T u \end{aligned}$$

hence x must also be a leading eigenvector of $\sum_{i\in I} a_i a_i^T$. ■

The previous lemma shows that given a candidate vector x, we can test the optimality of $X = xx^T$ for the semidefinite program (5) by solving a semidefinite feasibility problem in the variables $Y_i \in \mathbf{S}_n$. If this (rank one) solution xx^T is indeed optimal for the semidefinite relaxation, then x must also be *globally* optimal for the original nonconvex combinatorial problem in (2), so the above lemma provides sufficient global optimality conditions for the combinatorial problem (2) based on the (necessary and sufficient) optimality conditions for the convex relaxation (5) given in lemma 2. In practice, we are only given a sparsity pattern I (using the results of Section 3 for example) rather than the vector x, but Lemma 3 also shows that given I, we can get the vector x as the leading eigenvector of $\sum_{i\in I} a_i a_i^T$.

The next result provides more refined conditions under which such a pair (I,x) is optimal for some value of the penalty $\rho > 0$ based on a local optimality argument. In particular, they allow us to fully specify the dual variables Y_i for $i \in I$.

Proposition 4 *Let $A \in \mathbf{R}^{n\times n}$, $\rho \geq 0$ and denote by $a_1,\ldots,a_n \in \mathbf{R}^n$ the columns of A. Let x be the largest eigenvector of $\sum_{i\in I} a_i a_i^T$. Let I be such that:*

$$\max_{i\notin I}(a_i^T x)^2 < \rho < \min_{i\in I}(a_i^T x)^2, \tag{9}$$

the matrix $X = xx^T$ is optimal for problem (6) if and only if there are matrices $Y_i \in \mathbf{S}^n$ satisfying

$$\lambda_{\max}\left(\sum_{i\in I}\frac{B_i xx^T B_i}{x^T B_i x} + \sum_{i\in I^c} Y_i\right) \leq \sum_{i\in I}((a_i^T x)^2 - \rho),$$

with $Y_i \succeq B_i - \frac{B_i xx^T B_i}{x^T B_i x}$, $Y_i \succeq 0$, where $B_i = a_i a_i^T - \rho\mathbf{I}$, $i = 1,\ldots,n$.

Proof We first prove the necessary condition by computing a first order expansion of the functions $F_i : X \mapsto \mathbf{Tr}(X^{1/2} B_i X^{1/2})_+$ around $X = xx^T$. The expansion is based on the results in Appendix A which show how to compute derivatives of eigenvalues and projections on eigensubspaces. More precisely, Lemma 10 states that if $x^T Bx > 0$, then, for any $Y \succeq 0$:

$$F_i((1-t)xx^T + tY) = F_i(xx^T) + \frac{t}{x^T B_i x}\mathbf{Tr}\, B_i xx^T B_i (Y - xx^T) + O(t^{3/2}),$$

while if $x^T Bx < 0$, then, for any $Y \succeq 0$,:

$$F_i((1-t)xx^T + tY) = t_+ \mathbf{Tr}\left(Y^{1/2}\left(B_i - \frac{B_i xx^T B_i}{x^T B_i x}\right)Y^{1/2}\right)_+ + O(t^{3/2}).$$

Thus if $X = xx^T$ is a global maximum of $\sum_i F_i(X)$, then this first order expansion must reflect the fact that it is also local maximum, that is, for all $Y \in \mathbf{S}^n$ such that $Y \succeq 0$ and $\mathbf{Tr}\, Y = 1$, we must have:

$$\lim_{t\to 0_+}\frac{1}{t}\sum_{i=1}^n [F_i((1-t)xx^T + tY) - F_i(xx^T)] \leq 0,$$

which is equivalent to:

$$-\sum_{i\in I} x^T B_i x + \mathbf{Tr}\, Y\left(\sum_{i\in I}\frac{B_i xx^T B_i}{x^T B_i x}\right) + \sum_{i\in I^c}\mathbf{Tr}\left(Y^{1/2}\left(B_i - \frac{B_i xx^T B_i}{x^T B_i x}\right)Y^{1/2}\right)_+ \leq 0.$$

Thus if $X = xx^T$ is optimal, with $\sigma = \sum_{i\in I} x^T B_i x$, we get:

$$\max_{Y\succeq 0, \mathbf{Tr}Y=1} \mathbf{Tr}\, Y\left(\sum_{i\in I}\frac{B_i xx^T B_i}{x^T B_i x} - \sigma \mathbf{I}\right) + \sum_{i\in I^c} \mathbf{Tr}\left(Y^{1/2}\left(B_i - B_i x(x^T B_i x)^\dagger x^T B_i\right)Y^{1/2}\right)_+ \leq 0$$

which is also in dual form (using the same techniques as in the proof of Proposition 1):

$$\min_{\{Y_i \succeq B_i - \frac{B_i xx^T B_i}{x^T B_i x}, Y_i \succeq 0\}} \lambda_{\max}\left(\sum_{i\in I}\frac{B_i xx^T B_i}{x^T B_i x} + \sum_{i\in I^c} Y_i\right) \leq \sigma,$$

which leads to the necessary condition. In order to prove sufficiency, the only non trivial condition to check in Lemma 3 is that $x^T Y_i x = 0$ for $i \in I^c$, which is a consequence of the inequality:

$$x^T\left(\sum_{i\in I}\frac{B_i xx^T B_i}{x^T B_i x} + \sum_{i\in I^c} Y_i\right)x \leq \lambda_{\max}\left(\sum_{i\in I}\frac{B_i xx^T B_i}{x^T B_i x} + \sum_{i\in I^c} Y_i\right) \leq x^T\left(\sum_{i\in I}\frac{B_i xx^T B_i}{x^T B_i x}\right)x.$$

This concludes the proof. ■

The original optimality conditions in (3) are highly degenerate in Y_i and this result refines these optimality conditions by invoking the local structure. The local optimality analysis in proposition 4 gives more specific constraints on the dual variables Y_i. For $i \in I$, Y_i must be equal to $B_i xx^T B_i / x^T B_i x$, while if $i \in I^c$, we must have $Y_i \succeq B_i - B_i xx^T B_i / x^T B_i x$, which is a stricter condition than $Y_i \succeq B_i$ (because $x^T B_i x < 0$).

5.3 Efficient Optimality Conditions

The condition presented in Proposition 4 still requires solving a large semidefinite program. In practice, good candidates for Y_i, $i \in I^c$ can be found by solving for minimum trace matrices satisfying the feasibility conditions of proposition 4. As we will see below, this can be formulated as a semidefinite program which can be solved explicitly.

Lemma 5 *Let $A \in \mathbf{R}^{n\times n}$, $\rho \geq 0$, $x \in \mathbf{R}^n$ and $B_i = a_i a_i^T - \rho\mathbf{I}$ with $a_1, \ldots, a_n \in \mathbf{R}^n$ the columns of A. If $(a_i^T x)^2 < \rho$ and $\|x\| = 1$, an optimal solution of the semidefinite program:*

$$\begin{array}{ll} \textit{minimize} & \mathbf{Tr}\, Y_i \\ \textit{subject to} & Y_i \succeq B_i - \frac{B_i xx^T B_i}{x^T B_i x},\ x^T Y_i x = 0,\ Y_i \succeq 0, \end{array}$$

is given by:

$$Y_i = \max\left\{0, \rho\frac{(a_i^T a_i - \rho)}{(\rho - (a_i^T x)^2)}\right\}\frac{(\mathbf{I} - xx^T)a_i a_i^T(\mathbf{I} - xx^T)}{\|(\mathbf{I} - xx^T)a_i\|^2}. \tag{10}$$

Proof Let us write $M_i = B_i - \frac{B_i xx^T B_i}{x^T B_i x}$, we first compute:

$$\begin{aligned} a_i^T M_i a_i &= (a_i^T a_i - \rho)a_i^T a_i - \frac{(a_i^T a_i a_i^T x - \rho a_i^T x)^2}{(a_i^T x)^2 - \rho} \\ &= \frac{(a_i^T a_i - \rho)}{\rho - (a_i^T x)^2}\rho(a_i^T a_i - (a_i^T x)^2). \end{aligned}$$

When $a_i^T a_i \leq \rho$, the matrix M_i is negative semidefinite, because $\|x\| = 1$ means $a_i^T M a_i \leq 0$ and $x^T M x = a_i^T M x = 0$. The solution of the minimum trace problem is then simply $Y_i = 0$. We now assume that $a_i^T a_i > \rho$ and first check feasibility of the candidate solution Y_i in (10). By construction, we have $Y_i \succeq 0$ and $Y_i x = 0$, and a short calculation shows that:

$$\begin{aligned} a_i^T Y_i a_i &= \rho \frac{(a_i^T a_i - \rho)}{(\rho - (a_i^T x)^2)} (a_i^T a_i - (a_i^T x)^2) \\ &= a_i^T M_i a_i. \end{aligned}$$

We only need to check that $Y_i \succeq M_i$ on the subspace spanned by a_i and x, for which there is equality. This means that Y_i in (10) is feasible and we now check its optimality. The dual of the original semidefinite program can be written:

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}\, P_i M_i \\ \text{subject to} & \mathbf{I} - P_i + \nu x x^T \succeq 0 \\ & P_i \succeq 0, \end{array}$$

and the KKT optimality conditions for this problem are written:

$$\left\{ \begin{array}{l} Y_i(\mathbf{I} - P_i + \nu x x^T) = 0,\ P_i(Y_i - M_i) = 0, \\ \mathbf{I} - P_i + \nu x x^T \succeq 0, \\ P_i \succeq 0,\ Y_i \succeq 0,\ Y_i \succeq M_i,\ Y_i x x^T = 0, \quad i \in I^c. \end{array} \right.$$

Setting $P_i = Y_i \mathbf{Tr}\, Y_i / \mathbf{Tr}\, Y_i^2$ and ν sufficiently large makes these variables dual feasible. Because all contributions of x are zero, $\mathbf{Tr}\, Y_i(Y_i - M_i)$ is proportional to $\mathbf{Tr}\, a_i a_i^T (Y_i - M_i)$ which is equal to zero and Y_i in (10) satisifies the KKT optimality conditions. ■

We summarize the results of this section in the theorem below, which provides sufficient optimality conditions on a sparsity pattern I.

Theorem 6 *Let $A \in \mathbf{R}^{n \times n}$, $\rho \geq 0$, $\Sigma = A^T A$ with $a_1, \ldots, a_n \in \mathbf{R}^n$ the columns of A. Given a sparsity pattern I, setting x to be the largest eigenvector of $\sum_{i \in I} a_i a_i^T$, if there is a $\rho^* \geq 0$ such that the following conditions hold:*

$$\max_{i \in I^c} (a_i^T x)^2 < \rho^* < \min_{i \in I} (a_i^T x)^2 \quad \textit{and} \quad \lambda_{\max}\left(\sum_{i=1}^n Y_i \right) \leq \sum_{i \in I} ((a_i^T x)^2 - \rho^*),$$

with the dual variables Y_i for $i \in I^c$ defined as in (10) and:

$$Y_i = \frac{B_i x x^T B_i}{x^T B_i x}, \quad \textit{when } i \in I,$$

then the sparsity pattern I is globally optimal for the sparse PCA problem (2) with $\rho = \rho^*$ and we can form an optimal solution z by solving the maximum eigenvalue problem:

$$z = \operatorname*{argmax}_{\{z_{I^c} = 0,\ \|z\| = 1\}} z^T \Sigma z.$$

Proof Following proposition 4 and lemma 5, the matrices Y_i are dual optimal solutions corresponding to the primal optimal solution $X = xx^T$ in (5). Because the primal solution has rank one, the semidefinite relaxation (6) is tight so the pattern I is optimal for (2) and Section 2 shows that z is a globally optimal solution to (2) with $\rho = \rho^*$. ■

5.4 Gap Minimization: Finding the Optimal ρ

All we need now is an efficient algorithm to find ρ^* in theorem 6. As we will show below, when the dual variables Y_i^c are defined as in (10), the duality gap in (2) is a convex function of ρ hence, given a sparsity pattern I, we can efficiently search for the best possible ρ (which must belong to an *interval*) by performing a few binary search iterations.

Lemma 7 *Let $A \in \mathbf{R}^{n\times n}$, $\rho \geq 0$, $\Sigma = A^TA$ with $a_1,\ldots,a_n \in \mathbf{R}^n$ the columns of A. Given a sparsity pattern I, setting x to be the largest eigenvector of $\sum_{i\in I} a_i a_i^T$, with the dual variables Y_i for $i \in I^c$ defined as in (10) and:*

$$Y_i = \frac{B_i x x^T B_i}{x^T B_i x}, \quad \textit{when } i \in I.$$

The duality gap in (2) which is given by:

$$\mathrm{gap}(\rho) \equiv \lambda_{\max}\left(\sum_{i=1}^n Y_i\right) - \sum_{i\in I}((a_i^Tx)^2 - \rho),$$

is a convex function of ρ when

$$\max_{i\notin I}(a_i^Tx)^2 < \rho < \min_{i\in I}(a_i^Tx)^2.$$

Proof For $i \in I$ and $u \in \mathbf{R}^n$, we have

$$u^TY_iu = \frac{(u^Ta_ia_i^Tx - \rho u^Tx)^2}{(a_i^Tx)^2 - \rho},$$

which is a convex function of ρ (Boyd and Vandenberghe, 2004, p.73). For $i \in I^c$, we can write:

$$\frac{\rho(a_i^Ta_i - \rho)}{\rho - (a_i^Tx)^2} = -\rho + (a_i^Ta_i - (a_i^Tx)^2)\left(1 + \frac{(a_i^Tx)^2}{\rho - (a_i^Tx)^2}\right),$$

hence $\max\{0, \rho(a_i^Ta_i - \rho)/(\rho - (a_i^Tx)^2)\}$ is also a convex function of ρ. This means that:

$$u^TY_iu = \max\left\{0, \rho\frac{(a_i^Ta_i - \rho)}{(\rho - (a_i^Tx)^2)}\right\}\frac{(u^Ta_i - (x^Tu)(x^Ta_i))^2}{\|(\mathbf{I} - xx^T)a_i\|^2}$$

is convex in ρ when $i \in I^c$. We conclude that $\sum_{i=1}^n u^TY_iu$ is convex, hence:

$$\mathrm{gap}(\rho) = \max_{\|u\|=1} \sum_{i=1}^n u^TY_iu - \sum_{i\in I}((a_i^Tx)^2 - \rho)$$

is also convex in ρ as a pointwise maximum of convex functions of ρ. ■

This result shows that the set of ρ for which the pattern I is optimal must be an interval. It also suggests an efficient procedure for testing the optimality of a given pattern I. We first compute x as a leading eigenvector $\sum_{i\in I} a_i a_i^T$. We then compute an interval in ρ for which x satisfies the basic consistency condition:

$$\max_{i\notin I}(a_i^T x)^2 \equiv \rho_{\min} \leq \rho \leq \rho_{\max} \equiv \min_{i\in I}(a_i^T x)^2.$$

Note that this interval could be empty, in which case I cannot be optimal. We then minimize $\mathrm{gap}(\rho)$ over the interval $[\rho_{\min},\rho_{\max}]$. If the minimum is zero for some $\rho=\rho^*$, then the pattern I is optimal for the sparse PCA problem in (2) with $\rho=\rho^*$.

Minimizing the convex function $\mathrm{gap}(\rho)$ can be done very efficiently using binary search. The initial cost of forming the matrix $\sum_{i=1}^n Y_i$, which is a simple outer matrix product, is $O(n^3)$. At each iteration of the binary search, a subgradient of $\mathrm{gap}(\rho)$ can then be computed by solving a maximum eigenvalue problem, at a cost of $O(n^2)$. This means that the complexity of finding the optimal ρ over a given interval $[\rho_{\min},\rho_{\max}]$ is $O(n^2\log_2((\rho_{\max}-\rho_{\min})/\varepsilon))$, where ε is the target precision. Overall then, the total cost of testing the optimality of a pattern I is $O(n^3+n^2\log_2((\rho_{\max}-\rho_{\min})/\varepsilon))$.

Note that an additional benefit of deriving explicit dual feasible points Y_i is that plugging these solutions into the objective of problem (7):

$$\begin{array}{ll}\text{min.} & \lambda_{\max}\left(\sum_{i=1}^n Y_i\right)\\ \text{s.t.} & Y_i \succeq B_i,\ Y_i \succeq 0, \quad i=1,\ldots,n.\end{array}$$

produces an *upper bound* on the optimum value of the original sparse PCA problem (2) even when the pattern I is not optimal (all we need is a ρ satisfying the consistency condition).

5.5 Solution Improvements and Randomization

When these conditions are not satisfied, the relaxation (6) has an optimal solution with rank strictly larger than one, hence is not tight. At such a point, we can use a different relaxation such as DSPCA by d'Aspremont et al. (2007b) to try to get a better solution. We can also apply randomization techniques to improve the quality of the solution of problem (6) (Ben-Tal and Nemirovski, 2002).

6. Applications

In this section, we discuss some applications of sparse PCA to subset selection and compressed sensing.

6.1 Subset Selection

We consider p data points in $\mathbf{R}^n$, in a data matrix $X\in\mathbf{R}^{p\times n}$. We assume that we are given real numbers $y\in\mathbf{R}^p$ to predict from X using linear regression, estimated by least squares. We are thus looking for $w\in\mathbf{R}^n$ such that $\|y-Xw\|^2$ is minimum. In the subset selection problem, we are looking for sparse coefficients w, that is, a vector w with many zeros. We thus consider the problem:

$$s(k) = \min_{w\in\mathbf{R}^n,\ \mathbf{Card}\, w\leq k} \|y-Xw\|^2.$$

Using the sparsity pattern $u \in \{0,1\}^n$, and optimizing with respect to w, we have

$$s(\rho) = \min_{u \in \{0,1\}^n,\ \mathbf{1}^T u \leq k} \|y\|^2 - y^T X(u)(X(u)^T X(u))^{-1} X(u)^T y,$$

where $X(u) = X\,\mathbf{diag}(u)$. We can rewrite $y^T X(u)(X(u)^T X(u))^{-1} X(u)^T y$ as the largest generalized eigenvalue of the pair $(X(u)^T yy^T X(u), X(u)^T X(u))$, that is, as

$$y^T X(u)(X(u)^T X(u))^{-1} X(u)^T y = \max_{w \in \mathbf{R}^n} \frac{w^T X(u)^T yy^T X(u) w}{w^T X(u)^T X(u) w}.$$

We thus have:

$$s(k) = \|y\|^2 - \max_{u \in \{0,1\}^n, \mathbf{1}^T u \leq k} \max_{w \in \mathbf{R}^n} \frac{w^T\,\mathbf{diag}(u) X^T yy^T X\,\mathbf{diag}(u) w}{w^T\,\mathbf{diag}(u) X^T X\,\mathbf{diag}(u)) w}.$$

Given a pattern $u \in \{0,1\}^n$, let

$$s_0 = y^T X(u)(X(u)^T X(u))^{-1} X(u)^T y$$

be the largest generalized eigenvalue corresponding to the pattern u. The pattern is optimal if and only if the largest generalized eigenvalue of the pair $\{X(v)^T yy^T X(v), X(v)^T X(v)\}$ is less than s_0 for any $v \in \{0,1\}^n$ such that $v^T \mathbf{1} = u^T \mathbf{1}$. This is equivalent to the optimality of u for the sparse PCA problem with matrix $X^T yy^T X - s_0 X^T X$, which can be checked using the sparse PCA optimality conditions derived in the previous sections.

Note that unlike in the sparse PCA case, this convex relaxation does not immediately give a simple bound on the optimal value of the subset selection problem. However, we get a bound of the following form: when $v \in \{0,1\}^n$ and $w \in \mathbf{R}^n$ is such that $\mathbf{1}^T v = k$ with:

$$w^T \left(X(v)^T yy^T X(v) - s_0 X(v)^T X(v) \right) w \leq B,$$

where $B \geq 0$ (because s_0 is defined from u), we have:

$$\begin{aligned} \|y\|^2 - s_0 \geq s(k) &\geq \|y\|^2 - s_0 - B \left(\min_{v \in \{0,1\}^n, \mathbf{1}^T v = k} \lambda_{\min}(X(v)^T X(v)) \right)^{-1} \\ &\geq \|y\|^2 - s_0 - B \left(\lambda_{\min}(X^T X) \right)^{-1}. \end{aligned}$$

This bound gives a sufficient condition for optimality in subset selection, for any problem instance and any given subset. This is to be contrasted with the sufficient conditions derived for particular algorithms, such as the LASSO (Yuan and Lin, 2007; Zhao and Yu, 2006) or backward greedy selection (Couvreur and Bresler, 2000). Note that some of these optimality conditions are often based on sparse eigenvalue problems (see Meinshausen and Yu, 2006, §2), hence our convex relaxations helps both in checking sufficient conditions for optimality (before the algorithm is run) and in testing a posteriori the optimality of a particular solution.

6.2 Sparse Recovery

Following Candès and Tao (2005) (see also Donoho and Tanner, 2005), we seek to recover a signal $f \in \mathbf{R}^n$ from corrupted measurements $y = Af + e$, where $A \in \mathbf{R}^{m\times n}$ is a coding matrix and $e \in \mathbf{R}^m$ is an unknown vector of errors with low cardinality. This can be reformulated as the problem of finding the sparsest solution to an underdetermined linear system:

$$\begin{array}{ll} \text{minimize} & \|x\|_0 \\ \text{subject to} & Fx = Fy \end{array} \tag{11}$$

where $\|x\|_0 = \mathbf{Card}(x)$ and $F \in \mathbf{R}^{p\times m}$ is a matrix such that $FA = 0$. A classic trick to get good approximate solutions to problem (11) is to substitute the (convex) ℓ_1 norm to the (combinatorial) ℓ_0 objective above, and solve instead:

$$\begin{array}{ll} \text{minimize} & \|x\|_1 \\ \text{subject to} & Fx = Fy, \end{array}$$

which is equivalent to a linear program in $x \in \mathbf{R}^m$. Following Candès and Tao (2005), given a matrix $F \in \mathbf{R}^{p\times m}$ and an integer S such that $0 < S \leq m$, we define its *restricted isometry* constant δ_S as the smallest number such that for any subset $I \subset [1,m]$ of cardinality at most S we have:

$$(1-\delta_S)\|c\|^2 \leq \|F_I c\|^2 \leq (1+\delta_S)\|c\|^2, \tag{12}$$

for all $c \in \mathbf{R}^{|I|}$, where F_I is the submatrix of F formed by keeping only the columns of F in the set I. The following result then holds.

Proposition 8 ***Candès and Tao (2005)***. *Suppose that the restricted isometry constants of a matrix $F \in \mathbf{R}^{p\times m}$ satisfy*

$$\delta_S + \delta_{2S} + \delta_{3S} < 1 \tag{13}$$

for some integer S such that $0 < S \leq m$, then if x is an optimal solution of the convex program:

$$\begin{array}{ll} \textit{minimize} & \|x\|_1 \\ \textit{subject to} & Fx = Fy \end{array}$$

such that $\mathbf{Card}\, x \leq S$ *then x is also an optimal solution of the combinatorial problem:*

$$\begin{array}{ll} \textit{minimize} & \|x\|_0 \\ \textit{subject to} & Fx = Fy. \end{array}$$

In other words, if condition (13) holds for some matrix F such that $FA = 0$, then perfect recovery of the signal f given $y = Af + e$ provided the error vector satisfies $\mathbf{Card}(e) \leq S$. Our key observation here is that the restricted isometry constant δ_S in condition (13) can be computed by solving the following sparse maximum eigenvalue problem:

$$\begin{array}{lll} (1+\delta_S) \leq & \max. & x^T(F^TF)x \\ & \text{s. t.} & \mathbf{Card}(x) \leq S \\ & & \|x\| = 1, \end{array}$$

in the variable $x \in \mathbf{R}^m$ and another sparse maximum eigenvalue problem on $\alpha\mathbf{I} - FF^T$ with α sufficiently large, with δ_S computed from the tightest one. In fact, (12) means that:

$$\begin{aligned}(1+\delta_S) &\leq \max_{\{I\subset[1,m]:\ |I|\leq S\}} \ \max_{\|c\|=1} c^T F_I^T F_I c \\ &= \max_{\{u\in\{0,1\}^n:\ \mathbf{1}^T u\leq S\}} \ \max_{\|x\|=1} x^T \mathbf{diag}(u) F^T F \mathbf{diag}(u) x \\ &= \max_{\{\|x\|=1,\ \mathbf{Card}(x)\leq S\}} x^T F^T F x,\end{aligned}$$

hence we can compute an upper bound on δ_S by duality, with:

$$(1+\delta_S) \leq \inf_{\rho\geq 0} \phi(\rho) + \rho S$$

where $\phi(\rho)$ is defined in (2). This means that while Candès and Tao (2005) obtained an asymptotic proof that some random matrices satisfied the restricted isometry condition (13) with overwhelming probability (i.e., exponentially small probability of failure), whenever they are satisfied, the *tractable* optimality conditions and upper bounds we obtain in Section 5 for sparse PCA problems allow us to prove, *deterministically*, that a finite dimensional matrix satisfies the restricted isometry condition in (13). Note that Candès and Tao (2005) provide a slightly weaker condition than (13) based on restricted orthogonality conditions and extending the results on sparse PCA to these conditions would increase the maximum S for which perfect recovery holds. In practice however, we will see in Section 7.3 that the relaxations in (7) and d'Aspremont et al. (2007b) do provide very tight upper bounds on sparse eigenvalues of random matrices but solving these semidefinite programs for very large scale instances remains a significant challenge.

7. Numerical Results

In this section, we first compare the various methods detailed here on artificial examples, then test their performance on a biological data set. PathSPCA, a MATLAB code reproducing these results may be downloaded from the authors' web pages.

7.1 Artificial Data

We generate a matrix U of size 150 with uniformly distributed coefficients in $[0,1]$. We let $v \in \mathbf{R}^{150}$ be a sparse vector with:

$$v_i = \begin{cases} 1 & \text{if } i \leq 50 \\ 1/(i-50) & \text{if } 50 < i \leq 100 \\ 0 & \text{otherwise.} \end{cases}$$

We form a test matrix $\Sigma = U^T U + \sigma v v^T$, where σ is the signal-to-noise ratio. We first compare the relative performance of the algorithms in Section 3 at identifying the correct sparsity pattern in v given the matrix Σ. The resulting ROC curves are plotted in Figure 1 for $\sigma = 2$. On this example, the computing time for the approximate greedy algorithm in Section 3.3 was 3 seconds versus 37 seconds for the full greedy solution in Section 3.2. Both algorithms produce almost identical answers. We can also see that both sorting and thresholding ROC curves are dominated by the greedy algorithms.

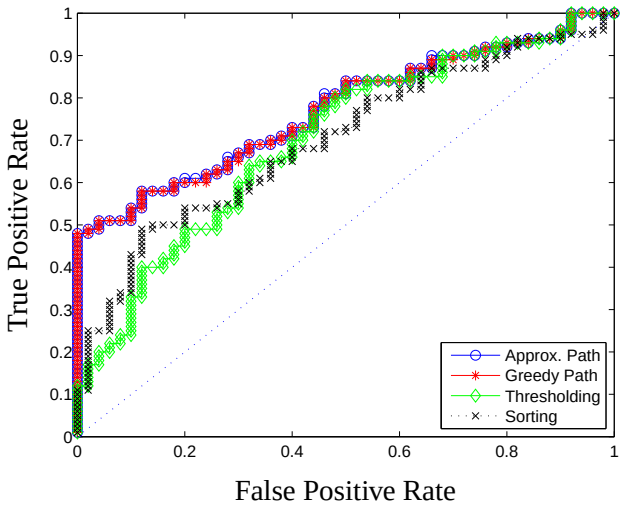


Figure 1: ROC curves for sorting, thresholding, fully greedy solutions (Section 3.2) and approximate greedy solutions (Section 3.3) for $\sigma = 2$.

We then plot the variance versus cardinality tradeoff curves for various values of the signal-to-noise ratio. In Figure 2, We notice that the magnitude of the error (duality gap) decreases with the signal-to-noise ratio. Also, because of the structure of our problem, there is a kink in the variance at the (exact) cardinality 50 in each of these curves. Note that for each of these examples, the error (duality gap) is minimal precisely at the kink.

Next, we use the DSPCA algorithm of d'Aspremont et al. (2007b) to find better solutions where the greedy codes have failed to obtain globally optimal solutions. In d'Aspremont et al. (2007b), it was shown that an upper bound on (2) can be computed as:

$$\phi(\rho) \leq \min_{|U_{ij}| \leq \rho} \lambda_{\max}(\Sigma + U).$$

which is a convex problem in the matrix $U \in \mathbf{S}_n$. Note however that the cost of solving this relaxation for a *single* ρ is $O(n^4\sqrt{\log n})$ versus $O(n^3)$ for a full path of approximate solutions. Also, the results in d'Aspremont et al. (2007b) do not provide any hint on the value of ρ, but we can use the breakpoints coming from suboptimal points in the greedy search algorithms in Section 3.3 and the consistency intervals in Eq. (9). In Figure 2 we plot the variance versus cardinality tradeoff curve for $\sigma = 10$. We plot greedy variances (solid line), dual upper bounds from Section 5.3 (dotted line) and upper bounds computed using DSPCA (dashed line).

7.2 Subset Selection

We now present simulation experiments on synthetic data sets for the subset selection problem. We consider data sets generated from a sparse linear regression problem and study optimality for the subset selection problem, given the exact cardinality of the generating vector. In this setting, it is known that regularization by the ℓ_1-norm, a procedure also known as the Lasso (Tibshirani, 1996), will asymptotically lead to the correct solution if and only if a certain consistency condition is satisfied (Yuan and Lin, 2007; Zhao and Yu, 2006). Our results provide here a tractable test

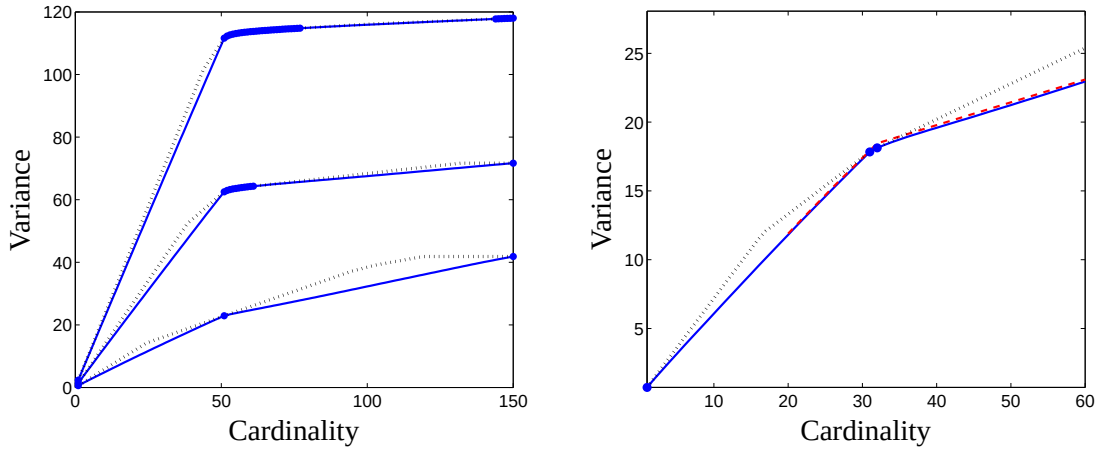


Figure 2: *Left:* variance versus cardinality tradeoff curves for $\sigma = 10$ (bottom), $\sigma = 50$ and $\sigma = 100$ (top). We plot the variance (solid line) and the dual upper bounds from Section 5.3 (dotted line) for each target cardinality. *Right:* variance versus cardinality tradeoff curve for $\sigma = 10$. We plot greedy variances (solid line), dual upper bounds from Section 5.3 (dotted line) and upper bounds computed using DSPCA (dashed line). Optimal points (for which the relative duality gap is less than 10^{-4}) are in bold.

the optimality of solutions obtained from various algorithms such as the Lasso, forward greedy or backward greedy algorithms.

In Figure 3, we consider two pairs of randomly generated examples in dimension 16, one for which the lasso is provably consistent, one for which it isn't. We perform 50 simulations with 1000 samples and varying noise and compute the average frequency of optimal subset selection for Lasso and greedy backward algorithm together with the frequency of provable optimality (i.e., where our method did ensure a posteriori that the point was optimal). We can see that the backward greedy algorithm exhibits good performance (even in the Lasso-inconsistent case) and that our sufficient optimality condition is satisfied as long as there is not too much noise. In Figure 4, we plot the average mean squared error versus cardinality, over 100 replications, using forward (dotted line) and backward (circles) selection, the Lasso (large dots) and exhaustive search (solid line). The plot on the left shows the results when the Lasso consistency condition is satisfied, while the plot on the right shows the mean squared errors when the consistency condition is not satisfied. The two sets of figures do show that the LASSO is consistent only when the consistency condition is satisfied, while the backward greedy algorithm finds the correct pattern if the noise is small enough (Couvreur and Bresler, 2000) even in the LASSO inconsistent case.

7.3 Sparse Recovery

Following the results of Section 6.2, we compute the upper and lower bounds on sparse eigenvalues produced using various algorithms. We study the following problem:

$$\begin{array}{ll} \text{maximize} & x^T \Sigma x \\ \text{subject to} & \mathbf{Card}(x) \leq S \\ & \|x\| = 1, \end{array}$$

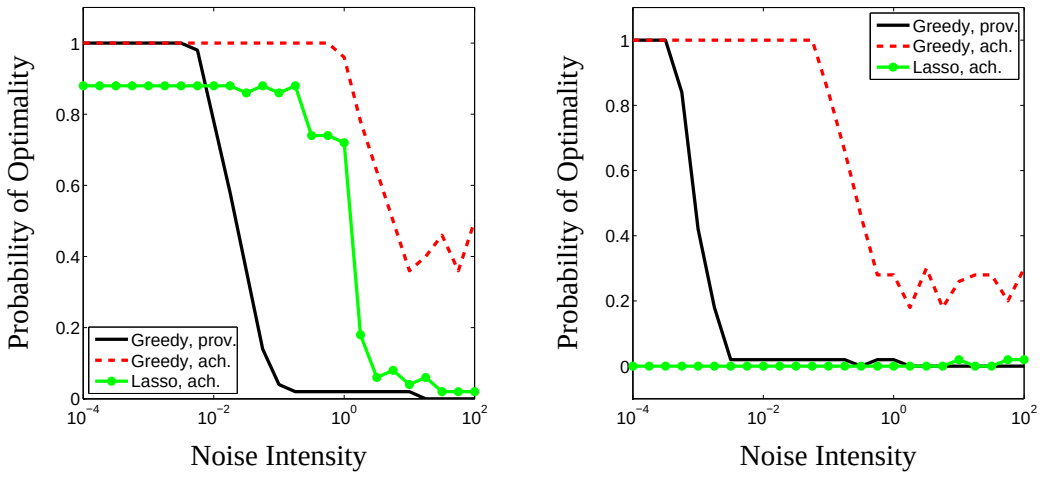


Figure 3: Backward greedy algorithm and Lasso. We plot the probability of achieved (dotted line) and provable (solid line) optimality versus noise for greedy selection against Lasso (large dots), for the subset selection problem on a noisy sparse vector. *Left:* Lasso consistency condition satisfied. *Right:* consistency condition not satisfied.

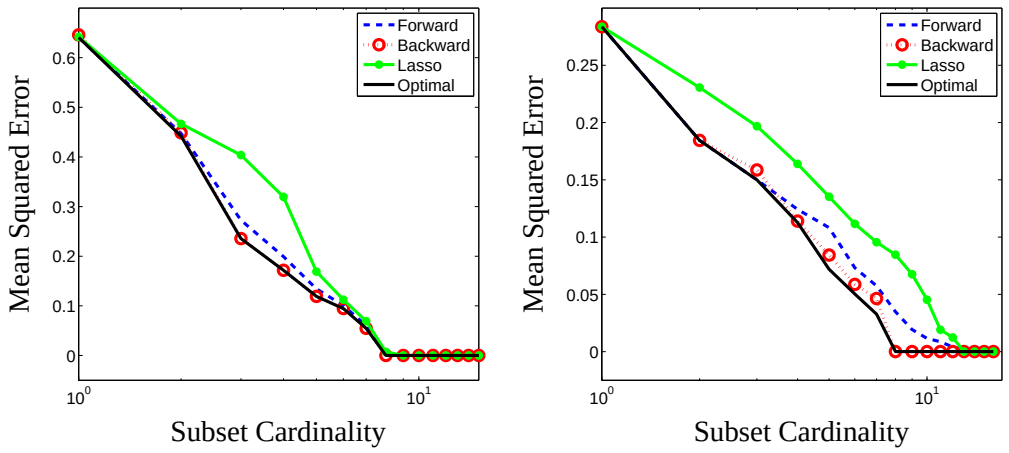


Figure 4: Greedy algorithm and Lasso. We plot the average mean squared error versus cardinality, over 100 replications, using forward (dotted line) and backward (circles) selection, the Lasso (large dots) and exhaustive search (solid line). *Left:* Lasso consistency condition satisfied. *Right:* consistency condition not satisfied.

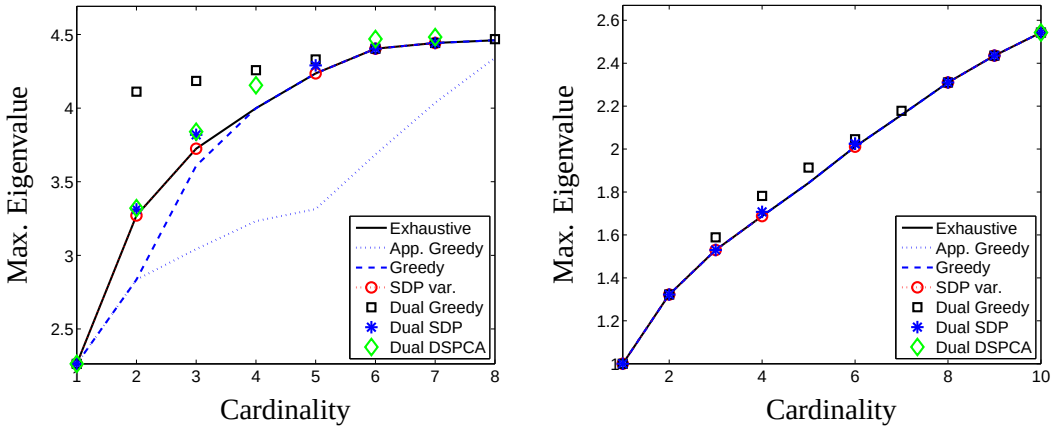


Figure 5: Upper and lower bound on sparse maximum eigenvalues. We plot the maximum sparse eigenvalue versus cardinality, obtained using exhaustive search (solid line), the approximate greedy (dotted line) and fully greedy (dashed line) algorithms. We also plot the upper bounds obtained by minimizing the gap of a rank one solution (squares), by solving the semidefinite relaxation explicitly (stars) and by solving the DSPCA dual (diamonds). *Left:* On a matrix F^TF with F Gaussian. *Right:* On a sparse rank one plus noise matrix.

where we pick F to be normally distributed and small enough so that computing sparse eigenvalues by exhaustive search is numerically feasible. We plot the maximum sparse eigenvalue versus cardinality, obtained using exhaustive search (solid line), the approximate greedy (dotted line) and fully greedy (dashed line) algorithms. We also plot the upper bounds obtained by minimizing the gap of a rank one solution (squares), by solving the semidefinite relaxation explicitly (stars) and by solving the DSPCA dual (diamonds). On the left, we use a matrix $\Sigma = F^TF$ with F Gaussian. On the right, $\Sigma = uu^T/\|u\|^2 + 2V$, where $u_i = 1/i,\ i = 1,\ldots,n$ and V is matrix with coefficients uniformly distributed in $[0,1]$. Almost all algorithms are provably optimal in the noisy rank one case (as well as in many of the biological examples that follow), while Gaussian random matrices are harder. Note however, that the duality gap between the semidefinite relaxations and the optimal solution is very small in both cases, while our bounds based on greedy solutions are not as good. This means that solving the relaxations in (7) and d'Aspremont et al. (2007b) could provide very tight upper bounds on sparse eigenvalues of random matrices. However, solving these semidefinite programs for very large values of n remains a significant challenge.

7.4 Biological Data

We run the algorithm of Section 3.3 on two gene expression data sets, one on Colon cancer from Alon et al. (1999), the other on Lymphoma from Alizadeh et al. (2000). We plot the variance versus cardinality tradeoff curve in Figure 6, together with the dual upper bounds from Section 5.3. In both cases, we consider the 500 genes with largest variance. Note that for many cardinalities, we have optimal or very close to optimal solutions. In Table 1, we also compare the 20 most important genes selected by the second sparse PCA factor on the colon cancer data set, with the top 10 genes selected by the RankGene software by Su et al. (2003). We observe that 6 genes (out of an original 4027 genes) were both in the top 20 sparse PCA genes and in the top 10 Rankgene genes.

Rank	Rankgene	GAN	Description
3	8.6	J02854	Myosin regul.
6	18.9	T92451	Tropomyosin
7	31.5	T60155	Actin
8	25.1	H43887	Complement fact. D prec.
10	2.1	M63391	Human desmin
12	32.3	T47377	S-100P Prot.

Table 1: 6 genes (out of 4027) that were both in the top 20 sparse PCA genes and in the top 10 Rankgene genes.

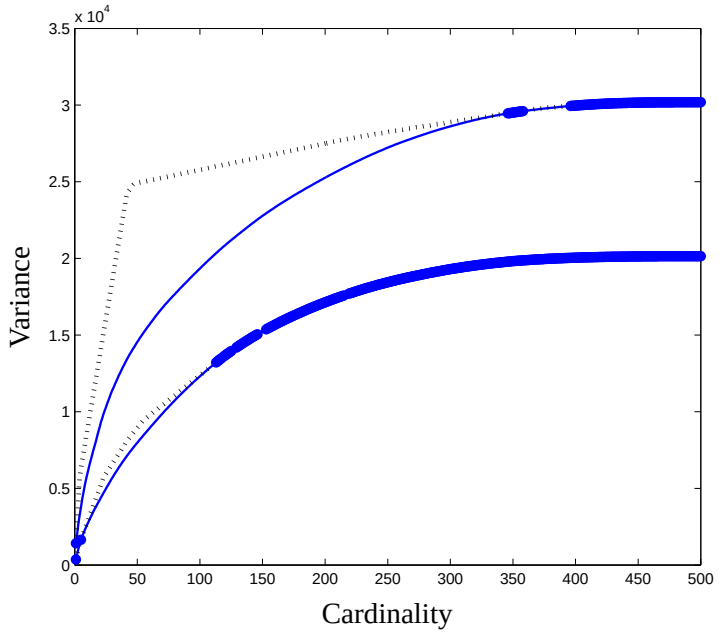


Figure 6: Variance (solid lines) versus cardinality tradeoff curve for two gene expression data sets, lymphoma (top) and colon cancer (bottom), together with dual upper bounds from Section 5.3 (dotted lines). Optimal points (for which the relative duality gap is less than 10^{-4}) are in bold.

8. Conclusion

We have presented a new convex relaxation of sparse principal component analysis, and derived tractable sufficient conditions for optimality. These conditions go together with efficient greedy algorithms that provide candidate solutions, many of which turn out to be optimal in practice. The resulting upper bounds also have direct applications to problems such as sparse recovery, subset selection or LASSO variable selection. Note that we extensively use this convex relaxation to test optimality and provide bounds on sparse extremal eigenvalues, but we almost never attempt to solve it numerically (except in some of the numerical experiments), which would provide optimal

bounds. Having n matrix variables of dimension n, the problem is of course extremely large and finding numerical algorithms to directly optimize these relaxation bounds would be an important extension of this work.

Acknowledgments

The first author acknowledges support from NSF grant DMS-0625352, ONR grant number N00014-07-1-0150 and a gift from Google, Inc. We would like to thank Katya Scheinberg and the organizers of the Banff workshop on optimization and machine learning, where most of this paper was written.

Appendix A. Expansion of Eigenvalues

In this appendix, we consider various results on expansions of eigenvalues we use in order to derive sufficient conditions. The following proposition derives a second order expansion of the set of eigenvectors corresponding to a single eigenvalue.

Proposition 9 *Let $N \in \mathbf{S}^n$. Let λ_0 be an eigenvalue of N, with multiplicity r and eigenvectors $U \in \mathbf{R}^{n\times r}$ (such that $U^TU = \mathbf{I}$). Let Δ be a matrix in $\mathbf{S}^n$. If $\|\Delta\|_F$ is small enough, the matrix $N+\Delta$ has exactly r (possibly equal) eigenvalues around λ_0 and if we denote by $(N+\Delta)_{\lambda_0}$ the projection of the matrix $N+\Delta$ onto that eigensubspace, we have:*

$$\begin{aligned}(N+\Delta)_{\lambda_0} \;=\;& \lambda_0 UU^T + UU^T\Delta UU^T + \lambda_0 UU^T\Delta(\lambda_0\mathbf{I}-N)^\dagger + \lambda_0(\lambda_0\mathbf{I}-N)^\dagger\Delta UU^T\\ &+UU^T\Delta UU^T\Delta(\lambda_0\mathbf{I}-N)^\dagger + (\lambda_0\mathbf{I}-N)^\dagger\Delta UU^T\Delta UU^T + UU^T\Delta(\lambda_0\mathbf{I}-N)^\dagger UU^T\\ &+\lambda_0 UU^T\Delta(\lambda_0\mathbf{I}-N)^\dagger\Delta(\lambda_0\mathbf{I}-N)^\dagger + \lambda_0(\lambda_0\mathbf{I}-N)^\dagger\Delta(\lambda_0\mathbf{I}-N)^\dagger\Delta UU^T\\ &+\lambda_0(\lambda_0\mathbf{I}-M)^\dagger\Delta UU^T\Delta(\lambda_0\mathbf{I}-M)^\dagger + O(\|\Delta\|_F^3)\end{aligned}$$

which implies the following expansion for the sum of the r eigenvalues in the neigborhood of λ_0:

$$\begin{aligned}\mathbf{Tr}(N+\Delta)_{\lambda_0} \;=\;& r\lambda_0 + \mathbf{Tr}\,U^T\Delta U + \mathbf{Tr}\,U^T\Delta(\lambda_0\mathbf{I}-N)^\dagger\Delta U\\ &+\lambda_0\,\mathbf{Tr}(\lambda_0\mathbf{I}-N)^\dagger\Delta UU^T\Delta(\lambda_0\mathbf{I}-N)^\dagger + O(\|\Delta\|_F^3).\end{aligned}$$

Proof We use the Cauchy residue formulation of projections on principal subspaces (Kato, 1966): given a symmetric matrix N, and a simple closed curve $\mathcal{C}$ in the complex plane that does not go through any of the eigenvalues of N, then

$$\Pi_C(N) = \frac{1}{2i\pi}\oint_{\mathcal{C}}\frac{d\lambda}{\lambda\mathbf{I}-N}$$

is equal to the orthogonal projection onto the orthogonal sum of all eigensubspaces of N associated with eigenvalues in the interior of $\mathcal{C}$ (Kato, 1966). This is easily seen by writing down the eigenvalue decomposition $N = \sum_{i=1}^n \lambda_i u_i u_i^T$, and the Cauchy residue formula ($\frac{1}{2i\pi}\oint_{\mathcal{C}}\frac{d\lambda}{\lambda-\lambda_i} = 1$ if λ_i is in the interior $\mathrm{int}(\mathcal{C})$ of $\mathcal{C}$ and 0 otherwise), and:

$$\frac{1}{2i\pi}\oint_{\mathcal{C}}\frac{d\lambda}{\lambda\mathbf{I}-N} = \sum_{i=1}^n u_iu_i^T \times \frac{1}{2i\pi}\oint_{\mathcal{C}}\frac{d\lambda}{\lambda-\lambda_i} = \sum_{i,\ \lambda_i\in\mathrm{int}(\mathcal{C})} u_iu_i^T.$$

See Rudin (1987) for an introduction to complex analysis and Cauchy residue formula. Moreover, we can obtain the restriction of N onto a specific sum of eigensubspaces as:

$$N\Pi_C(N) = \frac{1}{2i\pi}\oint_C \frac{Nd\lambda}{\lambda \mathbf{I} - N} = \frac{1}{2i\pi}\oint_C \frac{\lambda d\lambda}{\lambda \mathbf{I} - N}.$$

From there we can easily compute expansions around a given N by using expansions of $(\lambda \mathbf{I} - N)^{-1}$. The proposition follows by considering a circle around λ_0 that is small enough to exclude other eigenvalues of N, and applying several times the Cauchy residue formula. ■

We can now apply the previous proposition to our particular case:

Lemma 10 *For any $a \in \mathbf{R}^n$, $\rho > 0$ and $B = aa^T - \rho \mathbf{I}$, we consider the function $F : X \mapsto \mathbf{Tr}(X^{1/2} B X^{1/2})_+$ from $\mathbf{S}_+^n$ to $\mathbf{R}$. let $x \in \mathbf{R}^n$ such that $\|x\| = 1$. Let $Y \succeq 0$. If $x^T B x > 0$, then*

$$F((1-t)xx^T + tY) = x^T B x + \frac{t}{x^T B x}\mathbf{Tr}\, B x x^T B (Y - xx^T) + O(t^{3/2}),$$

while if $x^T B x < 0$, then

$$F((1-t)xx^T + tY) = \mathbf{Tr}\left(Y^{1/2}\left(B - \frac{Bxx^TB}{x^TBx}\right)Y^{1/2}\right)_+ + O(t^{3/2}).$$

Proof We consider $X(t) = (1-t)xx^T + tY$. We have $X(t) = U(t)U(t)^T$ with $U(t) = \begin{pmatrix} (1-t)^{1/2}x \\ t^{1/2}Y^{1/2} \end{pmatrix}$, which implies that the non zero eigenvalues of $X(t)^{1/2} B X(t)^{1/2}$ are the same as the non zero eigenvalues of $U(t)^T B U(t)$. We thus have

$$F(X(t)) = \mathbf{Tr}(M(t))_+,$$

with

$$\begin{aligned} M(t) &= \begin{pmatrix} (1-t)x^TBx & t^{1/2}(1-t)^{1/2}x^TBY^{1/2} \\ t^{1/2}(1-t)^{1/2}y^TBx & tY^{1/2}BY^{1/2} \end{pmatrix} \\ &= \begin{pmatrix} x^TBx & 0 \\ 0 & 0 \end{pmatrix} + t^{1/2}\begin{pmatrix} 0 & x^TBY^{1/2} \\ Y^{1/2}Bx & 0 \end{pmatrix} + t\begin{pmatrix} -x^TBx & 0 \\ 0 & Y^{1/2}BY^{1/2} \end{pmatrix} + O(t^{3/2}) \\ &= M(0) + t^{1/2}\Delta_1 + t\Delta_2 + O(t^{3/2}). \end{aligned}$$

The matrix $M(0)$ has a single (and simple) non zero eigenvalue which is equal to $\lambda_0 = x^TBx$ with eigenvector $U = (1,0)^T$. The only other eigenvalue of $M(0)$ is zero, with multiplicity n. Proposition 9 can be applied to the two eigenvalues of $M(0)$: there is one eigenvalue of $M(t)$ around x^TBx, while the n remaining ones are around zero. The eigenvalue close to λ_0 is equal to:

$$\begin{aligned} \mathbf{Tr}(M(t))_{\lambda_0} &= t\,\mathbf{Tr}\,U^\top \Delta_2 U + \lambda_0 + t\,\mathbf{Tr}\,U^T\Delta_1(\lambda_0\mathbf{I} - M(0))^\dagger \Delta_1 U \\ &\quad + \lambda_0\,\mathbf{Tr}(\lambda_0\mathbf{I} - M(0))^\dagger \Delta_1 U U^T \Delta_1 (\lambda_0 \mathbf{I} - M(0))^\dagger + O(t^{3/2}) \\ &= x^TBx + \frac{t}{x^TBx}\mathbf{Tr}\,Bxx^TB(Y - xx^T) + O(t^{3/2}). \end{aligned}$$

For the remaining eigenvalues, we get that the projected matrix on the eigensubspace of $M(t)$ associated with eigenvalues around zero is equal to

$$\begin{aligned}(M(t))_0 &= t(\mathbf{I}-UU^T)\Delta_2(\mathbf{I}-UU^T)+t(\mathbf{I}-UU^T)\Delta_1(-M(0))^\dagger(\mathbf{I}-UU^T)+O(t^{3/2}) \\ &= \begin{pmatrix} 0 & 0 \\ 0 & tY^{1/2}(B-\frac{Bxx^TB}{x^TBx})Y^{1/2} \end{pmatrix},\end{aligned}$$

which leads to a positive part equal to $t_+ \mathbf{Tr}\left[Y^{1/2}(B-\frac{Bxx^TB}{x^TBx})Y^{1/2}\right]_+$. When $x^TBx > 0$, then the matrix is negative definite (because $B = aa^T - \rho\mathbf{I}$), and thus the positive part is zero. By summing the two contributions, we obtain the desired result. ■

References

A. Alizadeh, M. Eisen, R. Davis, C. Ma, I. Lossos, and A. Rosenwald. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000.

A. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Cell Biology*, 96:6745–6750, 1999.

A. Ben-Tal and A. Nemirovski. On tractable approximations of uncertain linear matrix inequalities affected by interval uncertainty. *SIAM Journal on Optimization*, 12(3):811–833, 2002.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

J. Cadima and I. T. Jolliffe. Loadings and correlations in the interpretation of principal components. *Journal of Applied Statistics*, 22:203–214, 1995.

E. J. Candès and T. Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215, 2005.

C. Couvreur and Y. Bresler. On the optimality of the backward greedy algorithm for the subset selection problem. *SIAM J. Matrix Anal. Appl.*, 21(3):797–808, 2000.

A. d'Aspremont, F. R. Bach, and L. El Ghaoui. Full regularization path for sparse principal component analysis. In *Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML)*, 2007a.

A. d'Aspremont, L. El Ghaoui, M.I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448, 2007b.

D. L. Donoho and J. Tanner. Sparse nonnegative solutions of underdetermined linear equations by linear programming. *Proceedings of the National Academy of Sciences*, 102(27):9446–9451, 2005.

R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

I. T. Jolliffe. Rotation of principal components: choice of normalization constraints. *Journal of Applied Statistics*, 22:29–35, 1995.

I. T. Jolliffe, N.T. Trendafilov, and M. Uddin. A modified principal component technique based on the LASSO. *Journal of Computational and Graphical Statistics*, 12:531–547, 2003.

H.F. Kaiser. The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3): 187–200, 1958.

T. Kato. *Perturbation Theory for Linear Operators*. Springer-Verlag, 1966.

N. Meinshausen and B. Yu. Lasso-type recovery of sparse representations for highdimensional data. Technical report, Technical Report, Statistics Department, UC Berkeley, 2006, 2006.

B. Moghaddam, Y. Weiss, and S. Avidan. Spectral bounds for sparse PCA: Exact and greedy algorithms. *Advances in Neural Information Processing Systems*, 18, 2006a.

B. Moghaddam, Y. Weiss, and S. Avidan. Generalized spectral bounds for sparse LDA. In *Proc. ICML*, 2006b.

B. Moghaddam, Y. Weiss, and S. Avidan. Fast Pixel/Part Selection with Sparse Eigenvectors. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007.

B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, 1995.

JO Neuhaus and C. Wrigley. The quartimax method: an analytical approach to orthogonal simple structure. *British Journal of Statistical Psychology*, 7:81–91, 1954.

W. Rudin. *Real and Complex Analysis, Third edition*. McGraw-Hill, Inc., New York, NY, USA, 1987. ISBN 0070542341.

B.K. Sriperumbudur, D.A. Torres, and G.R.G. Lanckriet. Sparse eigen methods by DC programming. *Proceedings of the 24th international conference on Machine learning*, pages 831–838, 2007.

Y. Su, T. M. Murali, V. Pavlovic, M. Schaffer, and S. Kasif. Rankgene: Identification of diagnostic genes based on expression data. *Bioinformatics*, 19:1578–1579, 2003.

R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal statistical society, series B*, 58(1):267–288, 1996.

M. Yuan and Y. Lin. On the non-negative garrotte estimator. *Journal of The Royal Statistical Society Series B*, 69(2):143–161, 2007.

Z. Zhang, H. Zha, and H. Simon. Low rank approximations with sparse factors I: basic algorithms and error analysis. *SIAM journal on matrix analysis and its applications*, 23(3):706–727, 2002.

P. Zhao and B. Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.

H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational & Graphical Statistics*, 15(2):265–286, 2006.

Journal of Machine Learning Research 9 (2008) 1295-1342 Submitted 7/07; Revised 2/08; Published 7/08

Using Markov Blankets for Causal Structure Learning

Jean-Philippe Pellet[*] JEP@ZURICH.IBM.COM
André Elisseeff AEL@ZURICH.IBM.COM
Data Analytics Group
IBM Zurich Research Laboratory
Säumerstraße 4, CH–8803 Rüschlikon

Editor: David Maxwell Chickering

Abstract

We show how a generic feature-selection algorithm returning strongly relevant variables can be turned into a causal structure-learning algorithm. We prove this under the Faithfulness assumption for the data distribution. In a causal graph, the strongly relevant variables for a node X are its parents, children, and children's parents (or spouses), also known as the Markov blanket of X. Identifying the spouses leads to the detection of the V-structure patterns and thus to causal orientations. Repeating the task for all variables yields a valid partially oriented causal graph. We first show an efficient way to identify the spouse links. We then perform several experiments in the continuous domain using the Recursive Feature Elimination feature-selection algorithm with Support Vector Regression and empirically verify the intuition of this direct (but computationally expensive) approach. Within the same framework, we then devise a fast and consistent algorithm, Total Conditioning (TC), and a variant, TC_{bw}, with an explicit backward feature-selection heuristics, for Gaussian data. After running a series of comparative experiments on five artificial networks, we argue that Markov blanket algorithms such as TC/TC_{bw} or Grow-Shrink scale better than the reference PC algorithm and provides higher structural accuracy.

Keywords: causal structure learning, feature selection, Markov blanket, partial correlation, statistical test of conditional independence

1. Introduction

In this paper, we are interested in using concepts from the feature-selection field to help causal structure learning. Causal structure learning (Pearl, 2000; Spirtes et al., 2001) is a multivariate data-analysis approach that aims to build a directed acyclic graph (DAG) showing direct causal relations among the variables of interest of a given system. These so-called causal graphs can be used together with dedicated rules called *do*-calculus (Pearl, 1995) to predict the effect of interventions, that is, of structural changes in the data-generating process. In this sense, it differs significantly from traditional machine-learning techniques: given a set of interventions, we can predict the behavior of a set of variables whose joint probability distribution has changed since the model was trained.

Building the causal graph is a difficult task, subject to a series of assumptions, and provably correct algorithms have an exponential worst-case complexity. Identifying the exact causal graph is in general impossible. By means of non-interventional data, causal graphs can only be identified up to *observational equivalence*: only adjacencies and so-called V-structures (two independent causes

∗. Also at Pattern Analysis and Machine Learning Group, Swiss Federal Institute of Technology Zurich, Universitätstraße 6, CH–8092 Zurich.

leading to the same effect) can be specified exactly (Pearl, 2000, p. 19). Typical structure-learning algorithms thus return partially directed acyclic graphs (PDAGs). These algorithms can be roughly classified into two categories: the *score-based* algorithms associate a score function with a DAG or PDAG given a training data set and perform, for instance, a greedy search in the space of DAGs or PDAGs (e.g., the GES algorithm, Chickering, 2002); the *constraint-based* algorithms look for dependencies and conditional dependencies in the data and build the causal graph accordingly. Well-known examples are the PC (Spirtes et al., 2001) or the IC (Pearl and Verma, 1991) algorithms. In an effort to get the best of both worlds, other algorithms use both conditional-independence tests and scores to build the network; MMHC (Tsamardinos et al., 2006) is such an example.

The range of data sets that the typical algorithms can deal with is restricted: not any probability distribution can be *faithfully* represented by a DAG. Faithfulness of the distribution is a well-defined condition: it guarantees the existence of a DAG, called a *perfect map*, where there is a one-to-one mapping between the graphical criterion of *d*-separation and conditional independence in the data.[1] Nilsson et al. (2007) discuss faithful distributions and other types of distributions with respect to properties of conditional independence. In the literature, Faithfulness is a precondition to prove correctness of the algorithms.

In practice, both existing score-based and constraint-based techniques deal primarily with discrete data sets. Score-based approaches for continuous variables are computationally expensive;[2] as for the constraint-based approaches, only the multivariate Gaussian case has been dealt with efficiently (Scheines et al., 1995). Margaritis (2005) proposed a distribution-free test of conditional independence, which is very computationally expensive and cannot be readily used with the current constraint-based algorithms for all but very small networks.

Coming from the machine-learning community, feature selection (John et al., 1994; Guyon and Elisseeff, 2003) is a common technique that aims at reducing the number of variables or features used for building more efficient or more robust models. Techniques have evolved to be able to handle nonlinear relationships between variables, redundant variables, in both discrete and continuous domains. Feature selection and causal structure learning are related by a common concept: the *Markov blanket* of a variable X is the smallest set $\mathbf{Mb}(X)$ containing all variables carrying information about X that cannot be obtained from any other variable.[3] In feature selection, this is the set of *strongly relevant* features; that is, of features which carry information about the target that cannot be obtained from any other variable (Kohavi and John, 1997). In a causal graph, this is the set of all parents, children, and spouses of X. The feature-selection task and the causal graph construction task can both be stated to some extent as Markov blanket identification tasks.

Relating feature selection and causal structure learning is not new. Several algorithms identifying the Markov blanket of a single variable with techniques inspired from causal structure learning have been proposed as the optimal solution to the feature-selection problem in the case of a faithful distribution. Tsamardinos and Aliferis (2003) show that for faithful distributions, the Markov blanket of a variable Y is exactly the set of strongly relevant features, and prove its uniqueness. They propose the Incremental Association Markov Blanket (IAMB) algorithm to determine it. With the same Faithfulness assumption, the Min-Max Markov Blanket algorithm (MMMB) (Tsamardinos

1. Conditional independence and *d*-separation are defined formally in Section 2.
2. Computationally tractable methods to learn Bayesian networks from continuous data exist (Fu, 2005), like Bach and Jordan (2003), but do not offer the causality-related theoretical correctness guarantees.
3. Some authors write "Markov blanket" without the notion of minimality, and use "Markov boundary" to note the smallest Markov blanket $\mathbf{Mb}(X)$. Even if defined as minimal, $\mathbf{Mb}(X)$ is generally not unique.

et al., 2003) identifies the Markov blanket of a variable Y by calling a subroutine Min-Max Parents and Children (MMPC). This subroutine finds the direct parents and children of Y with association measures and conditional-independence tests. MMPC is again called on each of these nodes to find potential spouses of Y. False positives are then discarded with conditional-independence tests. MMMB was further discussed by Peña et al. (2005), who propose AlgorithmMB, a similar approach based on scores and conditional-independence tests to retrieve $\mathbf{Mb}(Y)$. The HITON_MB algorithm (Aliferis et al., 2003) is similar in its main steps, and selects an optimal subset of the Markov blanket of a target variable given the Faithfulness assumption. Nilsson et al. (2007) also propose a theoretical algorithm for consistent identification of strongly relevant features in polynomial time for the class of strictly positive distributions. They also argue that some common backward feature-elimination algorithms like Recursive Feature Elimination (Guyon et al., 2002) are actually consistent, in the sense that they return the set of strongly relevant features in the large sample limit.[4]

These are examples of using causal structure learning or similar constraint-based techniques to help feature selection (see Guyon et al., 2007, for a review of those techniques). In this paper, we propose a framework to do the converse. We present a generic approach using the outcome of a consistent feature-selection algorithm *FS* to build an approximate structure of the true causal graph. If we assume that *FS* returns the Markov blanket of the variables, we can show how to turn this approximate result, called *moral graph* (Lauritzen and Spiegelhalter, 1988), into a provably correct PDAG depicting the causal structure. This approach is also used in the Grow-Shrink algorithm (Margaritis and Thrun, 1999), which also builds a moral graph before adjusting the local structure.

This paper contributes a generic algorithm to build a causal graph which clearly separates the Markov blanket identification and the needed local adjustments, an efficient algorithm to perform those adjustments, and two fast instances of the generic algorithm for multivariate Gaussian data sets. This is presented as follows: in Section 2, we first review the needed terms and definitions from feature selection and causality. In Section 3, we make the link from the outcome of a feature-selection algorithm to a causal graph by detailing the needed local adjustments and detail an efficient way to perform them. We directly apply it in Section 4, where we describe how we can build causal graphs using the RFE feature-selection algorithm. As this direct application is very computationally intensive, we then show our more efficient instantiations of the generic algorithm optimized for the multivariate Gaussian case, the TC and TC_{bw} algorithms. We list our experimental results in Section 5, showing through empirical evaluation that Markov blanket algorithms are more scalable and more accurate than the reference PC algorithm. We finally conclude in Section 6 and list proofs in Appendix A.

1.1 Notation

Boldface capitals designate either matrices or sets of random variables or nodes in a graph, depending on the context. $\mathbf{V}$ is the set of all variables in the analysis. Italicized capitals like X, Y, Z are random variables or nodes and elements of $\mathbf{V}$. Vectors are set in boldface lowercase, as $\mathbf{b}$ or $\mathbf{w}$; scalars in italics, as the number of samples n or the number of variables (the problem dimension) d. We indiscriminately write "variable" or "feature" to refer to any variable in the causal analysis or

4. Actually, their definition of consistency has to do with returning the set of features relevant to the Bayes classifier, which is slightly stronger than strong relevance as used here.

any node in a causal graph, and write "predictor" to designate a variable used as input for a given classifier or regression model.

2. Background

We formalize the feature-selection task suited for our needs and provide relevant definitions. We do the same for the causal structure-learning task and prepare the needed basis for drawing the parallels between the two in the next section.

2.1 Feature Selection

We are given a data set of n samples $D = \{(\mathbf{x}_i, y_i), 1 \leq i \leq n\}$. Each data point $(\mathbf{x}_i, y_i)$ has $d-1$ inputs, modeled as a vector $\mathbf{x}_i \in \mathbb{R}^{d-1}$, and an output, or *target*, $y_i \in \mathbb{R}$ (we use $d-1$ and not d for the size of $\mathbf{x}_i$ for consistency with the rest of the paper). The data points are assumed to be drawn i.i.d. from a joint probability distribution over the random variables $\mathbf{V} = \mathbf{X} \cup \{Y\}$. The result of the feature-selection task we are interested in is a set of retained variables $\mathbf{F} \subseteq \mathbf{X}$. How many variables to retain and which variables to retain depends on the particular algorithm, and usually maximizes some tradeoff between efficiency and classification/regression error of a given learning task.

John et al. (1994) propose a classification of the input variables $\mathbf{X}$ with respect to their relevance to the target Y in terms of *conditional independence*.

Definition 1 (Conditional independence) *In a variable set* $\mathbf{V}$*, two random variables* X, Y *are* conditionally independent *given* $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$*, noted* $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$*, if:*

$$\forall x, y, \mathbf{z} : P(X = x \mid Y = y, \mathbf{Z} = \mathbf{z}) = P(X = x \mid \mathbf{Z} = \mathbf{z}),$$

provided that $\forall \mathbf{z} : P(\mathbf{Z} = \mathbf{z}) > 0$.

Conditional independence is a generalization of the traditional notion of statistical independence. If two variables X and Y are independent, then the joint distribution is the product of the marginals: $P(X = x, Y = y) = P(X = x)P(Y = y)$. If they are dependent given some conditioning set $\mathbf{Z}$, then we can write $P(X = x, Y = y \mid \mathbf{Z} = \mathbf{z}) = P(X = x \mid \mathbf{Z} = \mathbf{z})P(Y = y \mid \mathbf{Z} = \mathbf{z})$. Conditional independence is a key concept in Bayesian networks (Pearl, 1988) because of the factorizations of the joint probability distribution it allows.

In feature selection, relevance of predictors to the target as proposed by John et al. (1994) is expressed in terms of conditional independence. In the following definitions, we write X_i to note the ith input variable, and $\mathbf{X}_{\setminus i}$ to note all input variables but the ith one.

Definition 2 (Strong relevance) *A variable* X_i *is* strongly relevant *to the target* Y *if*

$$P(Y \mid \mathbf{X}_{\setminus i}) \neq P(Y \mid \mathbf{X}_{\setminus i}, X_i).$$

A variable is strongly relevant to the target if it carries information about Y that no other variable carries. This is expressed in the definition by a change in the probability distribution of the target between conditioning on all other variables, $\mathbf{X}_{\setminus i}$, and also including X_i in the conditioning set. If X_i carries no exclusive information about Y, the two distributions will be identical.

Definition 3 (Weak relevance) *A variable X_i is* weakly relevant *to the target Y if it is not strongly relevant and*

$$\exists \mathbf{S} \subseteq \mathbf{X}_{\setminus i} : P(Y \mid \mathbf{S}) \neq P(Y \mid \mathbf{S}, X_i).$$

We speak of weak relevance of a variable X_i when there exists a certain context $\mathbf{S}$ in which it carries information about the target. However, this is not necessarily exclusive information, as it may be possible to obtain it from other variables.

Corollary 4 (Irrelevance) *A variable X_i is* irrelevant *to the target Y if it is neither strongly nor weakly relevant, that is, if*

$$\forall \mathbf{S} \subseteq \mathbf{X}_{\setminus i} : P(Y \mid \mathbf{S}) = P(Y \mid \mathbf{S}, X_i).$$

A variable is irrelevant if carries no information about the target at all, no matter what the context is.

For our purposes, we assume that the feature-selection algorithm returns the set of all strongly relevant variables, and only those.[5] (In Section 5, we discuss with experiments whether this is a reasonable assumption with the RFE algorithm.) Put in terms of conditional independence, the result $\mathbf{F}_Y$ of the feature-selection task with target Y is, with $\mathbf{V} = \mathbf{X} \cup \{Y\}$:

$$\mathbf{F}_Y = \left\{ X \;\middle|\; (X \not\perp\!\!\!\perp Y \mid \mathbf{V} \setminus \{X, Y\}) \right\}. \tag{1}$$

That is the set of the variables that are dependent on the target Y, conditioned on all others. We need this property in Section 3 to use the output of the feature-selection task to build a causal graph. Note that if we repeat the feature-selection task using as target another variable $X \in \mathbf{V}$ yielding a result $\mathbf{F}_X$, we have:

$$X \in \mathbf{F}_Y \iff Y \in \mathbf{F}_X. \tag{2}$$

This follows as a direct consequence of (1) due to the symmetry of the conditional-independence relation $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$ with respect to X and Y.

2.2 Causal Structure Learning

In causal structure learning, we are interested in representing graphically conditional dependencies found in the data. Under a set of assumptions, they have a causal interpretation. For this task, we have a data set of n samples $D = \{\mathbf{v}_i, 1 \leq i \leq n\}$. We do not designate a specific target variable in $\mathbf{V}$ as we are interested in learning the full structure of the network.

The graphical representation of choice for causal models is directed acyclic graphs (DAGs) (Pearl, 2000). In a causal graph represented by a DAG, we want to represent direct causal relations with arcs between pairs of variables. Choosing DAGs for this task implies restrictions, an obvious one of which is that causal feedback loops are excluded from the analysis. More formally, the joint probability distribution has to be *faithful* (or *DAG-isomorphic*, Pearl, 1988, p. 128); that is, there must exist a DAG that represents all (conditional) dependencies and independencies entailed by the distribution. Such a graph is called a *perfect map* of the distribution if there is a one-to-one mapping between the conditional-independence relation defined on variables and the *d-separation criterion* defined on the graphical nodes.

5. In the general case, this set can be empty without excluding the existence of other weakly relevant variables (Tsamardinos and Aliferis, 2003). In the next subsection, we detail the Faithfulness hypothesis, which allows us to exclude this particular case.

Definition 5 (*d*-separation) *In a DAG $\mathcal{G}$, two nodes X, Y are* d-separated *by $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$, written $(X \nleftrightarrow Y \mid \mathbf{Z})$, if every path from X to Y is blocked by $\mathbf{Z}$. A path is* blocked *if at least one diverging or serially connected node is in $\mathbf{Z}$ or if at least one converging node and all its descendants are not in $\mathbf{Z}$. If X and Y are not* d-*separated by $\mathbf{Z}$, they are* d-*connected: $(X \leftrightarrow Y \mid \mathbf{Z})$.*

Determining whether two nodes in a graph are *d*-separated given some conditioning set is not visually immediate. It may for instance be unintuitive that whereas conditioning on a node W on a directed path $X \to W \to Y$ blocks the path from X to Y, conditioning on a common child Z of two variables X, Y in $X \to Z \leftarrow Y$ connects them. In the latter case, this common child is called a *collider*. If, furthermore, two parents X, Y of a node Z are nonadjacent in the full graph, then Z is called an *unshielded collider* for the pair (X, Y).

The definition of *d*-separation was worked out by Pearl (1988) to match as closely as possible the complicated nature of the conditional-independence relation with a graphical criterion, so that the class of faithful distributions, which can be represented by a perfect map, is as large as possible, while still keeping a natural graphical representation.

Definition 6 (Perfect map) *A DAG $\mathcal{G}$ is a* directed perfect map *of a joint probability distribution $P(\mathbf{V})$ if there is bijection between* d-*separation in $\mathcal{G}$ and conditional independence in P:*

$$\forall X, Y \in \mathbf{V}, \forall \mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\} : \big((X \nleftrightarrow Y \mid \mathbf{Z}) \iff (X \perp\!\!\!\perp Y \mid \mathbf{Z}) \big). \tag{3}$$

If we take apart the perfect-map equivalence, we distinguish two important concepts, known as the Causal Markov condition and the Faithfulness condition (Spirtes et al., 2001, p. 29).

The **Causal Markov condition** is said to hold for a graph $\mathcal{G} = \langle \mathbf{V}, \mathbf{E} \rangle$ and a probability distribution $P(\mathbf{V})$ if every variable is statistically independent of its graphical non-descendants (intuitively, of its non-effects, direct or indirect) conditional on its graphical parents (intuitively, its direct causes) in P. Pairs $\langle \mathcal{G}, P \rangle$ that satisfy the Causal Markov condition satisfy the implication

$$\forall X, Y \in \mathbf{V}, \forall \mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\} : \big((X \nleftrightarrow Y \mid \mathbf{Z}) \implies (X \perp\!\!\!\perp Y \mid \mathbf{Z}) \big).$$

This is called *I-map property* by Pearl (1988).

The **Faithfulness condition** can be interpreted as the converse of the Causal Markov condition, and states that the only conditional independencies to hold are those specified by the Causal Markov condition:

$$\forall X, Y \in \mathbf{V}, \forall \mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\} : \big((X \leftrightarrow Y \mid \mathbf{Z}) \implies (X \not\perp\!\!\!\perp Y \mid \mathbf{Z}) \big).$$

If the Causal Markov and Faithfulness conditions hold together for a pair $\langle \mathcal{G}, P \rangle$, then we find again the equivalence (3), and $\mathcal{G}$ is a perfect map of P.

In practice, the Causal Markov condition is used by the so-called constraint-based algorithms to perform conditional-independence tests on the data and build the graph accordingly, and Faithfulness is assumed to prove that the graph is correct. Hausman and Woodward (1999) discuss and explain in more detail the Causal Markov condition, and Steel (2005) discusses the Faithfulness condition and its motivations, pointing out cases where it can be violated. While the former is in general not violated simply by construction of the causal graph, violation of the latter occurs if the probability distribution is not faithful. A simple example is the *n*-bit parity problem where the prior probability of each bit is uniform, of which the XOR problem is a special case: each variable is

unconditionally independent of every other, but any variable pair becomes dependent conditioned on all other variables. On this problem, current constraint-based algorithms yield an empty graph because of the pairwise unconditional independencies, although it is not true that the data shows no dependency at all since one variable is a well-defined function of all others.

From this point on and for all proofs, we assume that the working data set D has a distribution that does not violate Faithfulness, and that it can thus be represented by a perfect map. In such a context, however, it is still not clear that causation can be inferred from conditional independence. We now proceed to explain the relation between causation and conditional independence.

Assuming Faithfulness, direct causation between X and Y, noted $X \dashrightarrow Y$, implies that X and Y are dependent given any conditioning set (Pearl and Verma, 1991, see definitions of potential and genuine causes):

$$X \dashrightarrow Y \implies \big(\forall \mathbf{S} \subseteq \mathbf{V} \setminus \{X,Y\} : (X \not\perp\!\!\!\perp Y \mid \mathbf{S})\big).$$

We denote the absence of direct causation by $X \not\dashrightarrow Y$. The exact converse of this implication does not hold. If we make the **Causal Sufficiency assumption** (Spirtes et al., 2001), that is, assume that no hidden common cause of two variables exists, we can write:

$$\big(\forall \mathbf{S} \subseteq \mathbf{V} \setminus \{X,Y\} : (X \not\perp\!\!\!\perp Y \mid \mathbf{S})\big) \implies X \dashrightarrow Y \text{ or } Y \dashrightarrow X. \tag{4}$$

Using (4), we can theoretically determine all adjacencies of the causal graph with conditional-independence tests, but we cannot orient the edges. But there is a special causation pattern where conditional-independence tests can reveal the direction of causation. It is known as a **V-structure** (Pearl, 2000): two common causes X, Y, initially independent,[6] become dependent when conditioned on a common effect Z, then acting as a collider. This is noted $X \dashrightarrow Z \dashleftarrow Y$, where we also require $X \not\dashrightarrow Y$ and, symmetrically, $Y \not\dashrightarrow X$. Formally, we have:

$$\begin{aligned} X \dashrightarrow Z \dashleftarrow Y \text{ and } X \not\dashrightarrow Y \text{ and } Y \not\dashrightarrow X \\ \implies \big(\exists \mathbf{S} \subseteq \mathbf{V} \setminus \{X,Y,Z\} : (X \perp\!\!\!\perp Y \mid \mathbf{S}) \text{ and } (X \not\perp\!\!\!\perp Y \mid \mathbf{S} \cup \{Z\})\big). \end{aligned}$$

The exact converse does not hold either. But using (4), we can find an equivalence relation defining a V-structure, still assuming Causal Sufficiency: first, we certify the existence of a link between X and Z and between Y and Z. Z is then identified as an unshielded collider if conditioning on it creates a dependency between X and Y:

$$\begin{aligned} X \dashrightarrow Z \dashleftarrow Y \iff \Big(&\big(\exists \mathbf{S} \subseteq \mathbf{V} \setminus \{X,Y,Z\} : (X \perp\!\!\!\perp Y \mid \mathbf{S}) \text{ and } (X \not\perp\!\!\!\perp Y \mid \mathbf{S} \cup \{Z\})\big) \\ &\text{and } \big(\forall \mathbf{S} \subseteq \mathbf{V} \setminus \{X,Z\} : (X \not\perp\!\!\!\perp Z \mid \mathbf{S})\big) \\ &\text{and } \big(\forall \mathbf{S} \subseteq \mathbf{V} \setminus \{Y,Z\} : (Y \not\perp\!\!\!\perp Z \mid \mathbf{S})\big)\Big). \end{aligned} \tag{5}$$

Actually, typical algorithms first establish the existence of a link between two variables by seeking a certificate equivalent to, or implicating the premise of, (4), and then look for orientation possibilities. Note that there is no guarantee that all links can be oriented into causal arcs, and that in

6. The two causes X and Y actually do not need to be unconditionally independent, but there must exist a (possibly empty) separating set $\mathbf{S}_{XY} \subseteq \mathbf{V} \setminus \{X,Y\}$ such that $(X \perp\!\!\!\perp Y \mid \mathbf{S}_{XY})$ for the collider to be identifiable. This implies that no direct causation $X \dashrightarrow Y$ or $Y \dashrightarrow X$ may exist: the collider must be unshielded.

general we therefore cannot recover the full causal structure with conditional-independence tests. This is the problem known as **causal underdetermination** (Spirtes et al., 2001, p. 62): for the structure-learning task given observational data, a correct graph is specified by its adjacencies and its V-structures only. Partially oriented graphs returned by structure-learning algorithms represent *observationally equivalent classes* of causal graphs (Pearl, 2000, p. 19). This means that for a given joint probability distribution $P(\mathbf{V})$, the set of all conditional-independence statements that hold in P does not yield a unique perfect map in general.

Formally, if we combine (3), (4) and (5), we find, for a perfect causal map $\mathcal{G}$ (using the symbol "$\multimap$" to denote direct causation and "$\rightarrow$" to denote an arc in the graph):

$$\begin{aligned} X,Y \text{ adjacent in } \mathcal{G} &\iff X \multimap Y \text{ or } Y \multimap X \\ X \rightarrow Z \leftarrow Y &\iff X \multimap Z \leftarrow Y. \end{aligned} \tag{6}$$

It is sometimes possible to orient further arcs in a graph by looking at already-oriented arcs and propagating constraints, preventing acyclicity and the creation of additional V-structures other than those already detected. The graph after this constraint-propagation step is called *completed PDAG*, *maximally oriented PDAG* (CPDAG), or *essential graph*, depending on the author.

3. Causal Network Construction Based on Feature Selection

We have looked at the ideal outcome of feature selection in (1) and how to read a causal graph in (6). We now turn to showing how feature selection can be used to build a causal graph. From now on and for the rest of this paper, we assume that the joint probability distribution over all variables $\mathbf{V}$ is faithful.

3.1 Identifying the Markov Blankets

In the context of directed graphical models, the Markov blanket of a node X, noted $\mathbf{Mb}(X)$, is the set of parents, children, and children's parents (spouses) of X. As an easy property, note that we have:

$$X \in \mathbf{Mb}(Y) \iff Y \in \mathbf{Mb}(X).$$

The following statement is a key property of Markov blankets.

Property 7 (Total conditioning) *In the context of a faithful causal graph $\mathcal{G}$, we have:*

$$\forall X,Y \in \mathbf{V}: \left(X \in \mathbf{Mb}(Y) \iff (X \not\perp\!\!\!\perp Y \mid \mathbf{V} \setminus \{X,Y\})\right).$$

(See Appendix A for the proof.) This property says that the Markov blanket of each node is the set of all variables that are dependent on it, conditioned on all other variables. In other words, in a causal graph, the parents, children, and spouses of Y store information about Y that cannot be obtained from any other variable. Note that $\mathbf{Mb}(Y)$ then has exactly the property of the output of feature selection, $\mathbf{F}_Y$, as characterized in (1). This links feature selection and causal structure learning in the sense that

$$\mathbf{F}_Y = \mathbf{Mb}(Y),$$

the Faithfulness assumption guaranteeing the unicity of $\mathbf{Mb}(Y)$. However, Markov blankets alone do not fully specify a causal graph. Thus, feature selection, even if guaranteed to find only strongly relevant features, cannot be directly used to construct the graph as we want it to be. The problem is that spouses of Y, even if not directly linked in the original graph, would be linked in $\mathbf{F}_Y$ and $\mathbf{Mb}(Y)$. An additional step is needed to transform the Markov blankets into parents, children, and spouses.

3.2 Recovering the Local Structure

The result of feature selection can be graphically shown by an undirected graph $\mathcal{G} = \langle \mathbf{V}, \mathbf{E} \rangle$ where $(X,Y) \in \mathbf{E} \Leftrightarrow X \in \mathbf{F}_Y$. This graph is close to the original causal graph in that it contains all arcs as undirected links, and additionally links spouses together, and is called the *moral graph* of the original directed graph (Lauritzen and Spiegelhalter, 1988, p. 166). The extra step needed to transform this graph into a causal PDAG is the deletion of the spouse links and the orientation of the arcs, a task which we call "resolving the Markov blankets."

An existing algorithm can resolve the Markov blankets, that is, use Markov blanket information to infer the local structure around a node: the Grow-Shrink (GS) algorithm, proposed by Margaritis and Thrun (1999). The full algorithm first finds the Markov blanket for each variable, and performs further conditional-independence tests around each variable to infer the structure locally. It then uses a heuristics to remove cycles possibly introduced by previous steps. We list in Algorithm 1 (using our notation) the steps of the algorithm responsible for building the local structure using the Markov blanket information, as this is exactly the task we are trying to solve. In the code, $\mathbf{Bd}(X)$ stands for the *boundary* of X; that is, the set of its direct neighbors in the graph $\mathcal{G}$. It is different from $\mathbf{Mb}(X)$ in that whereas $\mathbf{Mb}(X)$ is passed as input to the algorithm and is fixed, $\mathbf{Bd}(X)$ depends on the graph $\mathcal{G}$, which is modified throughout the algorithm. We note a conditional-independence test with a subroutine call to CONDINDEP$(X,Y,\mathbf{Z})$: ideally, this function returns *true* when $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$ holds, and *false* otherwise. More will be said about the actual implementation of such tests in Section 4. The command **break** is used to break out of the innermost loop, saving unnecessary computations.

The GS algorithm makes two passes over all variables and the members of their Markov blankets (or direct neighbors in the second pass). It first removes the possible spouse links between linked variables X and Y by looking for a d-separating set around X and Y. In a second pass, it orients the arcs whenever it finds that conditioning on a middle node creates a dependency. While searching for the appropriate conditioning set, GS selects the smallest base search set (set **B** in Algorithm 1) for each phase. This has two very desirable effects. First, it reduces the number of tests, which is useful because each phase contains a subset search, exponential in time complexity with respect to the searched set. Second, it reduces the average size of the conditioning set, which increases the power of the statistical tests, and thus helps reduce the number of Type II errors.

While the GS approach considerably reduces the number of tests to be performed with respect to a large subset search, it is possible to perform fewer tests while still identifying correctly the structure and orienting the arcs, and decreasing the average conditioning set size. A helpful observation is that orientation and removal of the spouse links can be done together in a single pass. We know, as discussed in the previous section, that only arcs in V-structures can be oriented: fortunately, V-structures are exactly spotted when we identify a spouse link to be removed. Two spouses X and Y that are not directly linked in the original causal graph can be d-separated by some set of

Algorithm 1 Resolve the Markov Blankets with the Grow-Shrink Algorithm

```
1:  procedure ResolveMarkovBlankets_GrowShrink
      Input:  Mb(·) :  the Markov blanket information for each node X ∈ V
      Output:  G :  partially oriented DAG
      /* Compute graph structure */
2:    G ← moral graph according to Mb(·)
3:    for each X ∈ V and Y ∈ Mb(X) do
4:      B ← smallest set of {Bd(X) \ {Y}, Bd(Y) \ {X}}
5:      for each S ⊆ B do
6:        if CondIndep(X, Y, S) then remove link X − Y from G; break
7:      end for
8:    end for
      /* Orient edges */
9:    for each X ∈ V and Y ∈ Bd(X) do
10:     for each Z ∈ Bd(X) \ Bd(Y) \ {Y} do
11:       orient Y → X    /* to be corrected if a test yields independence */
12:       B ← smallest set of {Mb(Y) \ {Z}, Mb(Z) \ {Y}}
13:       for each S ⊆ B do
14:         if CondIndep(Y, Z, S ∪ {X}) then remove orientation Y → X; break
15:       end for
16:       if Y → X then break
17:     end for
18:   end for
19:   return G
20: end procedure
```

nodes. Thus, if we can find a set $\mathbf{S}_{XY}$ that makes X and Y conditionally independent, we know that the link between them is a spouse link to be removed. Moreover, we know that any node Z part of the intersection of their Markov blankets not included in $\mathbf{S}_{XY}$ is a collider and thus a common child, and that the triplet (X,Z,Y) is actually a V-structure $X \rightarrow Z \leftarrow Y$ in the original graph. This follows from the definition of d-separation. What we need is an efficient search algorithm to find such d-separating sets.

An approach based on this observation has two main benefits. First, it only searches the triangles, that is, the cliques of three nodes, in the moral graph. Assuming that the information about the Markov blanket is correct, only triangles can hide spouse links and V-structures. Second, for each connected pair $X - Y$ in a triangle, decisions about possible spouse links and arc orientation are taken together and thus faster.

Pseudocode for the proposed search algorithm is listed in Algorithm 2, where the notation $\mathcal{G}^{\setminus XY}$ denotes the moral graph $\mathcal{G}$ where all direct links involving X or Y have been removed. The algorithm uses the following concept.

Definition 8 (Collider sets) *In an undirected graph $\mathcal{G} = \langle \mathbf{V}, \mathbf{E} \rangle$, let* $\mathbf{Tri}(X - Y)$ *(with $X, Y \in \mathbf{V}$ and $(X, Y) \in \mathbf{E}$) be the set of vertices forming a triangle with X and Y:*

$$\mathbf{Tri}(X - Y) = \{Z \in \mathbf{V} \mid (X, Z) \in \mathbf{E}, (Y, Z) \in \mathbf{E}\}.$$

Suppose that $\mathcal{G}$ is the moral graph of the DAG representing the causal structure of a faithful data set. A set of vertices $\mathbf{Z} \subseteq \mathbf{Tri}(X - Y)$ then has the Collider Set *property for the pair (X, Y) if it is the largest set that fulfills*

$$\exists \mathbf{S}_{XY} \subseteq \mathbf{V} \setminus \{X, Y\} \setminus \mathbf{Z} : (X \perp\!\!\!\perp Y \mid \mathbf{S}_{XY}) \tag{7}$$

$$\text{and} \quad \forall Z_i \in \mathbf{Z} : (X \not\perp\!\!\!\perp Y \mid \mathbf{S}_{XY} \cup \{Z_i\}). \tag{8}$$

The set $\mathbf{S}_{XY}$ is then a d*-separating set for X, Y.*

Lemma 9 *In the context of a faithful causal graph, the set $\mathbf{Z}$ that has the Collider Set property for a given pair (X, Y) exists if and only if X is neither a direct cause nor a direct effect of Y. This set $\mathbf{Z}$ is unique when it exists. (Proof in Appendix A.)*

The purpose of Algorithm 2 is thus to find these collider sets (in the pseudocode, the symbol $\subsetneq$ denotes the strict subset relation). The algorithm loops over all triangle links and performs a collider set search for each of them. Let $X - Y$ be one of these links: if it is not a spouse link, the search procedure will leave the value of the d-separating set $\mathbf{S}_{XY}$ to its default value, **null**. Otherwise, $\mathbf{S}_{XY}$ will be set to a (possibly empty[7]) set for X and Y. The collider set can be inferred by removing the d-separating set from the triangle nodes $\mathbf{Tri}(X - Y)$: as $\mathbf{Tri}(X - Y)$ contains nodes on a path of length 2 between X and Y, finding a d-separating set that does not contain some of these nodes proves that they can only be colliders according to the definition of d-separation.[8] For instance, if the procedure produces an empty set for a given linked pair $X - Y$, then X and Y are unconditionally independent, and therefore all nodes in $\mathbf{Tri}(X - Y)$ are colliders.

Two caveats have to be observed during this search, however. First, there might be other active, d-connecting paths between X and Y that are not going through any node of $\mathbf{Tri}(X - Y)$. Those nodes must be blocked by appropriate conditioning on the boundary of X or Y as determined by the base conditioning set at line 6. Second, this base conditioning set must be checked not to include any descendant of possible colliders. If it did, it would open a d-connecting path according to Definition 5. This check is performed at lines 13 to 21. At line 13, we build a set $\mathbf{D}$ that includes all possible descendants of currently conjectured colliders that intersect our base conditioning set $\mathbf{B}$. The following loop makes sure none of them was opening a path between X and Y.

Theorem 10 *In the large sample limit, for faithful, causally sufficient data sets, the procedure* RESOLVEMARKOVBLANKETS_COLLIDERSETS *correctly identifies all V-structures and all spouse links, assuming consistent statistical tests. (Proof in Appendix A.)*

This procedure is best understood with a graphical example. Consider the sample local structure in Figure 1, imagine it is part of a larger network, and suppose we are performing the search

7. Note that returning an empty d-separating set in $\mathbf{S}_{XY}$ is different from returning **null**, signaling the absence of any such set.
8. The next paragraphs describe patterns where this is not true and show how the algorithm still deals with them correctly.

Algorithm 2 Resolve the Markov Blankets with Collider Sets

1: **procedure** RESOLVEMARKOVBLANKETS_COLLIDERSETS
Input: $\mathbf{Mb}(\cdot)$: the Markov blanket information for each node $X \in \mathbf{V}$
Output: $\mathcal{G}$: partially oriented DAG
2: $\mathcal{G} \leftarrow$ moral graph according to $\mathbf{Mb}(\cdot)$
3: $\mathbf{C} \leftarrow \{\}$, an empty list of orientation directives
4: **for each** edge $X - Y$ part of a fully connected triangle **do**
5: $\mathbf{S}_{XY} \leftarrow$ **null** /* search for d-separating set */
6: $\mathbf{B} \leftarrow$ smallest set of $\{\mathbf{Bd}(X) \setminus \mathbf{Tri}(X-Y) \setminus \{Y\}, \mathbf{Bd}(Y) \setminus \mathbf{Tri}(X-Y) \setminus \{X\}\}$
7: **for each** $\mathbf{S} \subsetneq \mathbf{Tri}(X-Y)$ **do** /* subset search */
8: $\mathbf{Z} \leftarrow \mathbf{B} \cup \mathbf{S}$
9: **if** CONDINDEP$(X, Y, \mathbf{Z})$ **then**
10: $\mathbf{S}_{XY} \leftarrow \mathbf{Z}$
11: **break** to line 23
12: **end if**
13: $\mathbf{D} \leftarrow \mathbf{B} \cap \left\{\text{nodes reachable by } W \text{ in } \mathcal{G}^{\setminus XY} \mid W \in \left(\mathbf{Tri}(X-Y) \setminus \mathbf{S}\right)\right\}$
14: $\mathbf{B}' \leftarrow \mathbf{B} \setminus \mathbf{D}$
15: **for each** $\mathbf{S}' \subsetneq \mathbf{D}$ **do** /* descendant of collider may be opening a path */
16: $\mathbf{Z} \leftarrow \mathbf{B}' \cup \mathbf{S}' \cup \mathbf{S}$
17: **if** CONDINDEP$(X, Y, \mathbf{Z})$ **then**
18: $\mathbf{S}_{XY} \leftarrow \mathbf{Z}$
19: **break** to line 23
20: **end if**
21: **end for**
22: **end for**
23: **if** $\mathbf{S}_{XY} \neq$ **null then** /* save orientation directive */
24: mark link $X - Y$ as spouse link in $\mathcal{G}$
25: **for each** $Z \in \left(\mathbf{Tri}(X-Y) \setminus \mathbf{S}_{XY}\right)$ **do**
26: $\mathbf{C} \leftarrow \mathbf{C} \cup \{(X \rightarrow Z \leftarrow Y)\}$
27: **end for**
28: **end if**
29: **end for**
30: remove all spouse links (i.e., marked links) from $\mathcal{G}$
31: **for each** orientation directive $(X \rightarrow Z \leftarrow Y) \in \mathbf{C}$ **do** /* orient edges */
32: **if** edges $X - Z$ and $Y - Z$ still exist in $\mathcal{G}$ **then**
33: orient edges as $X \rightarrow Z \leftarrow Y$
34: **end if**
35: **end for**
36: **return** $\mathcal{G}$
37: **end procedure**

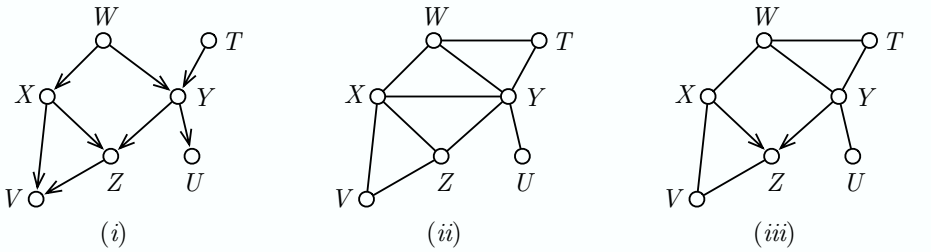


Figure 1: Sample local causal structure (i) and corresponding moral graph (ii). On (iii), the spouse link and orientation information that the collider set search for the linked pair $X-Y$ gives.

starting at line 5 in Algorithm 2. We are looking for a d-separating set for X and Y. Looking at the original graph, we see that $\{W\}$ is the smallest such set; let us see how the algorithm finds it. We have: $\mathbf{Tri}(X-Y)=\{W,Z\}$, $\mathbf{Bd}(X)=\{W,Y,Z,V\}$ and $\mathbf{Bd}(Y)=\{W,X,Z,U,T\}$. The base conditioning set $\mathbf{B}$ will thus be the smallest of $\big\{\{V\},\{U,T\}\big\}$, thus $\mathbf{B}=\{V\}$. At this stage, conditioning on V is justifiable: one cannot exclude situations where X and Y are d-connected given the empty set through T and V, for instance if T and V both had a common cause farther away in the network. But actually in this example, all (perfect) tests containing V in the conditioning set will yield dependence, because it is a descendant of the collider Z and thus opens a path by definition of d-separation. Eventually, in the iteration where $\mathbf{S}=\{W\}$, we will find conditional independence in the nested loop at lines 15 to 21. As $\mathbf{Tri}(X-Y)\setminus\mathbf{S}=\{Z\}$, $\mathbf{D}$ will be assigned the value $\{V\}$ and $\mathbf{B}'$ will be empty, so that we will perform exactly one extra test at line 17 with the conditioning set $\mathbf{S}_{XY}=\{W\}$, which yields independence. This in turn allows us to identify the link $X-Y$ as a spouse link and determine (line 25) that the set $\mathbf{Tri}(X-Y)\setminus\mathbf{S}_{XY}=\{Z\}$ is the set of all direct effects of X and Y; that is, fulfills the Collider Set property.

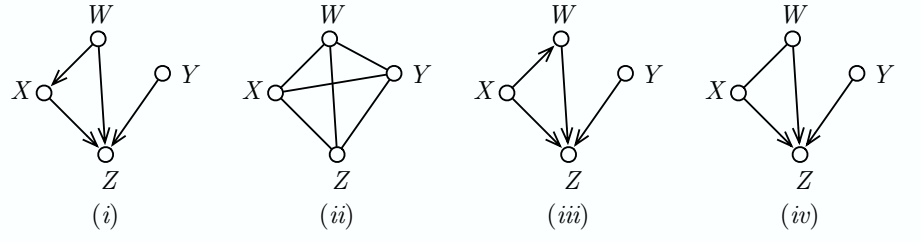


Figure 2: Another sample local causal structure (i) and corresponding moral graph (ii). On (iii), a wrong result if orientation is done immediately at line 26 of Algorithm 2. On (iv), the correct (non-)orientation if the condition at line 32 is added.

For some structures, the order in which arcs are removed and oriented must happen such that all spouse links are removed before proceeding to orientation. Consider another example, shown in Figure 2, and suppose again that that we are looking for a d-separating set for the pair (X,Y). As X and Y are unconditionally independent, $\mathbf{S}_{XY}=\emptyset$ is a valid d-separating set. We may thus remove the link $X-Y$, and considering that $\mathbf{Tri}(X-Y)=\{W,Z\}$, we could want to orient $X\rightarrow Z\leftarrow X$ and

$X \rightarrow W \leftarrow X$ (leaving the spouse link $W - Y$ to be removed later). This would be wrong, precisely because $W - Y$ is a spouse link, and thus the orientation $X \rightarrow W \leftarrow X$ is not allowed if one of the links to be oriented does not actually exist in the original graph. This is the reason why all orientation directives are saved in a list $\mathbf{C}$ at line 26 of Algorithm 2. After all spouse links have been removed, the orientations are done at line 33 only when both links to be oriented still exist, thus ensuring the existence of the V-structure $X \rightarrow Z \leftarrow Y$.

We do not claim that our algorithm uses the smallest possible conditioning set for the tests. There is a tradeoff between obtaining the minimal possible conditioning set and keeping the total number of tests low in the average case. In the empirical evaluation of this algorithm, we examine three behavioral criteria: the total number of tests, the average size of the conditioning set, and the maximum size of the conditioning set.

The complexity of the whole algorithm iterating over all triangle links, in terms of number of calls to CONDINDEP, is $O(d^2 2^{\alpha})$, where d is the number of variables and $\alpha = \max_{X \in \mathbf{V}} |\mathbf{Mb}(X)| - 1$. In the worst case of a fully connected graph, where $\mathbf{Mb}(X) = \mathbf{V} \setminus \{Y\}$, it is exponential in the number of variables due to the subset search. But in practice, the original graphs are often sparse enough so that the actual run time is not exponential. Many algorithms (e.g., MMMB, HITON_MB, AlgorithmMB, GS) perform subset searches in the (possibly augmented) Markov blanket and thus rely on graph sparseness to be efficient. Although the complexity of RESOLVEMARKOVBLANKETS_COLLIDERSETS is the same as that of RESOLVEMARKOVBLANKETS_GROWSHRINK, we show in the experimental results in Section 5 that the former performs fewer tests with a smaller average conditioning set size, while still providing comparable accuracy in structure learning.

3.3 A Generic Algorithm Based on Feature Selection

Thanks to the subroutine explained in the previous section, we can now draft a generic algorithm for structure learning based on feature-selection methods returning strongly relevant features. Algorithm 3 lists pseudocode for the three main steps of this approach:

1. Find the conjectured Markov blanket of each variable with feature selection and build the moral graph;
2. Remove spouse links and orient V-structures using collider sets;
3. Propagate orientation constraints.

For the sake of completeness, the constraint propagation rules of Step 3 have also been listed, in a separate subroutine (see Algorithm 4). They are common in structure learning to obtain a completed PDAG (Pearl and Verma, 1991). Meek (1995) proves that these three rules indeed return the maximally oriented graph when given a PDAG whose V-structures are oriented.

The challenge with this approach is twofold. One issue is efficiency: consistent but slow feature-selection algorithms will not beat existing causal learning algorithms, as they have to be run as many times as the number of variables d. The second and biggest issue is that consistent feature-selection algorithms are needed in order to prove correctness of this generic algorithm, in the sense that the result of feature selection should be equal to the set of strongly relevant features. This requirement is not always fulfilled. Hardin et al. (2004) study an SVM classifier and discuss feature selection based on the $\mathbf{w}$ weights: although irrelevant variables are not selected in the large sample limit, they show that the weights of the weakly relevant variables can be as close as one wishes to that of the strongly relevant variables due to the large-margin behavior of SVMs. Forward feature selection has been

Algorithm 3 Causal Structure Learning with Feature Selection

procedure GENERICSTRUCTURELEARNING
  Input: D : $n \times d$ data set with n d-dimensional data points
  Output: $\mathcal{G}$: maximally oriented partially directed acyclic graph

  /* Step 1: Markov blanket construction */
  for each variable $X \in \mathbf{V}$ **do**
    $\mathbf{F}_X \leftarrow$ FEATURESELECTIONALGORITHM(X, D)
  end for
  for each pair (X, Y) such that $Y \in \mathbf{F}_X$ and $X \in \mathbf{F}_Y$ **do** /* symmetry check */
    add X to $\mathbf{Mb}(Y)$ and Y to $\mathbf{Mb}(X)$
  end for

  /* Step 2: Spurious arc removal & V-structure detection */
  $\mathcal{G} \leftarrow$ RESOLVEMARKOVBLANKETS$(\mathbf{Mb}(\cdot))$

  /* Step 3: Constraint propagation */
  $\mathcal{G} \leftarrow$ COMPLETEPDAG$(\mathcal{G})$
  return $\mathcal{G}$
end procedure

Algorithm 4 Orient a PDAG maximally

procedure COMPLETEPDAG
  Input: $\mathcal{G}$: partially directed acyclic graph
  Output: $\mathcal{G}$: maximally oriented partially directed acyclic graph

  while $\mathcal{G}$ is changed by some rule **do** /* fixed-point iteration */
    for each X, Y, Z such that $X \rightarrow Y - Z$ **do**
      orient as $X \rightarrow Y \rightarrow Z$ /* no new V-structure */
    end for
    for each X, Y such that $X - Y$ and $\exists$ directed path from X to Y **do**
      orient as $X \rightarrow Y$ /* preserve acyclicity */
    end for
    for each X, Y s.t. $X - Y$ & $\exists$nonadj. Z, W s.t. $X - Z \rightarrow Y$ & $X - W \rightarrow Y$ **do**
      orient as $X \rightarrow Y$ /* three-fork V with married parents */
    end for
  end while
  return $\mathcal{G}$
end procedure

shown to miss strongly relevant variables (Guyon and Elisseeff, 2003). Nilsson et al. (2007) also describe forward selection as inconsistent, but claim that backward feature elimination is actually consistent in the large-sample limit.[9] For finite data sets, Statnikov et al. (2006) further show (among others) that even the weights of the irrelevant variables can get bigger than that of relevant variables, and that weakly relevant variables can be selected more often than strongly relevant variables in some cases.

These considerations are taken into account in our approach. In the next section, we describe an instantiation of the generic algorithm with an existing backward feature-elimination algorithm. Expecting the feature selection to be too inclusive, that is, to include features that are not strongly relevant, we add the filtering condition at line 5 of the generic outline in Algorithm 3: in order to link X and Y in the moral graph, we require the feature selection performed for X to have selected variable Y, and conversely, we require X to have been selected by the feature selection performed for Y. This does not theoretically guarantee the absence of "false positives," however. Further in the section, we replace the feature-selection step with a provably consistent algorithm in the multivariate Gaussian case, and analyze its complexity and behavior.

4. Algorithms for Causal Feature Selection

In this section, we show two algorithms (and a variant) as instantiations of the generic approach previously described. First, we explain an algorithm based on the Recursive Feature Elimination (RFE) algorithm (Guyon et al., 2002) as a direct application of existing methods. We then describe Total Conditioning (TC), a fast algorithm that can be proved correct under the multivariate Gaussian assumption. We also show a variant, TC_{bw}, that improves accuracy with low sample sizes by using an explicit backward feature-selection heuristics. In Section 5, we report on experiments including these algorithms.

4.1 An RFE-Based Approach

To empirically test the soundness of the approach, we propose to use RFE over a Support Vector Regression (SVR) learner (Smola and Schölkopf, 1998) with a linear kernel, assuming for this example that we will deal with multivariate Gaussian data. RFE is an instance of a backward feature-elimination algorithm. Given some learner (in this case, SVR), it iteratively trains it, ranks the features according to some criterion, and remove the feature (or the p features) with the smallest ranking criterion. This criterion can be the weights $\mathbf{w}$ attributed to the features by the learner, or some sensitivity measure of the features (Guyon et al., 2002). In our case, we used the weights $\mathbf{w}$ of SVR as described in Smola and Schölkopf (1998).

Using RFE, the Markov blanket identification is done in two steps:

1. Use RFE to rank the predictors according to their weights in the trained model and to provide what can be seen as a relevance ordering of the predictors;

2. Determine the size of the Markov blanket and thus the number of variables to select from the list returned by RFE.

9. This is subject to the assumption that the underlying classifier must itself be consistent, in the sense that it must return the Bayes classifier in the large-sample limit.

We do not have a theoretical guarantee that RFE/SVR will return the Markov blanket variables. Although Nilsson et al. (2007) shows that RFE/SVM as described in Guyon et al. (2002) is consistent (i.e., returns strongly relevant variables in the large-sample limit), the limitations of ranking variables on the **w** weights of an SVM with finite data sets have also been highlighted (Hardin et al., 2004; Statnikov et al., 2006). For now, we thus use this feature-selection step as a heuristics.

In order to determine the number of variables to select from the ranked list returned by RFE, we use the following criterion: starting with the first variable from the list, accept a new variable in the Markov blanket if the cross-validated training error of the SVR decreases with the new variable, and stop and return the current list if adding the next variable increases the error.

Algorithm 5 An RFE-Based Feature-Selection Step

1: **procedure** RFEFEATURESELECTION
   Input: X : the target variable to perform feature selection for
   D : $n \times d$ data set with n d-dimensional data points
   Output: $\mathbf{S}$: the set of selected variables
2: $\mathbf{w} \leftarrow$ weights of $\mathbf{V} \setminus X$ according to RFE(SVR)
3: $\mathbf{P} \leftarrow$ predictor variables sorted according to $\mathbf{w}$
4: $\mathbf{S} \leftarrow \emptyset$
5: $error_{opt} \leftarrow \text{var}[X]$ /* MSE of constant function */
6: $error \leftarrow$ TRAIN(cross-validated SVR with predictor $(\mathbf{P})_1$)
7: **while** $error < error_{opt}$ **do**
8: $error_{opt} \leftarrow error$
9: $\mathbf{S} \leftarrow \mathbf{S} \cup \{(\mathbf{P})_1\}$ /* add beneficial predictor */
10: $\mathbf{P} \leftarrow \mathbf{P} \setminus \{(\mathbf{P})_1\}$
11: $error \leftarrow$ TRAIN(cross-validated SVR with predictors $\mathbf{S} \cup \{(\mathbf{P})_1\}$)
12: **end while**
13: **return S**
14: **end procedure**

The symmetry condition (2), $X \in \mathbf{F}_Y \Leftrightarrow Y \in \mathbf{F}_X$, might not be satisfied: we rely on the check at line 5 of the generic approach of Algorithm 3 to make sure that we do not select spurious features in the Markov blanket. This conservative approach implies that we expect RFE to select at least all strongly relevant variables, plus possibly some others that we hope to identify with this simple condition.

As a conditional-independence test at lines 9 and 17 of the collider set search in Algorithm 2, we can use the distribution-free Recursive Median (RM) algorithm proposed by Margaritis (2005) to detect the V-structure and remove the spouse links, or a z-test as used in Scheines et al. (1995) in the case of Gaussian data.

Although we expect the resulting graph to be accurate in the large sample limit (see Section 5), we also expect the run time of such an approach to be much higher compared to existing algorithms. Training the SVR has a cubic complexity in terms of the number of samples, $O(n^3)$. To get an accurate ranking, RFE runs the training $d-1$ times. Then, a new SVR learner is trained and cross-validated several times (we used a 5-fold cross-validation) to get the validation error, which is repeated for each variable in the actual Markov blanket. The complexity for the whole feature-selection step is then $O(d^2n^3)$, with a large constant factor. We thus emphasize that this RFE-based

feature selection is not meant as a valid practical instantiation of the generic algorithm, but rather as a proof of concept to validate the approach. In order to be practical, the feature-selection step has to be redesigned so that it is done efficiently when run for all variables. This is what the next algorithm is meant to address in the specific case of multivariate Gaussian variables.

4.2 The TC Algorithm

We now propose in the procedure TCFEATURESELECTION (Algorithm 6) another instantiation of the feature-selection call at line 3 of the generic approach of Algorithm 3. The whole algorithm as determined by the feature-selection, collider-identification, and maximal-orientation steps is equivalent to the TC algorithm described in Pellet and Elisseeff (2007). (We thus write "TC" to refer to the whole algorithm and not only to the feature-selection procedure, referred to as TCFEATURESELECTION.)

For a given target variable X, TC estimates the coefficients of a multiple regression problem, considering all other variables $\mathbf{V} \setminus X$ as predictors. It then returns the significant predictors, according to a t-test on the coefficient of each variable. Its short listing is in Algorithm 6.

Algorithm 6 The Total Conditioning Feature-Selection Step

1: **procedure** TCFEATURESELECTION
   Input: X : the target variable to perform feature selection for
   D : $n \times d$ data set with n d-dimensional data points
   Output: $\mathbf{S}$: the set of selected variables
2: $\mathbf{b} \leftarrow$ weights of $\mathbf{V} \setminus X$ in the problem of regressing X on $\mathbf{V} \setminus X$
3: $\mathbf{S} \leftarrow \{$predictors whose b weight is significant$\}$
4: **return S**
5: **end procedure**

The conditional-independence tests to be performed at lines 9 and 17 of the collider set search of Algorithm 2 are done using partial correlation.

Definition 11 (Partial correlation) *In a variable set* $\mathbf{V}$*, the* partial correlation *between two random variables* $X, Y \in \mathbf{V}$ *given* $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$*, noted* $\rho_{XY \cdot \mathbf{Z}}$*, is the correlation of the residuals* R_X *and* R_Y *resulting from the least-squares linear regression of* X *on* $\mathbf{Z}$ *and of* Y *on* $\mathbf{Z}$*, respectively.*

TC was shown to be correct in the large sample limit (subject to the consistency of the statistical tests) in Pellet and Elisseeff (2007) under the Faithfulness and Causal Sufficiency assumptions. For the sake of completeness, we add the proof to Appendix A. The main points leading to the correctness of TC are the equivalence of a zero regression weight for some predictor Y while regressing X on all variables $\mathbf{V} \setminus X$ and a zero partial correlation $\rho_{XY \cdot \mathbf{V} \setminus \{X,Y\}}$, and the fact that this is zero if and only if $(X \perp\!\!\!\perp Y \mid \mathbf{V} \setminus \{X, Y\})$ holds in a Gaussian context (Baba et al., 2004). Then, our feature-selection step (Algorithm 6) gives the Markov blanket for each node, and the collider set search (Algorithm 2) then takes care of identifying the V-structures and removing the spouse links.

The other advantage of using linear regression and partial correlation is that it yields a fast algorithm. Actually, *all* regression weights and parameters needed for the feature-selection step of TC can be efficiently computed by inverting the sample correlation matrix $\mathbf{R} \in [-1, 1]^{d \times d}$. Building

graphs by inverting the correlation matrix is typically what is done with Gaussian Markov random fields, a special case of undirected graphical models (see, e.g., Talih, 2003).

The weight computation and the statistical significance tests are performed as follows. Let $\hat{b}_{ij}$ be the maximum likelihood estimator of the true regression weight b_{ij} of predictor X_j when X_i is the dependent variable, such that it solves the multiple regression equation for target X_i in the sense that it minimizes the sum of the squared residuals

$$SS_R = \sum_{k=1}^{n} \left(x_{ik} - \sum_{j=1, j\neq i}^{d} \hat{b}_{ij} x_{jk} \right)^2$$

where x_{ik} is the value of X_i for the kth sample. If we have the inverse correlation matrix $\mathbf{R}^{-1} = (r^{ij})$, the vector $\mathbf{b}$ at line 2 of Algorithm 7 can be found in linear time: $\hat{b}_{ij} = -r^{ij}/r^{ii}$ (Raveh, 1985). For instance, the list of weights to predict variable X_1 with all others is

$$\mathbf{b}_1 = (\hat{b}_{12}, \hat{b}_{13}, \cdots, \hat{b}_{1d}) = -(r^{12}, r^{13}, \cdots, r^{1d})/r^{11}. \tag{9}$$

The distribution of these weights is known (Judge et al., 1988):

$$\frac{\hat{b}_{ij} - b_{ij}}{\hat{\sigma}_{ij}} \sim t_{(n-(d-1))}, \tag{10}$$

where $\hat{\sigma}_{ij}$ is the standard error of the jth predictor for variable X_i; that is, that it follows a t distribution with a number of degrees of freedom $df =$ number of samples $-$ number of predictors $= n-(d-1)$. For our null hypothesis $H_0 : b_{ij} = 0$, we need $\hat{\sigma}_{ij}$ in addition to $\hat{b}_{ij}$ to compute the t-statistics $\hat{b}_{ij}/\hat{\sigma}_{ij}$. The estimate of the coefficient error $\hat{\sigma}_{ij}$ can be expressed as

$$\hat{\sigma}_{ij} = \hat{\sigma}_i \sqrt{\omega^{jj}/n},$$

where $\hat{\sigma}_i$ is an estimator of the standard error of the regression for target X_i, and ω^{jj} is the jth diagonal element of the inverse correlation matrix of the predictors (Judge et al., 1988, p. 243). (How to obtain the inverse correlation matrix of the predictors from the $\mathbf{R}^{-1}$ matrix in quadratic time is discussed in the next subsection.) The standard error $\hat{\sigma}_i$ can also be obtained in linear time from $\mathbf{R}^{-1}$ as follows.

Without loss of generality, we assume a zero mean and a unit standard deviation for all variables. Then $\sigma_i^2 = 1 - R_i^2$, where R_i^2 is the coefficient of determination of the regression for target X_i. This coefficient can be expressed as the scalar product of the $\mathbf{b}_i$ vector with the vector $\mathbf{r}_i$ of the pairwise correlation coefficients of the predictors with the target X_i (Raveh, 1985), which we read directly from the correlation matrix $\mathbf{R}$:

$$R_i^2 = \mathbf{b}_i^T \mathbf{r}_i.$$

An unbiased estimator $\hat{\sigma}_i$ for σ_i is then

$$\hat{\sigma}_i = \sqrt{\frac{n(1 - \mathbf{b}_i^T \mathbf{r}_i)}{n-d}}.$$

To sum up, we have a complexity of $O(nd^3)$ to build and invert the correlation matrix, and $O(d^3)$ to check for significance. This comes from having to obtain d times the inverse correlation matrix of $d-1$ predictors in $O(d^2)$, and then checking their significance in linear time. The overall complexity of TC, including the collider identification and the constraint-propagation steps, is then $O(nd^3+d^2 2^{\alpha})$.

The weaknesses of this approach are its infeasability when the correlation matrix $\mathbf{R}$ does not have full rank (including the special case $n<d$, that is, when there are fewer samples than variables), the low power of the statistical tests with small data sets, and multicollinearity in the predictors. The symptoms of the last two points are that the t-tests do not refute the null hypothesis of zero weight because (i) there is not enough data to support it, or (ii) multicollinearity makes the weights lower than they should be, such that it becomes harder to interpret them as depicting the independent contribution of each predictor. We try to deal with this problem in the next section with the TC_{bw} algorithm.

4.2.1 SIGNIFICANCE TESTS

Independently of low sample sizes or multicollinearity, the statistical tests on the weights of the linear regression equations are a delicate point in TC. The choice of the Type I error rate α needs investigating as it significantly influences the result of the algorithm.

In a network of d nodes, the feature-selection step performs $d(d-1)/2$ tests to determine the undirected skeleton. We will falsely reject the null hypothesis $b_{ij}=0$ about $m\cdot\alpha$ times on average, where $m<d(d-1)/2$ is the difference in the number of edges between the original DAG $\mathcal{G}_0$ and the complete graph. We will thus add on average $m\cdot\alpha$ wrong edges. We can set the significance level for the individual tests to be inversely proportional to $d(d-1)/2$ to avoid this problem (assuming a large m and thus rather sparse graphs), and check that it does not affect the Type II error rate too much, which we do now.

According to (10), the expression $(\hat{b}_{ij}-b_{ij})/\hat{\sigma}_{ij}$ follows a t distribution with $n-(d-1)$ degrees of freedom. If we call $\Psi(\cdot)$ the cumulative distribution function of a t distribution with $n-(d-1)$ degrees of freedom, we can write the Type II error rate β for each regression weight:

$$\beta_{ij}=\Psi(\Psi^{-1}(1-\alpha/2)-|b_{ij}|/\hat{\sigma}_{ij}).$$

The values for $\hat{\sigma}_{ij}$ can be computed from the inverse correlation matrix $\mathbf{R}^{-1}$ and thus depend on the particular data set being analyzed, but the true b_{ij} are unknown. What we could do in theory to optimize α is to minimize the average number of extraneous (T_e) and missing (T_m) links:

$$T=T_e+T_m=m\cdot\alpha+\sum_{(i,j)\in\mathbf{E}}\beta_{ij},$$

where m is the number of edges missing in the original DAG compared to a full graph, and $\mathbf{E}$ is the set of arcs in the original DAG, so that $m+|\mathbf{E}|=d(d-1)/2$. As m, $\mathbf{E}$ and b_{ij} are unknown, we can only find an upper bound for the number of missed links T_m, provided (i) we can estimate the graph sparseness to approximate m; (ii) we assume $|b_{ij}|\geq\delta$; and (iii) we choose $\mathbf{E}^{\star}$ such that it maximizes the sum in (11), with $|\mathbf{E}^{\star}|=d(d-1)/2-m$. Then we have:

$$T_m\leq\sum_{(i,j)\in\mathbf{E}^{\star}}\Psi(\Psi^{-1}(1-\alpha/2)-\delta/\hat{\sigma}_{ij}). \tag{11}$$

Although this bound was found too loose for practical use, we can model the Type I and Type II error rate as a function of α for artificial problems whose sparseness and regression weights are known. This is shown in Figure 3 for a specific instance of an Alarm data set (see Section 5 for details on this network) with two different sample sizes, $n = 50$ (left) and $n = 250$ (right). We did not use this information to tune α in the experiments, as it cannot be obtained without prior knowledge, but the curves showed that an α inversely proportional to $d(d-1)/2$ has the same order of magnitude as the optimal α on the data sets we analyzed.

What we also see is that the Type I error curve rapidly goes up, whereas the Type II error curve is upper-bounded by the total number of links in the original graph. In terms of pure number of errors, setting a low α will thus be more beneficial than setting a higher α to get a low β. It is worth discussing, however, depending on the particular problem to solve, which is more desirable: missing causal links or getting extra causal links. In terms of Bayesian networks, getting too few links prevents the model from being able to reconstruct the full joint probability distribution, because we lose the I-map property; whereas getting too many links implies having to estimate more parameters from the same data and thus complexifies a subsequent parameter learning task.

4.3 The TC_{bw} Algorithm

Despite correctness of TC, with a low number of samples n it fails to have enough evidence for rejecting the null hypothesis of zero regression weight, and thus misses links (see detailed results in Section 5), even for a high α. We now try to address this particular issue by successively eliminating the most insignificant predictors and reevaluating the remaining ones. This is actually a backward stepwise-regression method. Pseudocode for this heuristics is listed in Algorithm 7.

Algorithm 7 The Total Conditioning Backward Feature-Selection Step

```
1:  procedure TCBWFEATURESELECTION
      Input:   X :  the target variable to perform feature selection for
               D :  n × d data set with n d-dimensional data points
      Output:  S :  the set of selected variables
2:    P ← V \ X    /* all predictors */
3:    S ← ∅    /* significant predictors */
4:    while P ≠ ∅ and P ≠ S do
5:      b ← weights of P in the problem of regressing X on P
6:      S ← S ∪ {predictors whose b weight is significant}
7:      P ← P \ {the p less significant predictors}
8:    end while
9:    return S
10: end procedure
```

Intuitively, the problem to solve is that the regression weights cannot be high enough for significance with small sample sizes. By removing the most insignificant predictors and thus the most likely to be actually zero, we scale down the regression problem and increase the power of the tests. How many insignificant predictors to remove can be discussed; in our implementation, we compared $p = 1$ to $p = (\text{number of predictors})/2$ and found that the latter yielded results that were just as good with an important speed gain.

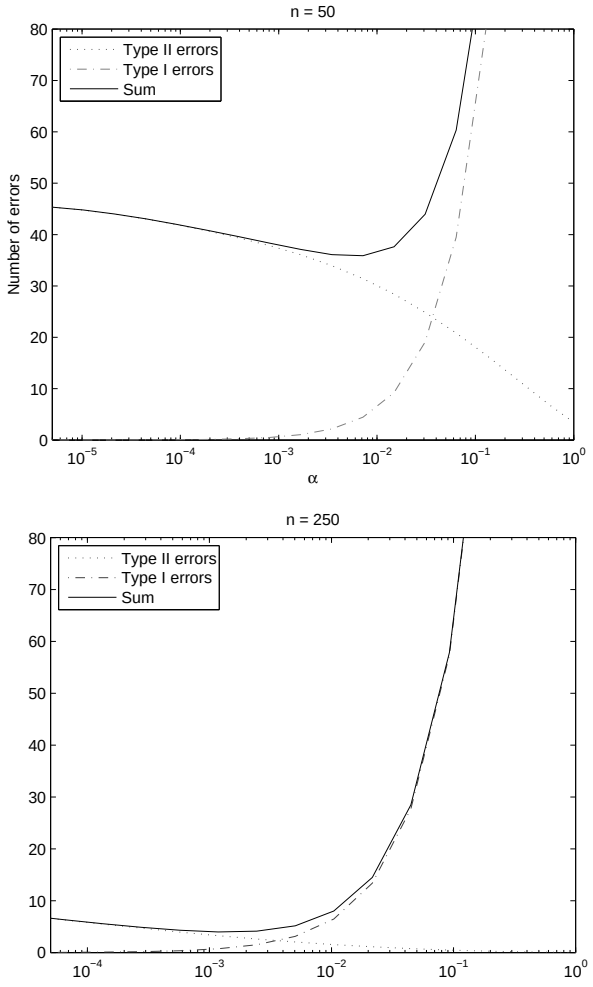


Figure 3: Expected Type I and II errors as a function of α

This stepwise regression raises some issues; notably, Tibshirani (1994) argues that the repeated tests on non-changing data are biased and that the remaining **b** coefficients are too large. We thus expect TC_{bw} to be biased and to include more false positives than TC. Ideally, one would need a criterion to predict when the additional false positives would outweigh the benefits of reducing the false negatives. Whether such a criterion, which would allow us to know a priori whether TC or TC_{bw} should be used, can be found, is an open question.

Solving a standard multiple regression problem with d predictors traditionally has complexity $O(nd^3)$. Naïvely solving $d-1$ regression problems d times in the case $p=1$ would have a complexity of $O(nd^5)$. But we can avoid reinverting matrices in the inner loop of the stepwise regression thanks to the following result.

Let $\Sigma = \mathbf{X}^T\mathbf{X}$ be n times the correlation matrix $\mathbf{R}$, where $\mathbf{X}$ is the $n \times d$ matrix representing a data set where all variables have zero mean and unit standard deviation. Then we can use Σ^{-1} to linearly find the weights of the regression problems and their standard error, which are needed for

the t-tests. Suppose we find that variable X_1 is the weakest predictor, and want to reevaluate the weights of the other predictors at line 5 of TC_{bw}. Let $\mathbf{X}_{\setminus i}$ be the data set where variable X_i has been removed. Then we need the matrix Ω^{-1} to solve the new problem, where $\Omega = \mathbf{X}_{\setminus 1}^T \mathbf{X}_{\setminus 1}$. As a special case of Strassen's blockwise matrix inversion formula, we have:

$$\Sigma = \begin{bmatrix} \sigma_{11} & \mathbf{c}^T \\ \mathbf{c} & \Omega \end{bmatrix}$$

$$\Longrightarrow \quad \Sigma^{-1} = \begin{bmatrix} \dfrac{1}{\sigma_{11} - \mathbf{c}^T \Omega^{-1} \mathbf{c}} & -\dfrac{\mathbf{c}^T \Omega^{-1}}{\sigma_{11} - \mathbf{c}^T \Omega^{-1} \mathbf{c}} \\ -\dfrac{\Omega^{-1} \mathbf{c}}{\sigma_{11} - \mathbf{c}^T \Omega^{-1} \mathbf{c}} & \Omega^{-1} + \dfrac{\Omega^{-1} \mathbf{c} \mathbf{c}^T \Omega^{-1}}{\sigma_{11} - \mathbf{c}^T \Omega^{-1} \mathbf{c}} \end{bmatrix}.$$

Let $\sigma^{ij} = (\Sigma^{-1})_{ij}$ and $\mathbf{b} = \Omega^{-1}\mathbf{c}$. Then $\mathbf{b}$ are the weights of the regression of X_1 on $X_2, \cdots, X_d$ and can be computed without knowing Ω^{-1} (Raveh, 1985), see (9). We have:

$$\sigma^{11} = 1/(\sigma_{11} - \mathbf{c}^T \mathbf{b})$$

and, $(\Sigma^{-1})_{\setminus 1}$ being the matrix Σ^{-1} where the first row and column have been removed,

$$(\Sigma^{-1})_{\setminus 1} = \Omega^{-1} + \mathbf{b}\mathbf{b}^T / (\sigma_{11} - \mathbf{c}^T \mathbf{b}).$$

We can thus compute Ω^{-1} given Σ^{-1} with complexity $O(d^2)$ as follows:

$$\Omega^{-1} = (\Sigma^{-1})_{\setminus 1} - \sigma^{11} \mathbf{b}\mathbf{b}^T. \tag{12}$$

This trick is also used in TC to find the inverse correlation matrix of the predictors from the inverse correlation matrix of the whole variable set.

Equation (12) is implemented in TC_{bw} such that we never need to invert another matrix again once Σ^{-1} has been obtained, and leads to a complexity of $O(d^2)$ for stepwise elimination of a predictor. In the most computationally expensive case $p = 1$, this elimination of row and column of the inverse matrix is repeated at most $d - 2$ for each variable, yielding a complexity of $O(nd^4)$ for the whole feature-selection step for all variables. The overall complexity of TC_{bw} is then $O(nd^4 + d^2 2^{\alpha})$. We are only adding one complexity degree in d with respect to TC with the additional stepwise regression.

5. Experimental Results

In this section, we report on experiments and results on two points separately. First, we test our procedure described in Algorithm 2 to recover the local structure with the collider set search given all Markov blankets, and compare it to the relevant steps of the GS algorithm, which are listed in Algorithm 1, with 5 different network topologies. For the sake of comparison, we also run the reference PC algorithm (Spirtes et al., 2001), initialized with the moral graph instead of the fully connected graph.

Second, we conduct experiments to investigate how the whole structure-learning algorithms behave. We first use the RFE-based approach. We then systematically compare TC, TC_{bw} and several reference algorithms, varying the data set size and the network size. Note that results for some algorithms may be sparser due to their prohibitive run times on some data sets.

5.1 Experimental Setup

In order to test the accuracy of the various algorithms, we chose to sample data from the following known networks, from the Bayes net repository (Elidan, 2001):

- Alarm network (Beinlich et al., 1989). This network has become a de facto standard benchmark for structure-learning algorithms: it contains 37 nodes, 46 arcs, 4 undirected in the PDAG of the equivalence class. It was originally designed to help interpret monitoring data to alert anesthesiologists to various situations in the operating room. It is depicted in Figure 4.

- Insurance (Binder et al., 1997), 27 nodes, 52 arcs, 18 undirected in its PDAG. It was designed to evaluate car insurance risks. This network has fewer nodes than Alarm but is denser, see Figure 5.

- Hailfinder (Abramson et al., 1996), 56 nodes, 66 arcs, 17 undirected in its PDAG. It is a normative system that forecasts severe summer hail in northeastern Colorado. See Figure 6.

- Carpo,[10] 61 nodes, 74 arcs, 24 undirected in its PDAG. It is meant to help diagnose the carpal tunnel syndrome. The version we used has three disconnected subgraphs, one of which is a single variable, and a relatively flat causal structure, as can be seen in Figure 7.

- A subset of Diabetes (Andreassen et al., 1991) with 104 nodes, 149 arcs, 8 undirected in its PDAG, which was designed as a preliminary model for insulin dose adjustment. This subset is made of 6 repeating patterns (there are 24 in the original network) of 17 nodes, plus 2 external nodes linked to every pattern. The first two of these patterns are shown in Figure 8.

We performed three series of experiments.

1. We compared our algorithm resolving the Markov blanket to the relevant steps of the Grow-Shrink algorithm, as described in Section 3.2, and to PC;

2. We tested the RFE-based approach and compared it to PC;

3. Finally, we compared TC and TC_{bw} to three reference algorithms and examine their accuracy, run time, and number of tests while varying the network structure, the network size, and the sample size.

The chosen reference algorithms are:

1. The PC algorithm. PC is, like TC and TC_{bw}, exponential in the worst case, when graphs are not sparse enough: we discuss which structural elements make PC or TC exhibit the exponential behavior;

2. The full Grow-Shrink algorithm, as described in Margaritis and Thrun (1999);

3. A state-of-the-art Bayesian structure-learning algorithm that works with continuous data sets, the Bach-Jordan scoring algorithm (Bach and Jordan, 2003), coupled with a greedy search in the space of DAGs. Note that Bayesian structure-learning algorithms are often score-based and return fully oriented DAGs. Maximizing the chosen score function might not minimize the number of structural errors as we report in these results.

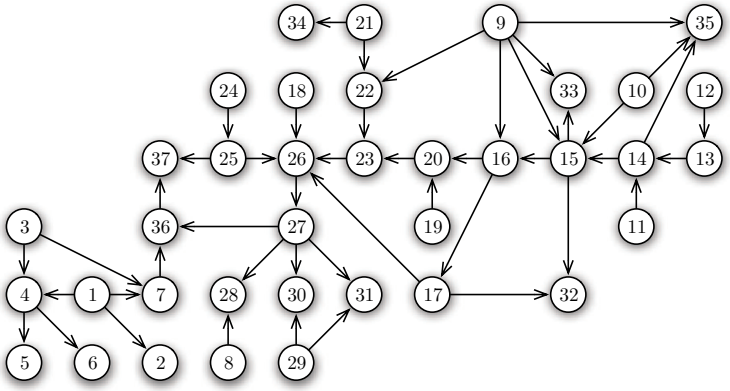


Figure 4: The Alarm network

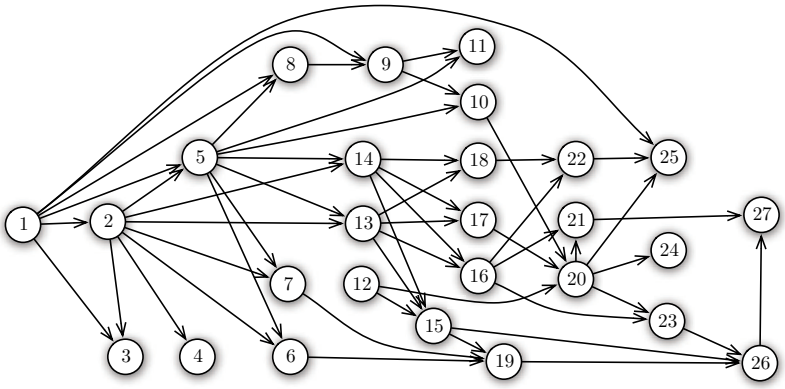


Figure 5: The Insurance network

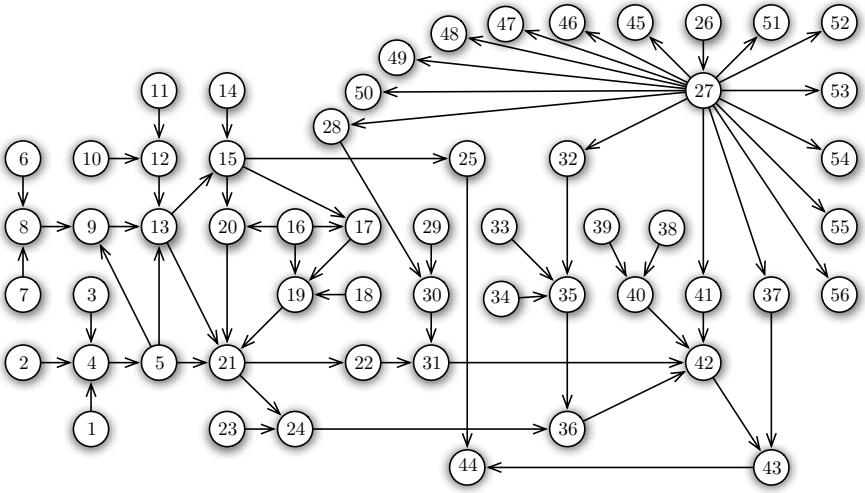


Figure 6: The Hailfinder network

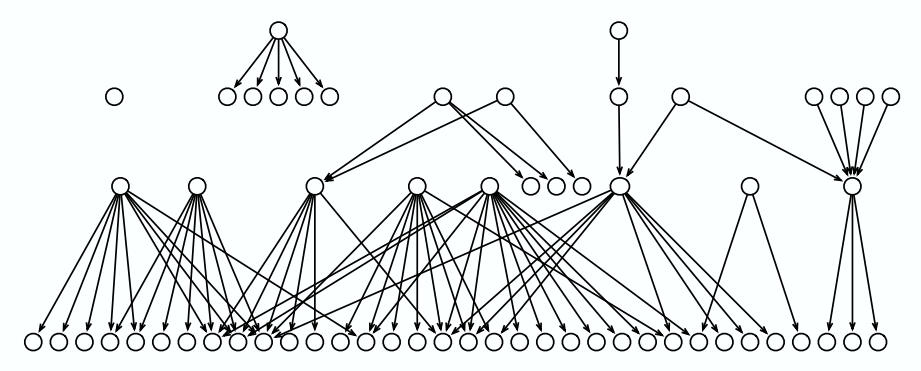


Figure 7: The Carpo network

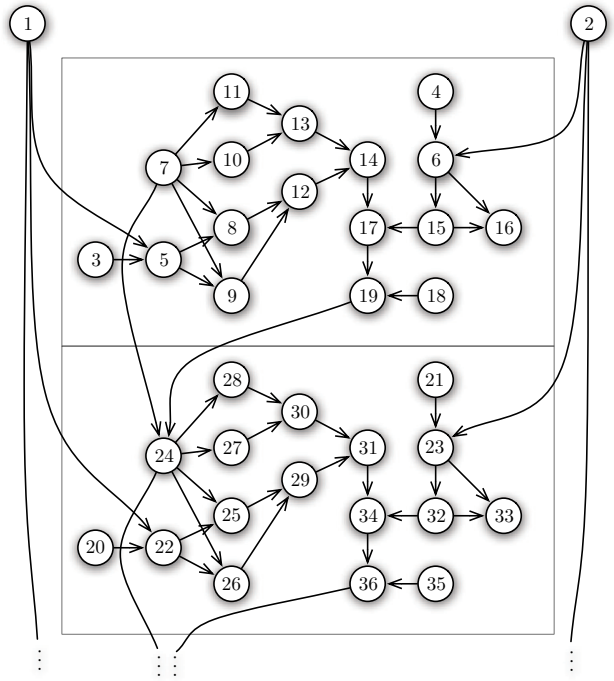


Figure 8: Two of the six patterns of the Diabetes network

For all simulation experiments, we generated the data sets by using the 5 graphs as a structure for a linear structural equations model: the parentless variables were sampled as Gaussians with zero mean and unit standard deviation; the other variables were defined as a linear combination of their parents with coefficients randomly distributed uniformly between 0.2 and 1, similarly to what was done in Scheines et al. (1995) for the evaluation of PC. The disturbance terms were also normally distributed. We compared the number of tests, the size of the conditioning sets, and the

10. Created by Alex Dagum with contributions from Mark Peot, as indicated on its page at the Bayes net repository. No corresponding publication was found.

structural errors in case of runs with artificial data. A structural error is an arc addition, deletion, or reversal with respect to the original graph.

We used the implementation of PC proposed by Leray and François (2004) in the BNT Structure Learning Matlab package. The implementation of TC and TC_{bw} was also done in Matlab. The statistical tests were done using Fisher's *z*-transform of the partial correlation, unless otherwise stated. For PC and GS, we chose the default value of $\alpha = 0.05$; we note though that the optimal value of α is problem dependent and that especially with low sample sizes, hand tuning α can return better results than those listed here. For both TC and TC_{bw}, we set $\alpha = 2/(d(d-1))$, according to the discussion at the end of Section 4.2.

5.2 Local Structure Recovery with Markov Blanket Information

In this series of experiments, we compare RESOLVEMARKOVBLANKETS_COLLIDERSETS (CS) to RESOLVEMARKOVBLANKETS_GROWSHRINK and to a modified version of PC, where the graph being built is initialized with the moral graph (instead of the full graph in the original version of PC). This represents exactly the Markov blanket information available to the two other algorithms and allows a direct comparison. Note that we observe the PDAG that PC obtains *before* the constraint-propagation step building the maximally oriented PDAG, such that, in all three tested algorithms, we only expect the V-structures to be oriented.

We tested the three algorithms on each network using two methods to check for conditional independence: first, using a *d*-separation oracle with the original graph (which is equivalent to a perfect test); and second, using Fisher's *z*-transform of the sample partial correlation coefficient as computed on artificial data, with significance $\alpha = 0.05$. Using the oracle always yields correct graphs.

Table 1 shows the results of these experiments. We first list the results obtained when using a *d*-separation oracle to decide upon conditional independence. For GS, we ran two versions of Algorithm 1: one, which we name GS(1), where the subset searches at lines 5 and 13 proceed with decreasing sizes of the chosen subset $\mathbf{S}$, and another, GS(2), with increasing subset sizes. GS(1) usually leads to fewer tests, but with larger conditioning sets. The order of the subset searches for our method (lines 7 and 15 in Algorithm 2) was fixed to decreasing subset sizes, as this always led to fewer tests *and* smaller conditioning sets.

The results for the modified PC algorithm are only shown for the sake of comparison: PC is a general-purpose algorithm which is not specialized in such local structure recognition given the Markov blankets. What the comparison shows, however, is that, whenever this Markov blanket information is available or cheap to obtain, there are much more efficient approaches.

GS(1) and GS(2) are close to one another in all scores, and outperform PC (by several orders of magnitude) in the number of tests and (significantly) in average and maximum size of the conditioning sets (except, artificially, for the results marked with a star), because it uses the Markov blanket information better. Our approach, however, is one order of magnitude better than GS(1) and GS(2) in terms of number of tests, while still using smaller average and maximum conditioning set sizes in all tested networks. Especially striking are the results on the Carpo network: this is an example where CS saves a lot of time ignoring the numerous links not part of triangles, whereas GS(1) and GS(2) also checks those, with the often large Markov blankets (Figure 7).

We then performed the same experiments, but using the statistical tests on data sampled from the networks as described in the previous sections. We used a fixed sample size $n = 500$ and averaged

Algorithm	Alarm	Insurance	Hailfinder	Carpo	Diabetes
modified PC					
# tests	11331	773572	19543985*	2025250*	93134*
avg. $\lvert\mathbf{Z}\rvert$	4.36	7.65	5.75*	5.47*	4.64*
max $\lvert\mathbf{Z}\rvert$	10	16	6*	6*	6*
GS(1)					
# tests	1485	6435	2809	209342	5414
avg. $\lvert\mathbf{Z}\rvert$	2.62	3.63	2.66	7.46	2.73
max $\lvert\mathbf{Z}\rvert$	8	11	7	15	10
GS(2)					
# tests	1472	7180	2979	200621	6197
avg. $\lvert\mathbf{Z}\rvert$	2.20	3.05	2.31	7.39	2.39
max $\lvert\mathbf{Z}\rvert$	7	8	7	15	8
CS					
# tests	214	1288	593	294	943
avg. $\lvert\mathbf{Z}\rvert$	1.80	2.69	2.30	1.79	2.13
max $\lvert\mathbf{Z}\rvert$	5	6	6	8	7

Table 1: Number of tests and size of the conditioning sets (noted $\lvert\mathbf{Z}\rvert$) as performed by various algorithms to recover the local network structure of the networks given perfect Markov blanket information. The star (*) notes PC results where the maximum size of the conditioning set has been set to 6 to avoid prohibitive run times.

over 9 different samplings for each network. We only compared PC, GS(1) and CS on this series of experiments, preferring GS(1) to GS(2) because of the lower number of tests it usually performs. The exhaustive results are listed in Table 2 for the sake of completeness, and the sum of the structural errors is also shown in Figure 9 for easier visualization.

First, we see that we get similar results as in Table 1 as far as the number of tests and size of the conditioning sets are concerned: CS is faster and consistently performs fewer tests with smaller conditioning sets, which leads to an increased power of the tests. However, that is sometimes balanced out by the fact that CS relies on a single series of tests both to remove spouse links and to orient (possibly multiple) V-structures at the same time, thus leading to a greater penalty if the outcome of a test is wrong with respect to the initial graph.

We see that GS(1) and PC can beat CS on certain arc scores; PC, in particular, is good at avoiding arc orientation mistakes in these experiments. GS(1), which checks not only triangle links but all links to try to orient them, makes more orientation mistakes, especially on the Carpo network. PC tends to miss a few more arcs than CS, which in turn misses a few more than GS(1). But in total, CS beats GS(1) significantly on Insurance, Hailfinder, and Carpo, while performing slightly better on Alarm and being slightly outperformed on Diabetes. Based on these results, we will now use our collider set search as the method of choice to break up the Markov blankets for the next series of experiments.

Algorithm	Alarm	Insurance	Hailfinder	Carpo	Diabetes
mod. PC					
# tests	2850 ± 285	13461 ± 3247	$9681105^{\dagger}$	412791 ± 104080	57153 ± 9910
avg. $\lvert\mathbf{Z}\rvert$	2.97 ± 0.17	3.50 ± 0.33	$5.54^{\dagger}$	5.17 ± 0.15	4.37 ± 0.14
max $\lvert\mathbf{Z}\rvert$	6	6	$6^{\dagger}$	6	6
arcs:					
missing	5.44 ± 0.53	9.56 ± 1.01	$6^{\dagger}$	14.22 ± 1.64	9.56 ± 1.88
extra	0.33 ± 0.5	0.11 ± 0.33	$\mathbf{0}^{\dagger}$	$\mathbf{0.22} \pm 0.44$	$\mathbf{1.11} \pm 0.60$
reversed	**0**	$\mathbf{0.22} \pm 0.67$	$1^{\dagger}$	0.11 ± 0.22	$\mathbf{2.00} \pm 1.39$
TOTAL	5.78 ± 0.72	9.89 ± 1.47	$7^{\dagger}$	14.56 ± 1.86	12.67 ± 2.69
GS(1)					
# tests	1304 ± 60	4544 ± 195	2415 ± 63	129265 ± 17033	5239 ± 46
avg. $\lvert\mathbf{Z}\rvert$	2.66 ± 0.10	3.66 ± 0.04	2.62 ± 0.02	7.49 ± 0.08	2.76 ± 0.01
max $\lvert\mathbf{Z}\rvert$	8	11	7.89 ± 0.33	15	10
arcs:					
missing	$\mathbf{1.56} \pm 0.53$	$\mathbf{5.44} \pm 0.53$	$\mathbf{3.11} \pm 0.33$	**0**	6.11 ± 0.78
extra	0.56 ± 0.73	0.33 ± 0.71	1 ± 0.71	$\mathbf{0.22} \pm 0.44$	2.78 ± 1.64
reversed	1.11 ± 1.05	3.67 ± 2.12	8 ± 2.29	16.78 ± 2.49	2.67 ± 2.00
TOTAL	3.22 ± 1.81	9.44 ± 2.39	12.11 ± 2.74	17 ± 2.62	$\mathbf{11.55} \pm 3.03$
CS					
# tests	173 ± 3	782 ± 19	507 ± 18	308 ± 14.39	907 ± 4
avg. $\lvert\mathbf{Z}\rvert$	1.55 ± 0.03	2.36 ± 0.02	2.08 ± 0.03	1.90 ± 0.12	2.17 ± 0.01
max $\lvert\mathbf{Z}\rvert$	5	6	5	8	7
arcs:					
missing	$\mathbf{1.56} \pm 0.73$	6.33 ± 0.5	3.44 ± 0.52	**0**	$\mathbf{5.11} \pm 1.17$
extra	$\mathbf{0.44} \pm 0.53$	$\mathbf{0.22} \pm 0.44$	0.67 ± 0.70	0.33 ± 0.50	1.44 ± 1.51
reversed	0.11 ± 0.33	0.33 ± 0.5	$\mathbf{0.11} \pm 0.33$	**0**	7.11 ± 1.05
TOTAL	$\mathbf{2.11} \pm 0.96$	$\mathbf{6.89} \pm 1.03$	$\mathbf{4.22} \pm 1.27$	$\mathbf{0.33} \pm 0.50$	13.66 ± 2.20

Table 2: Number of tests, size of the conditioning sets (noted $\lvert\mathbf{Z}\rvert$), and structural errors as returned by GS(1) and CS to recover the local network structure of the networks given perfect Markov blanket information. Results are given is the form "mean $\pm$ standard deviation over the 9 data sets." The best performer for each type of structural error has been highlighted in bold. All runs of PC were done with a forced maximum size of the conditioning set of 6. The dagger (†) notes PC results from a single data set instead of 9 because of the long completion times. Represented graphically in Figure 9.

5.3 RFE-Based Approach

In this series of experiments, we tested our RFE-based approach on the Alarm network with sample sizes $n = 100$, 200, 300, 400 and 500. Table 3 lists the results and shows the number of errors as measured at different stages of the algorithm:

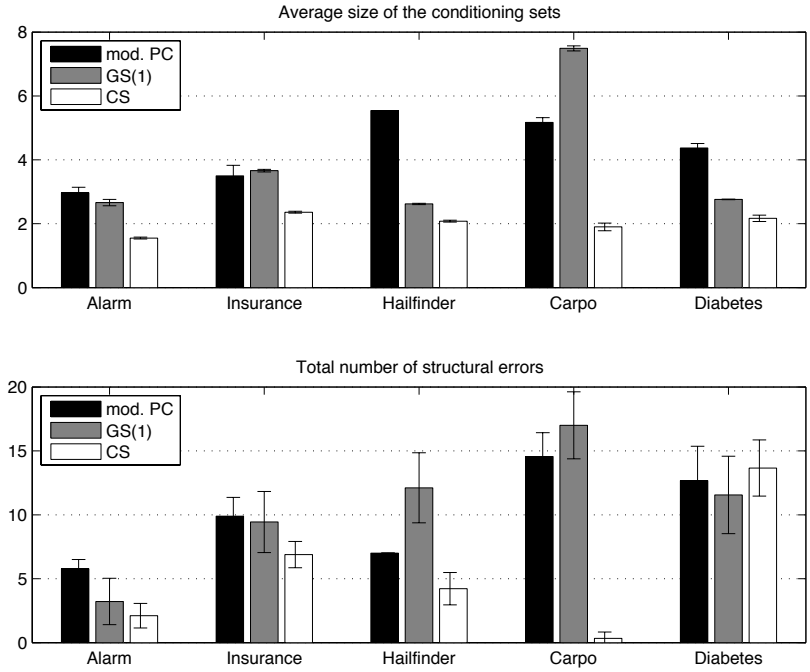


Figure 9: Average size of the conditioning sets and total number of errors for the three local structure discovery algorithms on various networks. Graphical representation corresponding to the results in Table 2.

1. Right after the Markov blanket identification, without adjustment. This compares the true Markov blanket of each variable with the identified Markov blanket as returned by Algorithm 5;

2. After building the moral graph. This notably excludes variables from Markov blankets if they do not satisfy the symmetry condition (2) due to the symmetry check performed at line 5 in the generic approach described in Algorithm 3;

3a. After removal of the spouse links using the Recursive Median (RM) algorithm (Margaritis, 2005) to check for conditional independence in the continuous domain;

3b. Alternatively, after removal of the spouse links using a test on Fisher's *z*-transform of partial correlation;

4a. After removal of the spouse links using RM *and* after maximal orientation. This is actually the result that can be compared to other full structure-learning algorithms;

4b. After removal of the spouse links using partial correlation tests *and* after maximal orientation;

5. Finally, we show how PC performs on the same instance for comparison.

Note that the RM test is a Bayesian distribution-free conditional-independence test. In this case, where we use multivariate Gaussian distributed data, we do not expect it to perform better

than the specialized z-test. We nevertheless include it in this series of experiments for two reasons. First, it allows the collider set search to be also distribution-free, in the sense that if "distribution-free feature selection" can be performed efficiently and consistently in the first phase, applying a subsequent collider set search does not make more assumptions on the distribution. Second, it allows to evaluate the cost of using a distribution-free algorithm on Gaussian data.

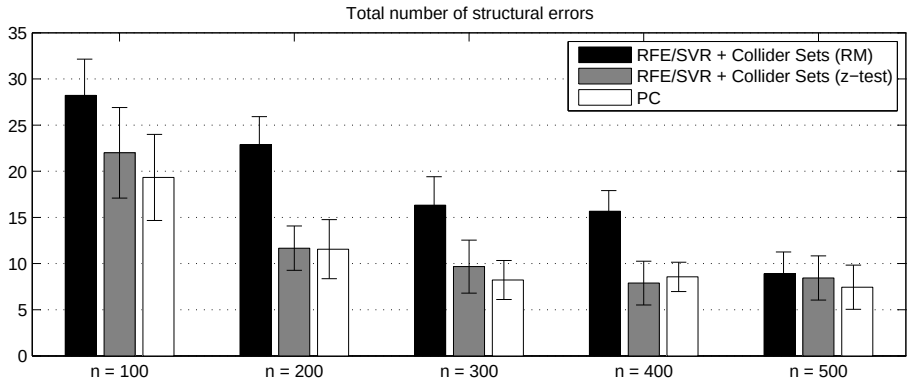


Figure 10: Total number of errors for the CPDAGs returned by the three global structure discovery algorithms on the Alarm network with various sample sizes. Graphical representation corresponding to the results in Table 3.

Detailed results are in Table 3 and the total number of structural errors is shown graphically in Figure 10. What we can read from the results is that, generally, the selected Markov blankets contain all variables from the true Markov blanket plus one or two additional variables. Starting at $n = 300$, on average, less than two variables were missed. Many spurious variable are selected, however, even for the larger data sets. This confirms the expectation the RFE approach also selects weakly relevant features: on average, the Markov blankets in the Alarm network have a size of 3.5, and on average 5.5 variables are selected per variable.

The symmetry check requiring Y to be part of $\mathbf{Mb}(X)$ and X to be part of $\mathbf{Mb}(Y)$ to add a link between X and Y fulfills its purpose, as even in the case $n = 200$ where on average about 73 variables enter wrong Markov blankets, only 4 extra links are added in the moral graph. As a side note, we thus argue that a global analysis can be beneficial to achieve better results on local tasks: we see here that determining via RFE the Markov blanket of a single variable is too inclusive, but that validating the selected variables globally, for instance with our Markov blanket symmetry check, allows to significantly reduce the number of false positives.

After the collider set search, the number of missing and extra arcs can both either increase or decrease. If the number of missing links increases, it is because the collider set search found d-separation too often while variables were actually dependent. If it decreases, it means that the missing arcs in the moral graph were spouse links, as their absence is not penalized in the PDAG any more. If the number of extra arcs increases, then the collider set search failed to identify spouse links; if it decreases, then the collider set search also removed through appropriate conditioning links that were not spouse links (which in turn possibly led to wrong orientations). Also, determining which part of the algorithm is responsible for a missed, extra, or reversed edge in a PDAG or CPDAG is not evident. As the feature-selection step is not alone responsible for the extra or missing links,

Stage	$n = 100$	$n = 200$	$n = 300$	$n = 400$	$n = 500$
1. Mb$(\cdot)$ ident.					
missing variables	17.33 ± 3.33	3.33 ± 1.48	0.78 ± 1.11	0.78 ± 1.48	0.33 ± 1.48
extra variables	80.66 ± 15.17	72.89 ± 9.25	74.78 ± 13.69	72.56 ± 9.99	73.33 ± 9.99
2. Moral graph					
missing arcs	23.44 ± 3.54	7.89 ± 3.48	4.33 ± 3.91	4.56 ± 3.47	4.33 ± 3.87
extra arcs	4.67 ± 2.24	4.11 ± 1.69	3.78 ± 1.20	3.11 ± 1.62	3.89 ± 2.20
TOTAL	28.11 ± 3.82	12.00 ± 2.55	8.11 ± 4.01	7.67 ± 4.06	8.22 ± 4.38
3a. PDAG/RM					
missing arcs	17.44 ± 2.35	10.00 ± 1.80	6.89 ± 2.67	6.33 ± 2.60	4.56 ± 1.51
extra arcs	4.78 ± 2.17	4.22 ± 1.56	3.33 ± 1.12	3.22 ± 1.48	3.44 ± 2.01
reversed arcs	1.22 ± 1.20	2.11 ± 1.27	3.11 ± 0.78	1.78 ± 0.97	0.67 ± 0.60
TOTAL	23.44 ± 1.67	16.33 ± 3.12	13.33 ± 2.65	11.33 ± 2.78	8.67 ± 2.37
3b. PDAG/z-t.					
missing arcs	11.89 ± 2.32	3.33 ± 1.32	2.67 ± 1.94	2.11 ± 1.17	2.56 ± 1.51
extra arcs	5.22 ± 2.22	4.00 ± 1.58	3.22 ± 1.20	2.78 ± 1.30	3.44 ± 2.01
reversed arcs	0.33 ± 0.50	0.56 ± 0.73	1.22 ± 0.67	0.78 ± 1.09	0.44 ± 1.13
TOTAL	17.44 ± 3.32	7.89 ± 2.15	7.11 ± 2.98	5.67 ± 2.12	6.44 ± 2.40
4a. CPDAG/RM					
missing arcs	17.44 ± 2.35	10.00 ± 1.80	6.89 ± 2.67	6.33 ± 2.60	4.56 ± 1.51
extra arcs	4.78 ± 2.17	4.22 ± 1.56	3.33 ± 1.12	3.22 ± 1.48	3.44 ± 2.01
reversed arcs	6.00 ± 3.87	8.67 ± 2.12	6.11 ± 2.32	6.11 ± 3.59	0.89 ± 0.60
TOTAL	28.22 ± 3.93	22.89 ± 3.02	16.33 ± 3.08	15.67 ± 2.24	8.89 ± 2.37
4b. CPDAG/z-t.					
missing arcs	11.89 ± 2.32	3.33 ± 1.32	2.67 ± 1.94	2.11 ± 1.17	2.56 ± 1.51
extra arcs	5.22 ± 2.22	4.00 ± 1.58	3.22 ± 1.20	2.78 ± 1.30	3.44 ± 2.01
reversed arcs	4.89 ± 3.33	4.33 ± 1.73	3.78 ± 1.09	3.00 ± 1.80	2.44 ± 1.13
TOTAL	22.00 ± 4.90	$\mathbf{11.67} \pm 2.40$	9.67 ± 2.87	$\mathbf{7.89} \pm 2.37$	8.44 ± 2.40
5. CPDAG/PC					
missing arcs	12.11 ± 2.52	7.44 ± 1.42	4.22 ± 0.97	5.67 ± 1.12	4.78 ± 0.83
extra arcs	4.56 ± 2.19	2.67 ± 1.87	2.78 ± 1.48	2.11 ± 0.93	2.00 ± 1.66
reversed arcs	2.67 ± 1.80	1.44 ± 1.51	1.22 ± 1.09	0.78 ± 1.09	0.67 ± 0.87
TOTAL	$\mathbf{19.33} \pm 4.66$	$\mathbf{11.56} \pm 3.2$	$\mathbf{8.22} \pm 2.11$	8.56 ± 1.59	$\mathbf{7.44} \pm 2.40$

Table 3: Structural errors at various stages of the RFE-based approach, showing the missing, extra and reversed arcs with respect to the original graph. For Step 1, identification of the Markov blanket, the figures are averages over the 37 variables; that is, the count of the extra or missing variables per Markov blanket, and thus not directly comparable to the other steps. The sums of the errors for the CPDAGs are represented in Figure 10.

the collider set search is not responsible for all orientation mistakes. In the collider set search, if a wrong spouse link is removed, it is because a wrong V-structure has been identified, so that the absence of an arc will be linked to the wrong orientation of the falsely recognized V-structure. It is also possible to construct cases where missing a variable in the feature-selection step will lead not only to a missing arc, but also to the detection of a spurious V-structure, even if all subsequent tests are perfect.

For the PDAGs obtained using z-tests, the number of missing arcs always decreases with respect to the moral graph, and so does the number of extra links for $n \geq 200$. We find that the more general RM test seems to return independence too often, as for $n \geq 200$ more links are missing in the PDAG than in the moral graph. (On highly nonlinear data, we would, however, expect RM to perform better than a z-test, which assumes Gaussianity.)

The CPDAGs do not have a number of adjacency errors different from their PDAGs; this step can only add directionality errors. We have nevertheless copied the results in order to improve the readability and to make the comparison with PC easier. Although the RFE approach can outperform PC in adjacency errors, PC still consistently makes fewer directionality errors. We remark, however, that the overall performance of RFE/SVR with z-tests is very comparable to that of PC, as also shown in Figure 10, which empirically justifies the intuition behind this approach.

5.4 TC and TC_{bw} vs. Competitors

For this series of experiments, we performed more systematic testing of TC, TC_{bw}, PC, the full GS and the Bach-Jordan method on data sets sampled from Alarm, Insurance, Hailfinder, Carpo, and Diabetes, varying the sample size. The Bach-Jordan method consists of a scoring function based on Mercer kernels coupled with a greedy search in the space of DAGs and was designed to learn Bayesian networks. It does not guarantee that the formal semantics of a causal graph are respected in the large-sample limit, but has been included in the experiments for the sake of comparison. Other possible competitors like SCA (Friedman et al., 2000) or AlgorithmMB (Peña et al., 2005) were inapplicable because generalizing them to handle continuous variables require techniques that are too computationally expensive, notably because of score-based subroutines that are hard to generalize.

The structural errors, like before, are missing, extra, and reversed arcs in the returned CPDAG with respect to the generating graph. For the Bach-Jordan method, similarly to what was done in Fu (2005), we converted the returned DAG to its essential graph first before checking for structural errors to avoid penalizing statistically equivalent structures. For all experiments, we also compare the run times and the number of tests performed by TC, TC_{bw}, GS, and PC.

Specific to the Bach-Jordan method is the issue of choosing the appropriate kernel parameters; that is, in our case, the σ width in the Gaussian kernel. Bach and Jordan (2003) claim that the algorithm is in general robust to the choice of σ. We have found, however, that for varying sample sizes, the number of structural errors is quite sensitive to σ. As the authors do not propose a heuristics to set it, we systematically tested the algorithm with $\sigma = 2$, 1, 0.5, and 0.3 for each run, and chose the outcome with the smallest sum of structural errors. In general, smaller data sets preferred $\sigma = 2$, while the larger ones preferred a smaller σ. The change of σ is not directly visible in the following plots of the errors, but it often leads to "zigzags" in the Bach-Jordan curves. This is due to the fact that we only tested a fixed number of values for σ and did not perform a full optimization of this parameter for each run. The results shown are thus not the best results obtainable with this method.

5.4.1 ALARM

The figures on p. 1330 show the structural errors, run times and number of statistical tests against the number of samples for Alarm. For each sample size, 5 data sets were drawn from the model; the error bars picture the standard deviation over these 5 runs.

The numbers of extra and missing links seem to clearly decrease on average for all algorithms with an increasing number of samples, except for Bach-Jordan, which sometimes has the tendency to add more links when more data points are available. Note that Bach-Jordan's σ changes between the last two runs, explaining the abrupt change in the extra arcs. The number of reversed arcs seems to less satisfactory, in particular for TC. The explanation is that TC misses many arcs with low sample sizes, and thus does not actually get the opportunity to make many directionality errors for these cases. TC_{bw} exhibits a related behavior, although much less stronger. We also see that Bach-Jordan makes the most directionality errors (this is actually valid for all networks). GS reaches repeatedly a zero extra arc score for $n > 1000$, although it misses some more than the others.

Starting at about 200 samples, TC equals or outperforms PC, GS and Bach-Jordan. TC_{bw} beats both TC and PC, and the converging curves of TC and TC_{bw} show that the stepwise regression becomes unnecessary with about 400 samples. On average, TC was about 20 times faster than the implementation of PC we used, although the factor tended to decrease with larger sample sizes. TC_{bw} was naturally slower than TC, although only marginally compared to the speed difference with PC.

Overall, the constraint-based methods seem to perform approximately equally well for $n > 400$, and TC_{bw} and GS perform slightly better than PC for low sample sizes. Note that for low sample sizes, TC is always outperformed by TC_{bw}, PC, and GS, but is often the one to perform best when the sample size gets larger. The graphs in Figure 12 show that TC and TC_{bw} are fastest, although GS performed fewer tests that TC_{bw}.

5.4.2 INSURANCE

For this network, we find similar behaviors to Alarm, shown in the graphs on p. 1331. The most striking difference is the clear tendency of Bach-Jordan to add more arcs when more data is available for this more densely connected network. Between the 5th and 6th sample sizes, there is again a change of σ. Comparing the curves of the missing and extra arcs, we see that this changes the tradeoff between false negatives and false positives.

In this case, too, TC_{bw} outperforms TC with low sample sizes (because it misses fewer arcs) but is outperformed with bigger data sets (because it adds too many). Both PC and GS, while being slightly better than TC_{bw} for $n < 100$, are outperformed starting at about $n = 500$. Note the overall good performance of GS in terms of arc orientation errors. The corresponding curve also decreases more smoothly with larger data sets. The Bach-Jordan method is unexpectedly fast on this data set, although poorly accurate. The pattern of the number of statistical tests is very similar to that of Alarm.

5.4.3 HAILFINDER

This network poses a problem to PC: we divided its run time and the number of tests by 10 in the graphs of Figure 16 on p. 1332. Because of its long run times, PC was run only once for each point in the plots, so that the error bars are missing. PC runs into trouble because of the node cluster around variable 27 in the network (see Figure 6): it tries to separate it from the other nodes by doing

subset searches on its large number of neighbors. In order to speed it up, we set the maximum node fan-in parameter to 6, so that PC would not attempt to conduct conditional-independence tests with conditioning sets larger than 6 (we see in Figure 16 how this imposes an upper bound on the run times of PC). TC and TC_{bw} do not run into this problem, because this cluster is correctly left alone after the feature-selection step, done in $O(d^3)$ operations. Note that TC, TC_{bw}, and GS *would* also spend a long time on this cluster if all neighbors of variable 27 were its parents, because they would contain a lot of extra spouse links to be checked with an exponential number of combinations. But this example shows that a local lack of sparseness is fatal to the efficiency of PC, whereas other algorithms can still deal with it if the density of the connections is caused by children rather than parents.

This network shows more clearly the missing arc problem that TC has with low sample size, and the benefits of using TC_{bw} rather than TC here, at least for $n < 2000$. On this network, GS performs overall well. It is beaten by TC only for $n > 2000$, but performs better than all others for $n < 200$. Bach-Jordan still exhibits the same tendency to add more arcs when more data is available. For this data set, σ changes twice, between the 4th and 5th, and between the 5th and 6th data set sizes. The 5th sample size seems to have generated an unfavorable data set for PC, as the number of extra arcs is particularly high.

Examining the run times designates TC as the fastest. This is important especially with larger sample sizes, as TC is often both the fastest and most accurate algorithm.

5.4.4 CARPO

The results for this network are shown on p. 1333. The structural particularity of this network is multiple cases of a single variable having many children. PC performs overall badly on this network. For $n < 200$, GS is the clear winner: all other algorithms make many more errors. The plain TC especially misses many arcs. For $n > 500$, however, both TC and TC_{bw} slightly but consistently outperform GS. At $n = 800$, TC beats TC_{bw}. Bach-Jordan, although fast on this instance, adds again too many extra links, and makes numerous directionality errors.

5.4.5 DIABETES

This is our largest and final test network. The error patterns are most similar to those of the Insurance network, with the exception of Bach-Jordan, which performs more poorly here. We can detect two changes of σ: between the 3rd and 4th, and between the 5th and 6th sample sizes.

Starting at $n > 1000$, all constraint-based methods seem to yield similar overall accuracy. GS is better in terms of directionality errors; TC and TC_{bw} are better in terms of missed links. For $n > 4000$, TC and TC_{bw} have the same accuracy and slightly beat GS and PC, while they are beaten significantly for $n < 800$. We note that the extra links added by GS seem to allow it to obtain a better directionality accuracy than in our first series of experiments, where it was given the exact moral graph as input.

5.4.6 DISCUSSION

Both TC and TC_{bw} slightly but consistently beat the other competitors when the sample size exceeds one or two thousand, depending on the network. They are usually weaker with low sample sizes because of missed arcs. GS beats TC with small data sets, because of the way that PC goes through conditioning sets for the statistical tests (Tsamardinos et al., 2006, discuss in detail this particular

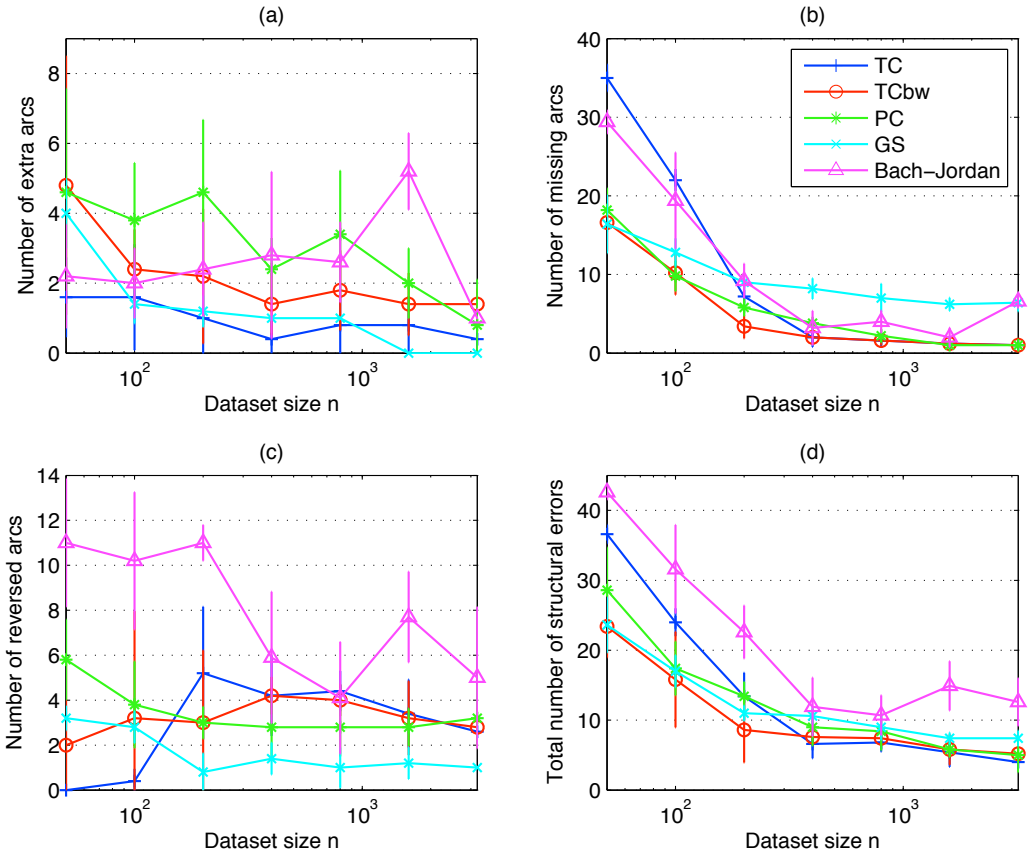


Figure 11: Differentiated errors on Alarm as a function of the sample size n: (a) extra arcs; (b) missing arcs; (c) reversed arcs; (d) total sum.

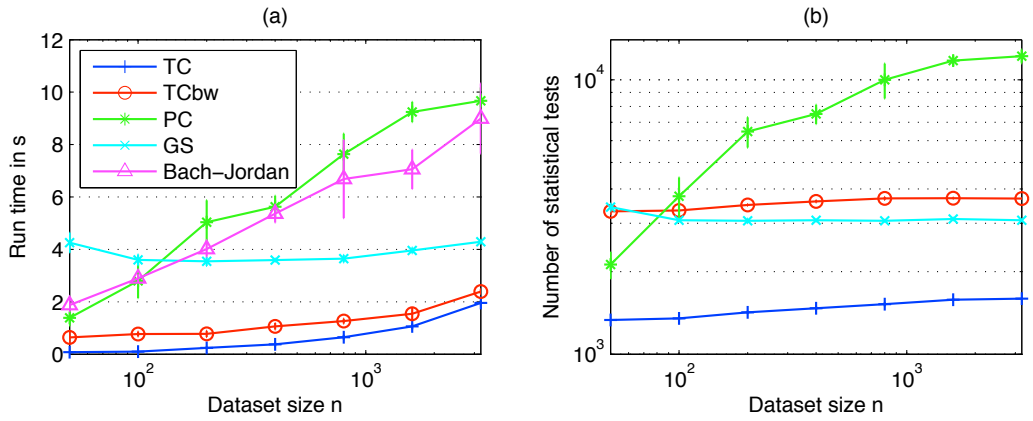


Figure 12: Alarm: (a) run times and (b) number of statistical tests as a function of the sample size n.

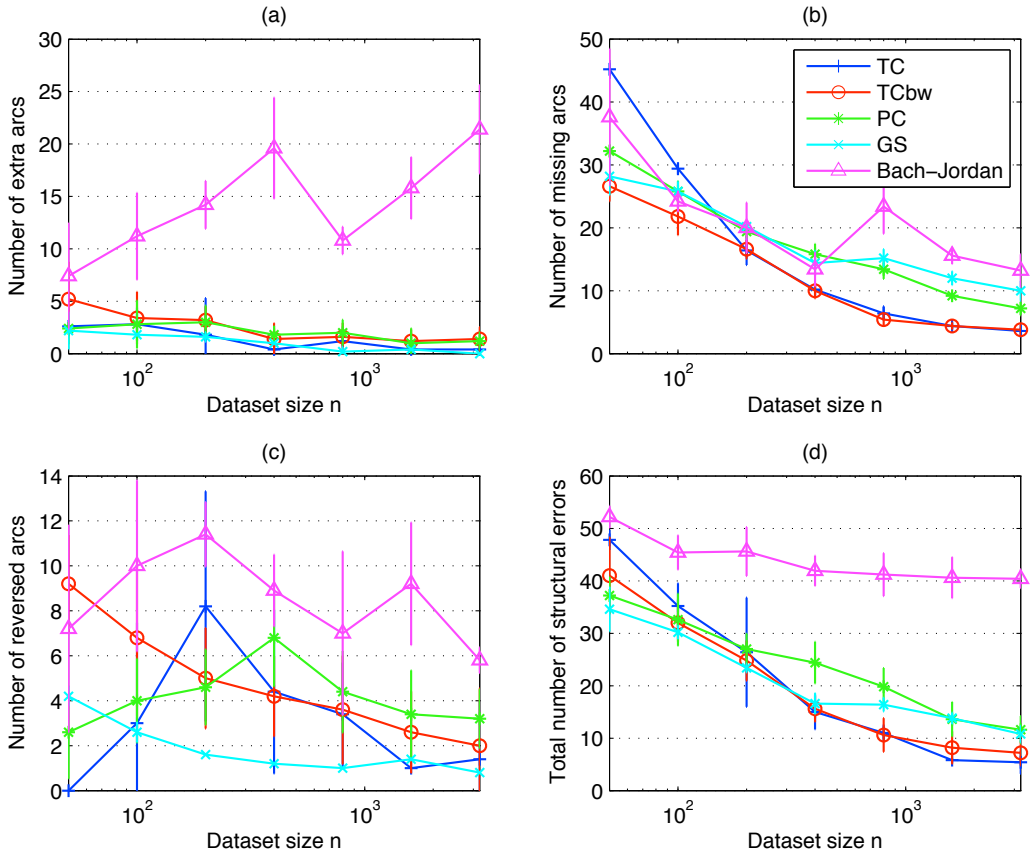


Figure 13: Differentiated errors on Insurance as a function of the sample size n: (a) extra arcs; (b) missing arcs; (c) reversed arcs; (d) total sum.

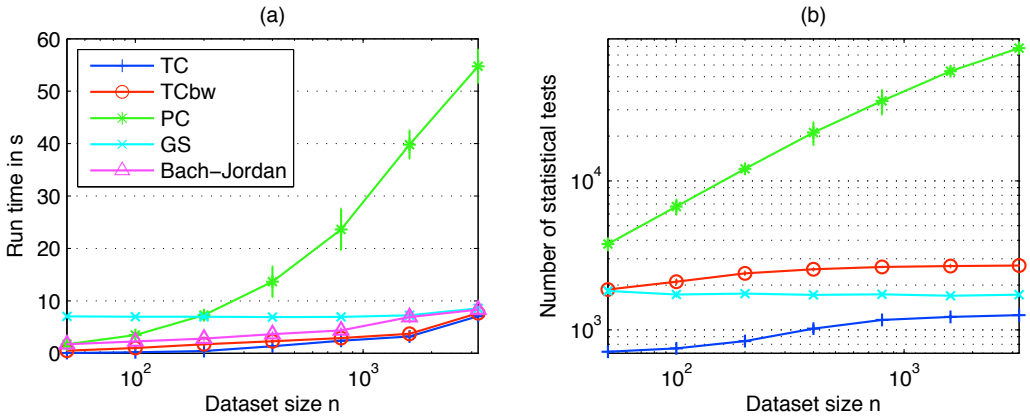


Figure 14: Insurance: (a) run times and (b) number of statistical tests as a function of the sample size n.

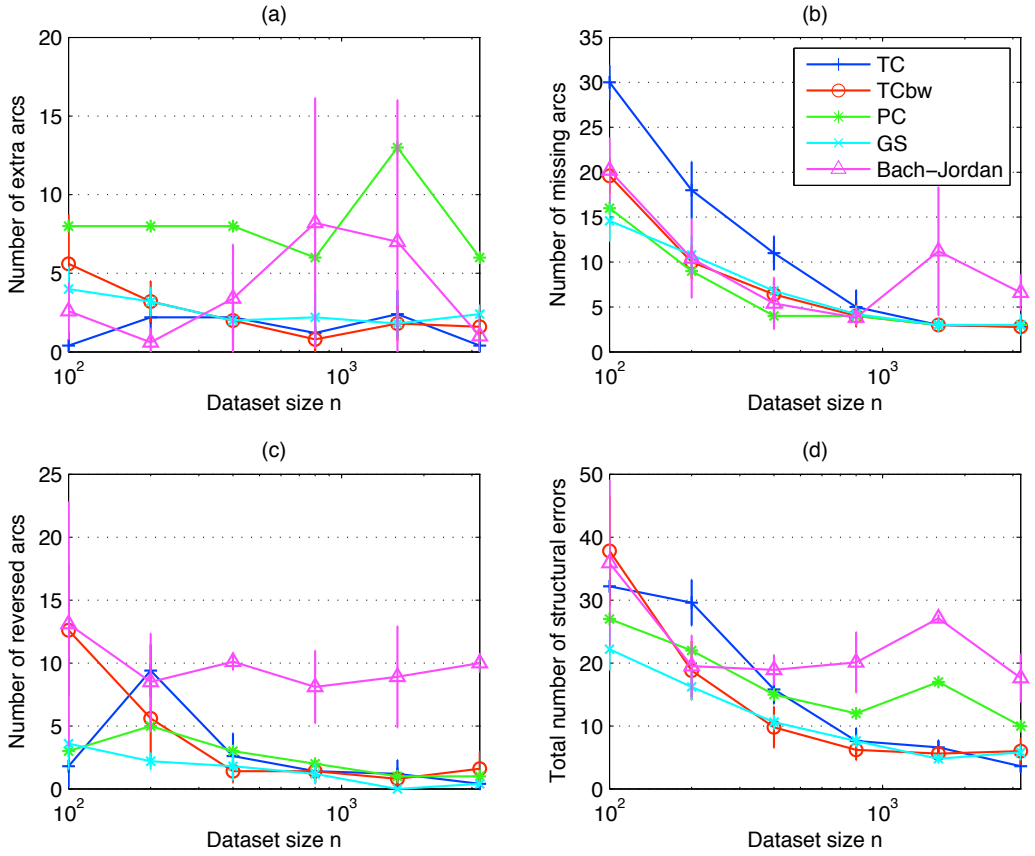


Figure 15: Differentiated errors on Hailfinder as a function of the sample size n: (a) extra arcs; (b) missing arcs; (c) reversed arcs; (d) total sum.

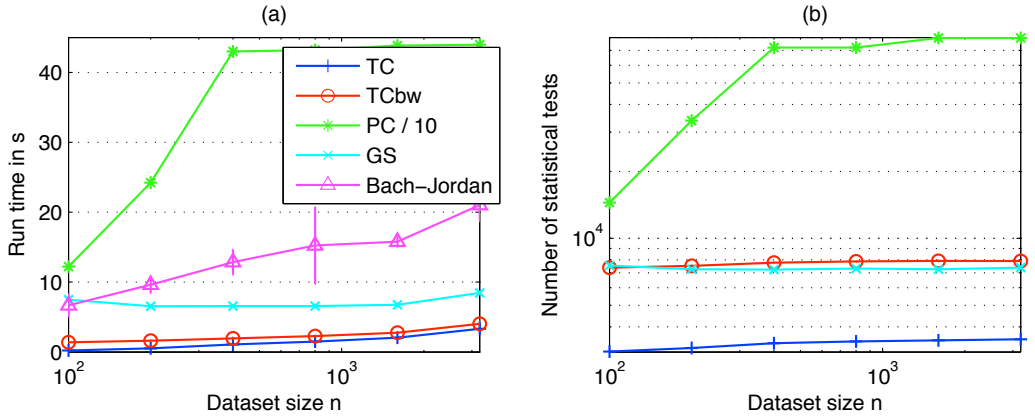


Figure 16: Hailfinder: (a) run times and (b) number of statistical tests as a function of the sample size n.

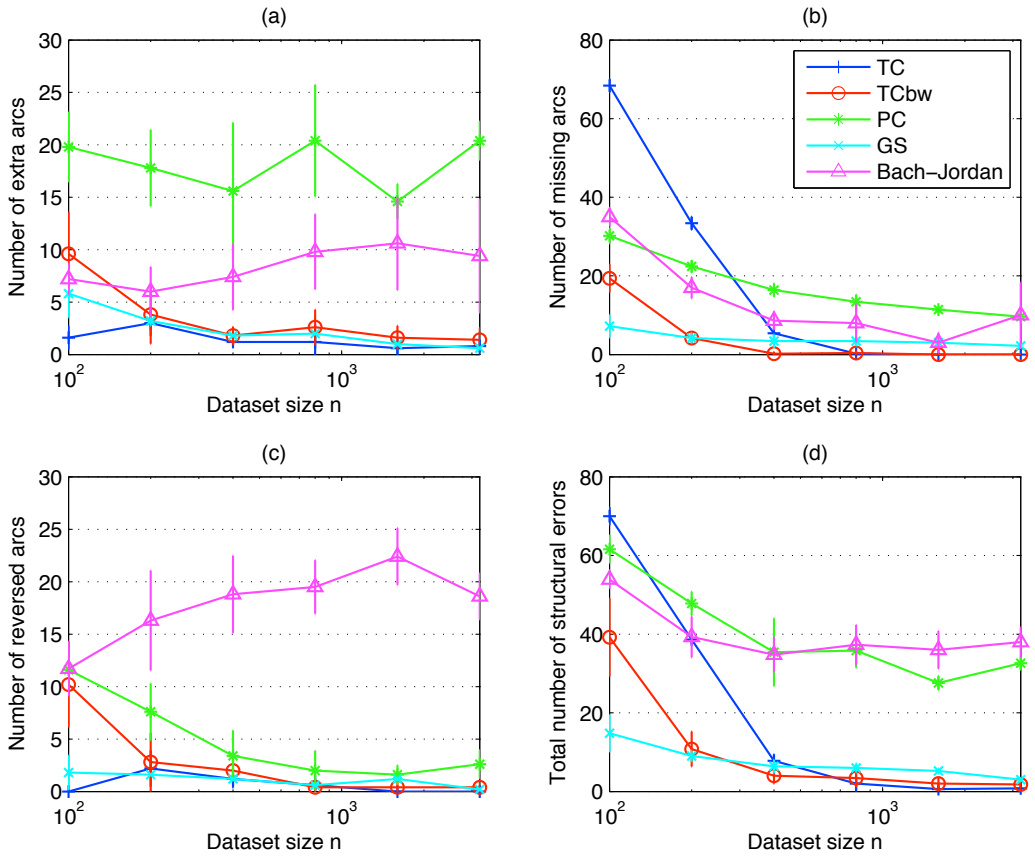


Figure 17: Differentiated errors on Carpo as a function of the sample size n: (a) extra arcs; (b) missing arcs; (c) reversed arcs; (d) total sum.

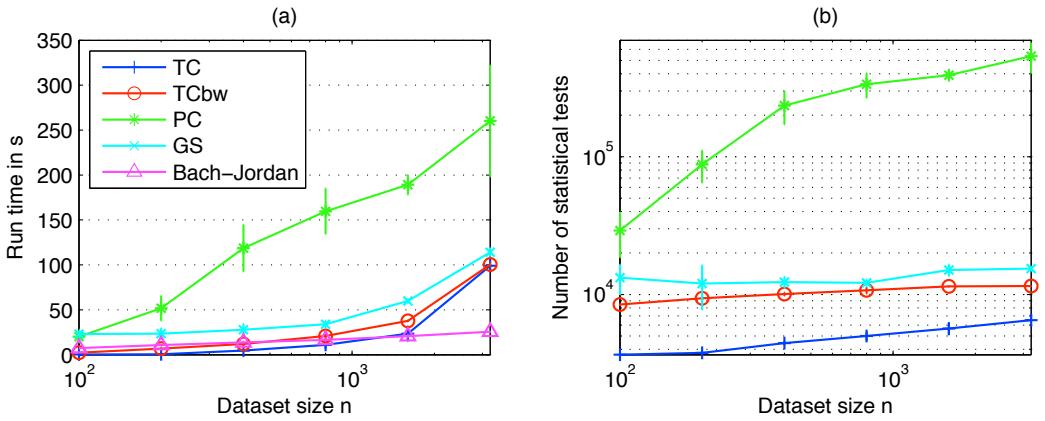


Figure 18: Carpo: (a) run times and (b) number of statistical tests as a function of the sample size n.

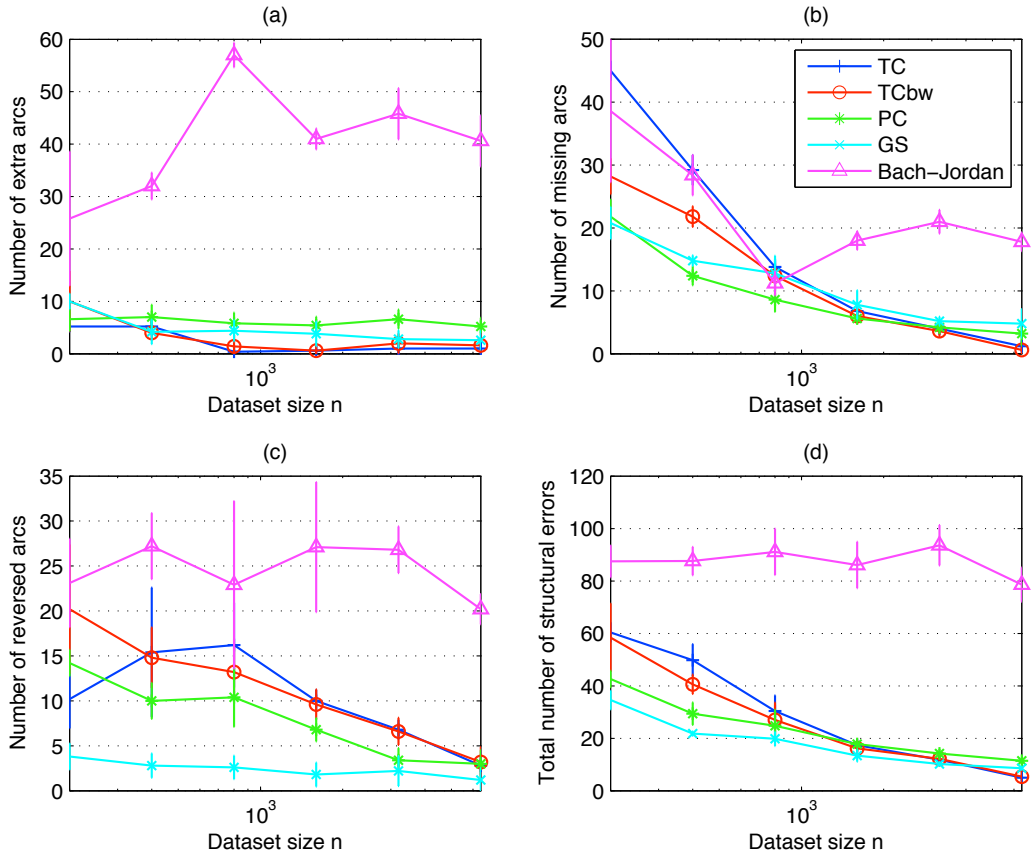


Figure 19: Differentiated errors on Diabetes as a function of the sample size n: (a) extra arcs; (b) missing arcs; (c) reversed arcs; (d) total sum.

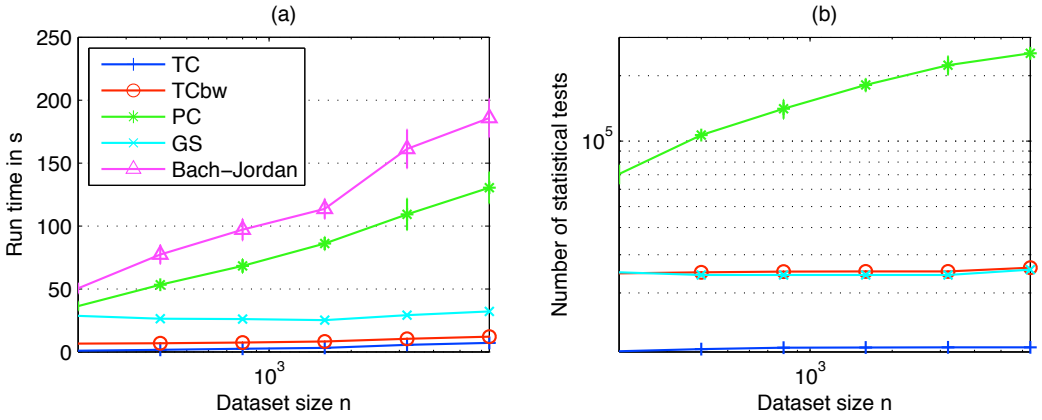


Figure 20: Diabetes: (a) run times and (b) number of statistical tests as a function of the sample size n.

issue in the case of tests with discrete variables). The score-based Bach-Jordan method was found difficult to tune with the parameter σ. For this multivariate Gaussian case, its performance is usually worse than the other tested algorithms. This also reflects the fact PC, GS, TC and TC_{bw} with *z*-tests are all "tuned" for multivariate Gaussian data. The additional errors made by Bach-Jordan reflect the price of being more generic.

In terms of run time, PC is slowed down by nodes with a high degree, whereas TC or GS handle them without the exponential time complexity growth if they are not part of triangles, as in Hailfinder. In general, TC and TC_{bw} resolve all conditional-independence relations (up to married parents) in the feature-selection step in $O(d^3)$ and $O(d^4)$, respectively, whereas all PC can do in $O(d^{2+\alpha})$ is resolve conditional-independence relations with conditioning sets of cardinality α. It is then reasonable to expect algorithms like GS, TC and TC_{bw} to scale better than PC on sparse networks where nodes have a small number of parents. The exponential growth in PC can be seen in case the nodes have a high degree, be it parents or children; in TC and GS, it is due to large fully-connected triangle structures and to spouse links coming from the Markov blanket-construction step. And whereas these large structures imply a high degree, the converse is not true (for instance in the Hailfinder network). So, PC will exhibit an exponential behavior on all problem instances where TC and GS also exhibits this behavior, but the converse is not true.

It is interesting to investigate what kind of high-degree structure is more likely to appear. If it is a node with many children (as node 27 in Hailfinder), which we call an *explosion pattern*, TC can handle it efficiently. If it is a node with many parents, an *implosion pattern*, then none of these algorithms can recognize it in polynomial time. Explosion patterns correspond to a single cause that has many effects; implosion patterns correspond to many causes leading to the same effect. It remains open for discussion to know which one is more likely to occur with real-life data sets.

GS could not be beaten on small sample sizes. It is yet an unsolved challenge for TC and TC_{bw} to handle problems where the number of variables exceeds the number of samples, as in gene expression networks, thus leading to an attempt at inverting a matrix that does not have full rank. Regularizing the covariance matrix might help make TC_{bw} more robust in this case. Computationally, TC_{bw} does add a degree of complexity with respect to TC, and the number of tests that TC_{bw} performs is usually comparable to GS.

TC_{bw} helps solving problems with TC and small data sets, but still cannot operate below the $n = d$ threshold. The exact sample size where TC_{bw} stops performing better than TC does not appear to be a simple function of the n or d but depends on the structure of the network. It would be useful to investigate when the feature-selection addition of TC_{bw} becomes irrelevant. And as GS is more accurate with small sample sizes, finding a similarly testable condition predicting the threshold where TC is more accurate than GS would allow to merge the approaches into a single algorithm that knows which Markov blanket approach to use in order to achieve better results.

6. Conclusion

Causal discovery and feature selection are closely related: optimal feature selection discovers Markov blanket as sets of strongly relevant features, and causal discovery discovers Markov blankets as direct causes, direct effects, and common causes of direct effects. By performing perfect feature selection on each variable, we get the undirected moral graph as an approximation of the causal graph. An extra step, the collider set identification, is needed in order to transform the Markov blankets into parents, children, and spouses. This step is exponential in the worst case,

but is actually efficient provided the graph is sparse enough—a common assumption of many algorithms. We proposed an algorithm to do this task and compared it favorably to the similar steps of the Grow-Shrink algorithm.

Determining the Markov blanket with existing backward feature elimination like RFE eliminates the irrelevant variables in the large sample limit, but remains too inclusive. Global corrections have to be made, such as for instance insuring that a variable in the selected Markov blanket of a target also includes this target in its own selected Markov blanket. We conducted experiments that confirmed that this adjustment discards most false positives, and thus provided a hint that the approach is consistent in the large-sample limit. The main challenge is to perform feature selection for all variables in an efficient way. This task is tractable with the multivariate Gaussian assumption. We presented the TC and the TC_{bw} algorithms, which fit into the described framework, and compared them to PC, GS, and a Bayesian structure-learning method. For small sample sizes, GS usually makes fewer structural errors, and TC/TC_{bw} are better for larger samples sizes.

We are convinced of the superiority of the Markov blanket approaches as described in this paper. We invoke as support for this claim the high run times of PC, and the good low and high sample size accuracy of GS and TC/TC_{bw}, respectively. Not only are Markov blanket techniques much more scalable, they can be more accurate; they are also more easily modifiable to construct only parts the network deemed relevant by some criterion.

The biggest challenges we face now with causal structure learning include robust and consistent distribution-free structure learning with continuous and potentially highly nonlinear data. In the future, we intend to make use of this framework to develop such techniques and thus try to get rid of the Gaussianity assumption, often impractical with real-life data sets.

Acknowledgments

We would like to thank Dimitris Margaritis, who kindly provided us with his C implementation of the Recursive Median conditional-independence test; and Francis Bach and Lawrence Fu, who provided us with a Matlab/C implementation of the Bach-Jordan algorithm. We also thank the anonymous reviewers for their helpful comments and pointers, which led to a significantly enhanced version of this paper.

Appendix A.

For all proofs, we assume the given data set D is faithful.

Lemma 12 *In a DAG $\mathcal{G}$, any (undirected) path π of length $\ell(\pi) > 2$ can be blocked by conditioning on any two consecutive nodes in π.*

Proof It follows from Definition 5 that a path π is blocked when either at least one collider (or one of its descendants) is not in the conditioning set $\mathbf{S}$, or when at least one non-collider is in $\mathbf{S}$. It therefore suffices to show that conditioning on two consecutive nodes always includes a non-collider. This is the case because two consecutive colliders would require bidirected arrows, which is a structural impossibility with simple DAGs. ■

Lemma 13 *In a DAG $\mathcal{G}$, two nodes X,Y are* d*-connected given all other nodes $\mathbf{S} = \mathbf{V} \setminus \{X,Y\}$ if and only if any of the following conditions holds:*

(i) There is an arc from X to Y or from Y to X (i.e., $X \to Y$ or $X \leftarrow Y$);
(ii) X and Y have a common child Z (i.e., $X \to Z \leftarrow Y$).

Proof We prove this by first proving an implication and then its converse.

($\Longleftarrow$) If (i) holds, then X and Y cannot be d-separated by any set. If (ii) holds, then Z is included in the conditioning set and d-connects X and Y by Definition 5.

($\Longrightarrow$) X and Y are d-connected given a certain conditioning set when at least one path remains open. Using the conditioning set $\mathbf{S}$, paths of length > 2 are blocked by Lemma 12 since $\mathbf{S}$ contains all nodes on those paths. Paths of length 2 contain a mediating variable Z between X and Y; by Definition 5, $\mathbf{S}$ blocks them unless Z is a common child of X and Y. Paths of length 1 cannot be blocked by any conditioning set. So the two possible cases where X and Y will be d-connected are (i) or (ii). ■

Corollary 14 *Two variables X,Y are dependent given all other variables $\mathbf{S} = \mathbf{V} \setminus \{X,Y\}$ if and only if any of the following conditions holds:*

(i) X causes Y or Y causes X;
(ii) X and Y have a common effect Z.

Proof It follows directly from Lemma 13 due to the faithful structure, which ensures that there exists a DAG where conditional independence and d-separation map one-to-one. Lemma 13 can then be reread in terms of conditional independence and causation instead of d-separation and arcs. ■

Property 7 (Total conditioning) *In the context of a faithful causal graph $\mathcal{G}$, we have:*

$$\forall X,Y \in \mathbf{V}: \ \big(X \in \mathbf{Mb}(Y) \iff (X \not\perp\!\!\!\perp Y \mid \mathbf{V} \setminus \{X,Y\})\big).$$

Proof This is a direct consequence of Corollary 14, where points *(i)* and *(ii)* lead to the definition of the Markov blanket of Y as *(i)* all its causes and effects, and *(ii)* the other direct causes of its effects. This is equivalent to $\mathbf{Mb}(Y)$ in $\mathcal{G}$. ■

Lemma 15 *When it exists, the subset $\mathbf{Z}$ that has the Collider Set property for the pair (X,Y) is the set of all direct common effects of X and Y.*

Proof We prove this using $\mathbf{Z}$ and a corresponding $\mathbf{S}_{XY}$ that fulfills (7).

($\Longrightarrow$) ($Z_i \in \mathbf{Z} \implies X \to Z_i \leftarrow Y$.) By (7) and (8), we know that each Z_i opens a dependence path between X and Y (which are independent given $\mathbf{S}_{XY}$) by conditioning on $\mathbf{S}_{XY} \cup \{Z_i\}$. By Definition 5, conditioning on Z_i opens a path if Z_i is either a colliding node or one of its descendants. As, by

definition, $\mathbf{Z} \subseteq \mathbf{Tri}(X - Y)$, we are in the first case. We conclude that Z_i is a direct effect of both X and Y.

($\Longleftarrow$) ($X \twoheadrightarrow Z_i \twoheadleftarrow Y \implies Z_i \in \mathbf{Z}$.) Note that (7) and (8) together are implied in presence of a V-structure $X \twoheadrightarrow Z_i \twoheadleftarrow Y$. Thus, a direct effect is compatible with the conditions. The fact that $\mathbf{Z}$ captures all direct effects follows from the maximization of its cardinality. ■

Lemma 9 *In the context of a faithful causal graph, the set* $\mathbf{Z}$ *that has the Collider Set property for a given pair* (X,Y) *exists if and only if* X *is neither a direct cause nor a direct effect of* Y*, and is unique when it exists.*

Proof The fact that $\mathbf{Z}$ exists if and only if X is neither a direct cause nor a direct effect of Y is a direct consequence of (7), which states that X and Y can be made conditionally independent. This is in contradiction with direct causation.

We now show unicity, using interchangeably the criteria of d-separation and conditional independence, as allowed by the Faithfulness assumption. Suppose that, for a pair (X,Y), two sets $\mathbf{Z}$, $\mathbf{W}$ have been found that fulfill the Collider Set property, with the corresponding d-separating sets $\mathbf{S}_{XY}^{\mathbf{Z}} \subseteq \mathbf{V} \setminus \{X,Y\} \setminus \mathbf{Z}$ and $\mathbf{S}_{XY}^{\mathbf{W}} \subseteq \mathbf{V} \setminus \{X,Y\} \setminus \mathbf{W}$ fulfilling (7). Let $\mathbf{Z}^\star = \mathbf{Z} \setminus \mathbf{W}$. Due to symmetry, proving that $\mathbf{Z}^\star$ is empty proves that $\mathbf{Z} = \mathbf{W}$.

Suppose that $\mathbf{Z}^\star \neq \emptyset$; that is, $\exists Z \in \mathbf{Z}^\star$. Then, by definition, we have that $\left(X \perp\!\!\!\perp Y \;\middle|\; \mathbf{S}_{XY}^{\mathbf{Z}}\right)$ and $\left(X \not\perp\!\!\!\perp Y \;\middle|\; \mathbf{S}_{XY}^{\mathbf{Z}} \cup \{Z\}\right)$. We now have two cases: either (i) $Z \notin \mathbf{S}_{XY}^{\mathbf{W}}$, or (ii) $Z \in \mathbf{S}_{XY}^{\mathbf{W}}$. In the former case (i), consider the set $\mathbf{W}' = \mathbf{W} \cup \{Z\}$. Then $\mathbf{W}'$ also fulfills the Collider Set property with the same d-separating set $\mathbf{S}_{XY}^{\mathbf{W}}$: the only additional condition is $\left(X \not\perp\!\!\!\perp Y \;\middle|\; \mathbf{S}_{XY}^{\mathbf{W}} \cup \{Z\}\right)$. This holds because, as shown by Lemma 15, Z is a direct child of X and Y, and conditioning on it opens a path, no matter what the conditioning set is. But all this is in contradiction with the definition stating that any set fulfilling this property must be the largest set to do so, because the cardinality of $\mathbf{W}'$ is greater than that of $\mathbf{W}$.

In the latter case (ii), the d-separating set $\mathbf{S}_{XY}^{\mathbf{W}}$ contains Z. But this is impossible due to the same reason that Z is a direct child of both X and Y and that thus any set containing Z cannot d-separate X and Y. We therefore conclude $\mathbf{Z}^\star = \emptyset$ and $\mathbf{Z} = \mathbf{W}$, which leads to the uniqueness of the set fulfilling the Collider Set property. ■

Theorem 10 *In the large sample limit, for faithful, causally sufficient data sets, the procedure* RESOLVEMARKOVBLANKETS_COLLIDERSETS *correctly identifies all V-structures and all spouse links, assuming consistent statistical tests.*

Proof First, we note that in a moral graph, a node X is connected to its parents, children, and spouses. Thus, all spouse links to be removed are in the moral graph, and, by the definition of spouse, each spouse link between X and Y corresponds to at least one unshielded collider for the pair (X,Y). Additionally, by the definition of unshielded collider, X and Y are nonadjacent, so that for each spouse link $X - Y$ there is a set $\mathbf{S}_{XY}$ such that $(X \perp\!\!\!\perp Y \mid \mathbf{S}_{XY})$ by the contraposition of (4). So, when such a set $\mathbf{S}_{XY}$ is found, the link $X - Y$ is removed, and for each Z such that $X - Z - Y$ and $Z \notin \mathbf{S}_{XY}$, we orient the triplet as $X \rightarrow Z \leftarrow Y$ for the exact same reason that allows IC (or PC) to do the same in Step 2 of the algorithm (Pearl, 2000). The proof boils down to proving that the proposed search procedure always identify a d-separating set $\mathbf{S}_{XY}$ when there is one.

If some S_{XY} exists, then the link between X and Y is a spouse link by definition of a moral graph, which implies that X and Y have a nonempty set of common effects $\mathbf{Z}$. Each $Z \in \mathbf{Z}$ is linked to both X in Y and is thus in $\mathbf{Tri}(X-Y)$ by definition. Let us assume we can d-separate X and Y by some set: then, by the definition of d-separation, only conditioning on a common effect or a descendant of a common effect can create a dependency. In Algorithm 2, all possible colliders (line 7) and descendants of currently conjectured colliders (line 13) undergo a subset search, such that there will always be one iteration where all colliders and their descendants will be left out of the conditioning set. It is then enough to show that all d-connecting paths between X and Y that are not due to conditioning on a collider or collider's descendant go through the base conditioning set as determined at line 6.

To prove this, we note that the subset search at line 7 will always go through an iteration where it blocks all such d-connecting paths of length 2, that is, patterns of the type $X \to W \to Y$ and $X \leftarrow W \to Y$. As a direct consequence of the fact that we are working on the moral graph, all longer dependency paths go both through a node W in the set of immediate neighbors $\mathbf{Bd}(X)$ of X, and through a node in $\mathbf{Bd}(Y)$. Let us look at $\mathbf{Bd}(X)$. We have two cases: either (i) $W \in \mathbf{Tri}(X-Y)$ and will eventually be blocked by the subset search at line 7, or (ii) $W \in \mathbf{Bd}(X) \setminus \mathbf{Tri}(X-Y)$ (and thus $W \in \mathbf{Bd}(X) \setminus \mathbf{Tri}(X-Y) \setminus \{Y\}$ because $W \neq Y$). This set is exactly the set selected as base conditioning set at line 6, blocking all such paths, up to some symmetry with Y. The fact that we may choose the smaller of the two possible base conditioning sets is due to symmetry reasons. ■

Theorem 16 *If the variables are jointly distributed according to a multivariate Gaussian, TC returns the maximally oriented PDAG of the Markov equivalence class of the DAG representing the causal structure of the data-generating process in the large sample limit, assuming statistically consistent tests.*

Proof An edge is added between X and Y in the feature selection if we find that $\rho_{XY \cdot \mathbf{V} \setminus \{X,Y\}} \neq 0$. We conclude $(X \not\perp\!\!\!\perp Y \mid \mathbf{V} \setminus \{X,Y\})$ owing to the multivariate Gaussian distribution. Corollary 14 says that this implies that X causes Y or Y causes X, or that they share a common child. Therefore, each V-structure is turned into a triangle by the end of the feature-selection step. The collider set search then examines each link $X-Y$ part of a triangle, and by Lemma 15, we know that if the search for a set $\mathbf{Z}$ that has the Collider Set property succeeds, there must be no link between X and Y. We know by the same lemma that this set includes all colliders for the pair (X,Y), so that all V-structures are correctly identified. Step 3 is the same as in the IC or PC algorithms; see Pearl and Verma (1991) and Spirtes et al. (2001). ■

References

B. Abramson, J. Brown, A. Murphy, and R. L. Winkler. Hailfinder: A Bayesian system for forecasting severe weather. *International Journal of Forecasting*, 12:57–71, 1996.

C. F. Aliferis, I. Tsamardinos, and A. Statnikov. HITON, a novel Markov blanket algorithm for optimal variable selection. In *Proceedings of the 2003 American Medical Informatics Association (AMIA) Annual Symposium*, pages 21–25, 2003.

S. Andreassen, R. Hovorka, J. Benn, Kristian G. Olesen, and E. R. Carson. A model-based approach to insulin adjustment. In *Proc. of the Third Conf. on AI in Medicine*, pages 239–248. Springer-Verlag, 1991.

K. Baba, R. Shibata, and M. Sibuya. Partial correlation and conditional correlation as measures of conditional independence. *Australian & New Zealand Journal of Statistics*, 46(4), 2004.

F. R. Bach and M. I. Jordan. Learning graphical models with Mercer kernels. In *Advances in Neural Information Processing Systems 15*, 2003.

I. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proc. of the Second European Conf. on AI in Medicine*, pages 247–256, 1989.

J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29, 1997.

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

G. Elidan. Bayes net repository. Website, 2001. URL `http://compbio.cs.huji.ac.il/Repository/`.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. In *RECOMB*, pages 127–135, 2000.

L. D. Fu. A comparison of state-of-the-art algorithms for learning Bayesian network structures from continuous data. Master's thesis, Vanderbilt University, 2005.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

I. Guyon, C. Aliferis, and A. Elisseeff. Causal feature selection. In H. Liu and H. Motoda, editors, *Computational Methods of Feature Selection*. Chapman and Hall/CRC Press, 2007.

D. Hardin, I. Tsamardinos, and C. F. Aliferis. A theoretical characterization of linear SVM-based feature selection. In *Proceedings of the Twenty First International Conference on Machine Learning*, 2004.

D. M. Hausman and J. Woodward. Independence, invariance and the causal Markov condition. *British Journal for the Philosophy of Science*, 50:521–583, 1999.

G. H. John, R. Kohavi, and K. Pfleger. Irrelevant feature and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 121–129, 1994.

G. G. Judge, R. Carter Hill, W. E. Griffiths, H. Lütkepohl, and T.-C. Lee. *Introduction to the Theory and Practice of Econometrics, 2nd Edition*. Wiley, 1988.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997.

S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50:157–224, 1988.

P. Leray and O. François. BNT structure learning package, 2004. URL `banquiseasi.insa-rouen.fr/projects/bnt-slp/`.

D. Margaritis. Distribution-free learning of Bayesian network structure in continuous domains. In *Proc. of the 20th National Conf. on AI*, 2005.

D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems 12*, 1999.

C. Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, 1995.

R. Nilsson, J. M. Peña, J. Björkegren, and J. Tegnér. Consistent feature selection for pattern recognition in polynomial time. *The Journal of Machine Learning Research*, 8:589–612, 2007.

J. M. Peña, J. Björkegren, and J. Tegnér. Scalable, efficient and correct learning of Markov boundaries under the faithfulness assumption. In *Proceedings of the Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning under Uncertainty*, pages 136–147, 2005.

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

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, Los Altos, 1988.

J. Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–709, 1995.

J. Pearl and T. Verma. A theory of inferred causation. In *Proc. of the Second Int. Conf. on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, 1991.

J.-P. Pellet and A. Elisseeff. A partial correlation-based algorithm for causal structure discovery with continuous variables. In *7th International Symposium on Intelligent Data Analysis*, 2007.

A. Raveh. On the use of the inverse of the correlation matrix in multivariate data analysis. *The American Statistician*, 39:39–42, 1985.

R. Scheines, P. Spirtes, C. Glymour, C. Meek, and T. Richardson. The TETRAD project: Constraint based aids to causal model specification. Technical report, Carnegie Mellon University, Dpt. of Philosophy, 1995.

A. J. Smola and B. Schölkopf. A tutorial on support vector regression. Technical report, NeuroCOLT2, 1998.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search, Second Edition*. The MIT Press, 2001. ISBN 0262194406.

A. Statnikov, D. Hardin, and C. F. Aliferis. Using SVM weight-based methods to identify causally relevant and non-causally relevant variables. Technical report, Vanderbilt University, USA, 2006.

D. Steel. Homogeneity, selection, and the faithfulness condition. Technical report, Michigan State University, Department of Philosophy, 2005.

M. Talih. *Markov Random Fields on Time-Varying Graphs, with an Application to Portfolio Selection*. PhD thesis, Hunter College, 2003.

R. Tibshirani. Regression shrinkage and selection via the lasso. Technical report, University of Toronto, 1994.

I. Tsamardinos and C. Aliferis. Towards principled feature selection: Relevancy. *Artificial Intelligence and Statistics*, 2003.

I. Tsamardinos, C. F. Aliferis, and A. Statnikov. Time and sample efficient discovery of Markov blankets and direct causal relations. In ACM Press, editor, *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 673–678, 2003.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 2006.

Journal of Machine Learning Research 9 (2008) 1343-1368 Submitted 12/06; Revised 2/08; Published 7/08

A Bahadur Representation of the Linear Support Vector Machine

Ja-Yong Koo JYKOO@KOREA.AC.KR
Department of Statistics
Korea University
Seoul, 136-701, Korea

Yoonkyung Lee YKLEE@STAT.OSU.EDU
Department of Statistics
The Ohio State University
Columbus, OH 43210, USA

Yuwon Kim YUWONKIM@NAVER.COM
Data Mining Team
NHN Inc.
Gyeonggi-do 463-847, Korea

Changyi Park PARK463@UOS.AC.KR
Department of Statistics
University of Seoul
Seoul, 130-743, Korea

Editor: John Shawe-Taylor

Abstract

The support vector machine has been successful in a variety of applications. Also on the theoretical front, statistical properties of the support vector machine have been studied quite extensively with a particular attention to its Bayes risk consistency under some conditions. In this paper, we study somewhat basic statistical properties of the support vector machine yet to be investigated, namely the asymptotic behavior of the coefficients of the linear support vector machine. A Bahadur type representation of the coefficients is established under appropriate conditions, and their asymptotic normality and statistical variability are derived on the basis of the representation. These asymptotic results do not only help further our understanding of the support vector machine, but also they can be useful for related statistical inferences.

Keywords: asymptotic normality, Bahadur representation, classification, convexity lemma, Radon transform

1. Introduction

The support vector machine (SVM) introduced by Cortes and Vapnik (1995) has been successful in many applications due to its high classification accuracy and flexibility. For reference, see Vapnik (1996), Schölkopf and Smola (2002), and Cristianini and Shawe-Taylor (2000). In parallel with a wide range of applications, statistical properties of the SVM have been studied by many researchers recently in addition to the statistical learning theory by Vapnik (1996) that originally motivated the SVM. These include studies on the Bayes risk consistency of the SVM (Lin, 2002; Zhang, 2004; Steinwart, 2005) and its rate of convergence to the Bayes risk (Lin, 2000; Blanchard, Bousquet, and Massart, 2008; Scovel and Steinwart, 2007; Bartlett, Jordan, and McAuliffe, 2006). While the

existing theoretical analysis of the SVM largely concerns its asymptotic risk, there are some basic statistical properties of the SVM that seem to have eluded our attention. For example, to the best of our knowledge, large sample properties of the coefficients in the linear SVM have not been studied so far although the magnitude of each coefficient is often the determining factor of feature selection for the SVM in practice.

In this paper, we address this basic question of the statistical behavior of the linear SVM as a first step to the study of more general properties of the SVM. We mainly investigate asymptotic properties of the coefficients of variables in the SVM solution for linear classification. The investigation is done in the standard way that parametric methods are studied in a finite dimensional setting, that is, the number of variables is assumed to be fixed and the sample size grows to infinity. Additionally, in the asymptotics, the effect of regularization through maximization of the class margin is assumed to vanish at a certain rate so that the solution is ultimately governed by the empirical risk. Due to these assumptions, the asymptotic results become more pertinent to the classical parametric setting where the number of features is moderate compared to the sample size and the virtue of regularization is minute than to the situation with high dimensional inputs. Despite the difference between the practical situation where the SVM methods are effectively used and the setting theoretically posited in this paper, the asymptotic results shed a new light on the SVM from a classical parametric point of view. In particular, we establish a Bahadur type representation of the coefficients as in the studies of sample quantiles and estimates of regression quantiles. See Bahadur (1966) and Chaudhuri (1991) for reference. It turns out that the Bahadur type representation of the SVM coefficients depends on Radon transform of the second moments of the variables. This representation illuminates how the so called margins of the optimal separating hyperplane and the underlying probability distribution within and around the margins determine the statistical behavior of the estimated coefficients. Asymptotic normality of the coefficients then follows immediately from the representation. The proximity of the hinge loss function that defines the SVM solution to the absolute error loss and its convexity allow such asymptotic results akin to those for least absolute deviation regression estimators in Pollard (1991).

In addition to providing an insight into the asymptotic behavior of the SVM, we expect that our results can be useful for related statistical inferences on the SVM, for instance, feature selection. For introduction to feature selection, see Guyon and Elisseeff (2003), and for an extensive empirical study of feature selection using SVM-based criteria, see Ishak and Ghattas (2005). In particular, Guyon, Weston, Barnhill, and Vapnik (2002) proposed a recursive feature elimination procedure for the SVM with an application to gene selection in microarray data analysis. Its selection or elimination criterion is based on the absolute value of a coefficient not its standardized value. The asymptotic variability of estimated coefficients that we provide can be used in deriving a new feature selection criterion which takes inherent statistical variability into account.

This paper is organized as follows. Section 2 contains the main results of a Bahadur type representation of the linear SVM coefficients and their asymptotic normality under mild conditions. An illustrative example is then provided in Section 3 followed by simulation studies in Section 4 and a discussion in Section 5. Proofs of technical lemmas and theorems are collected in Section 6.

2. Main Results

In this section, we first introduce some notations and discuss our asymptotic results for the linear SVM coefficients.

2.1 Preliminaries

Let (X,Y) be a pair of random variables with $X \in \mathcal{X} \subset \mathbb{R}^d$ and $Y \in \{1,-1\}$. The marginal distribution of Y is given by $\mathbb{P}(Y=1)=\pi_+$ and $\mathbb{P}(Y=-1)=\pi_-$ with $\pi_+,\pi_- > 0$ and $\pi_+ + \pi_- = 1$. Let f and g be the densities of X given $Y=1$ and -1 with respect to the Lebesgue measure. Let $\{(X^i,Y^i)\}_{i=1}^n$ be a set of training data, independently drawn from the distribution of (X,Y). Denote the input variables as $x=(x_1,\ldots,x_d)^\top$ and their coefficients as $\beta_+=(\beta_1,\ldots,\beta_d)^\top$. Let $\widetilde{x}=(\widetilde{x}_0,\ldots,\widetilde{x}_d)^\top=(1,x_1,\ldots,x_d)^\top$ and $\beta=(\beta_0,\beta_1,\ldots,\beta_d)^\top$. We consider linear classifications with hyperplanes defined by $h(x;\beta)=\beta_0+x^\top\beta_+=\widetilde{x}^\top\beta$. Let $\|\cdot\|$ denote the Euclidean norm of a vector. For separable cases, the SVM finds the hyperplane that maximizes the geometric margin, $2/\|\beta_+\|^2$ subject to the constraints $y^i h(x^i;\beta)\geq 1$ for $i=1,\ldots,n$. For non-separable cases, a soft-margin SVM is introduced to minimize

$$C\sum_{i=1}^n \xi_i + \frac{1}{2}\|\beta_+\|^2$$

subject to the constraints $\xi_i \geq 1-y^i h(x^i;\beta)$ and $\xi_i \geq 0$ for $i=1,\ldots,n$, where $C>0$ is a tuning parameter and $\{\xi_i\}_{i=1}^n$ are called the slack variables. Equivalently, the SVM minimizes the unconstrained objective function

$$l_{\lambda,n}(\beta)=\frac{1}{n}\sum_{i=1}^n\Big[1-y^i h(x^i;\beta)\Big]_+ + \frac{\lambda}{2}\|\beta_+\|^2 \tag{1}$$

over $\beta\in\mathbb{R}^{d+1}$, where $[z]_+=\max(z,0)$ for $z\in\mathbb{R}$ and $\lambda>0$ is a penalization parameter; see Vapnik (1996) for details. Let the minimizer of (1) be denoted by $\widehat{\beta}_{\lambda,n}=\arg\min_\beta l_{\lambda,n}(\beta)$. Note that $C=(n\lambda)^{-1}$. Choice of λ depends on the data, and usually it is estimated via cross validation in practice. In this paper, we consider only nonseparable cases and assume that $\lambda\to 0$ as $n\to\infty$. We note that separable cases require a different treatment for asymptotics because λ has to be nonzero in the limit for the uniqueness of the solution.

Before we proceed with a discussion of the asymptotics of the $\widehat{\beta}_{\lambda,n}$, we introduce some notation and definitions first. The population version of (1) without the penalty term is defined as

$$L(\beta)=\mathbb{E}\Big[1-Yh(X;\beta)\Big]_+ \tag{2}$$

and its minimizer is denoted by $\beta^*=\arg\min_\beta L(\beta)$. Then the population version of the optimal hyperplane defined by the SVM is

$$\widetilde{x}^\top\beta^*=0. \tag{3}$$

Sets are identified with their indicator functions. For example, $\int_{\mathcal{X}} x_j\{x_j>0\}f(x)dx = \int_{\{x\in\mathcal{X}\,:\,x_j>0\}} x_j f(x)dx$. Letting $\psi(z)=\{z\geq 0\}$ for $z\in\mathbb{R}$, we define $S(\beta)=(S(\beta)_j)$ to be the $(d+1)$-dimensional vector given by

$$S(\beta)=-\mathbb{E}\Big(\psi(1-Yh(X;\beta))Y\widetilde{X}\Big)$$

and $H(\beta)=(H(\beta)_{jk})$ to be the $(d+1)\times(d+1)$-dimensional matrix given by

$$H(\beta)=\mathbb{E}\Big(\delta(1-Yh(X;\beta))\widetilde{X}\widetilde{X}^\top\Big),$$

where δ denotes the Dirac delta function with $\delta(t) = \psi'(t)$ in distributional sense. Provided that $S(\beta)$ and $H(\beta)$ are well-defined, $S(\beta)$ and $H(\beta)$ are considered as the gradient and Hessian matrix of $L(\beta)$, respectively. Formal proofs of these relationships are given in Section 6.1.

For explanation of $H(\beta)$, we introduce a Radon transformation. For a function s on $\mathcal{X}$, define the Radon transform $\mathcal{R}s$ of s for $p \in \mathbb{R}$ and $\xi \in \mathbb{R}^d$ as

$$(\mathcal{R}s)(p,\xi) = \int_{\mathcal{X}} \delta(p - \xi^\top x) s(x) dx.$$

For $0 \le j,k \le d$, the (j,k)-th element of the Hessian matrix $H(\beta)$ is given by

$$H(\beta)_{jk} = \pi_+ (\mathcal{R}f_{jk})(1-\beta_0, \beta_+) + \pi_- (\mathcal{R}g_{jk})(1+\beta_0, -\beta_+), \tag{4}$$

where $f_{jk}(x) = \widetilde{x}_j \widetilde{x}_k f(x)$ and $g_{jk}(x) = \widetilde{x}_j \widetilde{x}_k g(x)$. Equation (4) shows that the Hessian matrix $H(\beta)$ depends on the Radon transforms of f_{jk} and g_{jk} for $0 \le j,k \le d$. For Radon transform and its properties in general, see Natterer (1986), Deans (1993), or Ramm and Katsevich (1996).

For a continuous integrable function s, it can be easily proved that $\mathcal{R}s$ is continuous. If f and g are continuous densities with finite second moments, then f_{jk} and g_{jk} are continuous and integrable for $0 \le j,k \le d$. Hence $H(\beta)$ is continuous in β when f and g are continuous and have finite second moments.

2.2 Asymptotics

Now we present the asymptotic results for $\widehat{\beta}_{\lambda,n}$. We state regularity conditions for the asymptotics first. Some remarks on the conditions then follow for exposition and clarification. Throughout this paper, we use $C_1, C_2, \ldots$ to denote positive constants independent of n.

(A1) The densities f and g are continuous and have finite second moments.

(A2) There exists $B(x_0, \delta_0)$, a ball centered at x_0 with radius $\delta_0 > 0$ such that $f(x) > C_1$ and $g(x) > C_1$ for every $x \in B(x_0, \delta_0)$.

(A3) For some $1 \le i^* \le d$,

$$\int_{\mathcal{X}} \{x_{i^*} \ge G_{i^*}^-\} x_{i^*} g(x) dx < \int_{\mathcal{X}} \{x_{i^*} \le F_{i^*}^+\} x_{i^*} f(x) dx$$

or

$$\int_{\mathcal{X}} \{x_{i^*} \le G_{i^*}^+\} x_{i^*} g(x) dx > \int_{\mathcal{X}} \{x_{i^*} \ge F_{i^*}^-\} x_{i^*} f(x) dx.$$

Here $F_{i^*}^+, G_{i^*}^+ \in [-\infty, \infty]$ are upper bounds such that $\int_{\mathcal{X}} \{x_{i^*} \le F_{i^*}^+\} f(x) dx = \min\left(1, \frac{\pi_-}{\pi_+}\right)$ and $\int_{\mathcal{X}} \{x_{i^*} \le G_{i^*}^+\} g(x) dx = \min\left(1, \frac{\pi_+}{\pi_-}\right)$. Similarly, lower bounds $F_{i^*}^-$ and $G_{i^*}^-$ are defined as $\int_{\mathcal{X}} \{x_{i^*} \ge F_{i^*}^-\} f(x) dx = \min\left(1, \frac{\pi_-}{\pi_+}\right)$ and $\int_{\mathcal{X}} \{x_{i^*} \ge G_{i^*}^-\} g(x) dx = \min\left(1, \frac{\pi_+}{\pi_-}\right)$.

(A4) For an orthogonal transformation A_{j^*} that maps $\beta_+^*/\|\beta_+^*\|$ to the j^*-th unit vector e_{j^*} for some $1 \le j^* \le d$, there exist rectangles

$$\mathcal{D}^+ = \{x \in M^+ : l_i \le (A_{j^*}x)_i \le v_i \text{ with } l_i < v_i \text{ for } i \ne j^*\}$$

and

$$\mathcal{D}^- = \{x \in M^- : l_i \le (A_{j^*}x)_i \le v_i \text{ with } l_i < v_i \text{ for } i \ne j^*\}$$

such that $f(x) \ge C_2 > 0$ on $\mathcal{D}^+$, and $g(x) \ge C_3 > 0$ on $\mathcal{D}^-$, where $M^+ = \{x \in \mathcal{X} \mid \beta_0^* + x^\top \beta_+^* = 1\}$ and $M^- = \{x \in \mathcal{X} \mid \beta_0^* + x^\top \beta_+^* = -1\}$.

Remark 1

- *(A1) ensures that $H(\beta)$ is well-defined and continuous in β.*
- *When f and g are continuous, the condition that $f(x_0) > 0$ and $g(x_0) > 0$ for some x_0 implies (A2).*
- *The technical condition in (A3) is a minimal requirement to guarantee that β_+^*, the normal vector of the theoretically optimal hyperplane is not zero. Roughly speaking, it says that for at least one input variable, the mean values of the class conditional distributions f and g have to be different in order to avoid the degenerate case of $\beta_+^* = 0$. Some restriction of the supports through $F_{i^*}^+$, $G_{i^*}^+$, $F_{i^*}^-$ and $G_{i^*}^-$ is necessary in defining the mean values to adjust for potentially unequal class proportions. When $\pi_+ = \pi_-$, $F_{i^*}^+$ and $G_{i^*}^+$ can be taken to be $+\infty$ and $F_{i^*}^-$ and $G_{i^*}^-$ can be $-\infty$. In this case, (A3) simply states that the mean vectors for the two classes are different.*
- *(A4) is needed for the positive-definiteness of $H(\beta)$ around β^*. The condition means that there exist two subsets of the classification margins, M^+ and M^- on which the class densities f and g are bounded away from zero. For mathematical simplicity, the rectangular subsets $\mathcal{D}^+$ and $\mathcal{D}^-$ are defined as the mirror images of each other along the normal direction of the optimal hyperplane. This condition can be easily met when the supports of f and g are convex. Especially, if $\mathbb{R}^d$ is the support of f and g, it is trivially satisfied. (A4) requires that $\beta_+^* \ne 0$, which is implied by (A1) and (A3); see Lemma 4 for the proof. For the special case $d = 1$, M^+ and M^- consist of a point. $\mathcal{D}^+$ and $\mathcal{D}^-$ are the same as M^+ and M^-, respectively, and hence (A4) means that f and g are positive at those points in M^+ and M^-.*

Under the regularity conditions, we obtain a Bahadur-type representation of $\widehat{\beta}_{\lambda,n}$ (Theorem 1). The asymptotic normality of $\widehat{\beta}_{\lambda,n}$ follows immediately from the representation (Theorem 2). Consequently, we have the asymptotic normality of $h(x;\widehat{\beta}_{\lambda,n})$, the value of the SVM decision function at x (Corollary 3).

Theorem 1 *Suppose that (A1)-(A4) are met. For $\lambda = o(n^{-1/2})$, we have*

$$\sqrt{n}(\widehat{\beta}_{\lambda,n} - \beta^*) = -\frac{1}{\sqrt{n}} H(\beta^*)^{-1} \sum_{i=1}^{n} \psi(1 - Y^i h(X^i;\beta^*)) Y^i \widetilde{X}^i + o_{\mathbb{P}}(1).$$

Theorem 2 *Suppose (A1)-(A4) are satisfied. For $\lambda = o(n^{-1/2})$,*

$$\sqrt{n}(\widehat{\beta}_{\lambda,n} - \beta^*) \to N\Big(0, H(\beta^*)^{-1} G(\beta^*) H(\beta^*)^{-1}\Big)$$

in distribution, where

$$G(\beta) = \mathbb{E}\Big(\psi(1 - Y h(X;\beta)) \widetilde{X}\widetilde{X}^\top\Big).$$

Remark 2 *Since $\widehat{\beta}_{\lambda,n}$ is a consistent estimator of β^* as $n \to \infty$, $G(\beta^*)$ can be estimated by its empirical version with β^* replaced by $\widehat{\beta}_{\lambda,n}$. To estimate $H(\beta^*)$, one may consider the following nonparametric estimate:*

$$\frac{1}{n}\left[\sum_{i=1}^{n} p_b\Big(1 - Y^i h(X^i;\widehat{\beta}_{\lambda,n})\Big)\widetilde{X}^i(\widetilde{X}^i)^\top\right],$$

where $p_b(t) \equiv p(t/b)/b$, $p(t) \geq 0$ and $\int_{\mathbb{R}} p(t)dt = 1$. Note that $p_b(\cdot) \to \delta(\cdot)$ as $b \to 0$. However, estimation of $H(\beta^*)$ requires further investigation.

Corollary 3 *Under the same conditions as in Theorem 2,*

$$\sqrt{n}\Big(h(x;\widehat{\beta}_{\lambda,n}) - h(x;\beta^*)\Big) \to N\Big(0, \tilde{x}^\top H(\beta^*)^{-1}G(\beta^*)H(\beta^*)^{-1}\tilde{x}\Big)$$

in distribution.

Remark 3 *Corollary 3 can be used to construct a confidence bound for $h(x;\beta^*)$ based on an estimate $h(x;\widehat{\beta}_{\lambda,n})$, in particular, to judge whether $h(x;\beta^*)$ is close to zero or not given x. This may be useful if one wants to abstain from prediction at a new input x if it is close to the optimal classification boundary $h(x;\beta^*) = 0$.*

3. An Illustrative Example

In this section, we illustrate the relation between the Bayes decision boundary and the optimal hyperplane determined by (2) for two multivariate normal distributions in $\mathbb{R}^d$. Assume that f and g are multivariate normal densities with different mean vectors μ_f and μ_g and a common covariance matrix Σ. Suppose that $\pi_+ = \pi_- = 1/2$.

We verify the assumptions (A1)-(A4) so that Theorem 2 is applicable. For normal densities f and g, (A1) holds trivially, and (A2) is satisfied with

$$C_1 \;=\; |2\pi\Sigma|^{-1/2}\exp\left(-\sup_{\|x\|\leq\delta_0}\Big\{(x-\mu_f)^\top\Sigma^{-1}(x-\mu_f),(x-\mu_g)^\top\Sigma^{-1}(x-\mu_g)\Big\}\right)$$

for $\delta_0 > 0$. Since $\mu_f \neq \mu_g$, there exists $1 \leq i^* \leq d$ such that i^*-th elements of μ_f and μ_g are different. By taking $F^+_{i^*} = G^+_{i^*} = +\infty$ and $F^-_{i^*} = G^-_{i^*} = -\infty$, we can show that one of the inequalities in (A3) holds as mentioned in Remark 1. Since $\mathcal{D}^+$ and $\mathcal{D}^-$ can be taken to be bounded sets of the form in (A4) in $\mathbb{R}^{d-1}$, and the normal densities f and g are bounded away from zero on such $\mathcal{D}^+$ and $\mathcal{D}^-$, (A4) is satisfied. In particular, $\beta^*_+ \neq 0$ as implied by Lemma 4.

Denote the density and cumulative distribution function of $N(0,1)$ as ϕ and Φ, respectively. Note that β^* should satisfy the equation $S(\beta^*) = 0$, or

$$\Phi(a_f) = \Phi(a_g) \tag{5}$$

and

$$\mu_f\Phi(a_f) - \phi(a_f)\Sigma^{1/2}\omega^* = \mu_g\Phi(a_g) + \phi(a_g)\Sigma^{1/2}\omega^*, \tag{6}$$

where $a_f = \dfrac{1-\beta_0^*-\mu_f^\top\beta_+^*}{\|\Sigma^{1/2}\beta_+^*\|}$, $a_g = \dfrac{1+\beta_0^*+\mu_g^\top\beta_+^*}{\|\Sigma^{1/2}\beta_+^*\|}$ and $\omega^* = \Sigma^{1/2}\beta_+^*/\|\Sigma^{1/2}\beta_+^*\|$. From (5) and the definition of a_f and a_g, we have $a^* \equiv a_f = a_g$. Hence

$$(\beta_+^*)^\top(\mu_f+\mu_g) = -2\beta_0^*. \tag{7}$$

It follows from (6) that

$$\beta_+^*/\|\Sigma^{1/2}\beta_+^*\| = \frac{\Phi(a^*)}{2\phi(a^*)}\Sigma^{-1}(\mu_f-\mu_g). \tag{8}$$

First we show the existence of a proper constant a^* satisfying (8) and its relationship with a statistical distance between the two classes. Define $\Upsilon(a) = \phi(a)/\Phi(a)$ and let $d_\Sigma(u,v) = \{(u-v)^\top\Sigma^{-1}(u-v)\}^{1/2}$ denote the Mahalanobis distance between u and $v \in \mathbb{R}^d$. Since $\|\omega^*\| = 1$, we have $\Upsilon(a^*) = \|\Sigma^{-1/2}(\mu_f-\mu_g)\|/2$. Since $\Upsilon(a)$ is monotonically decreasing in a, there exists $a^* = \Upsilon^{-1}(d_\Sigma(\mu_f,\mu_g)/2)$ that depends only on μ_f, μ_g, and Σ. For illustration, when the Mahalanobis distances between the two normal distributions are 2 and 3, a^* is given by $\Upsilon^{-1}(1) \approx -0.303$ and $\Upsilon^{-1}(1.5) \approx -0.969$, respectively. The corresponding Bayes error rates are about 0.1587 and 0.06681. Figure 1 shows a graph of $\Upsilon(a)$ and a^* when $d_\Sigma(\mu_f,\mu_g)$=2 and 3.

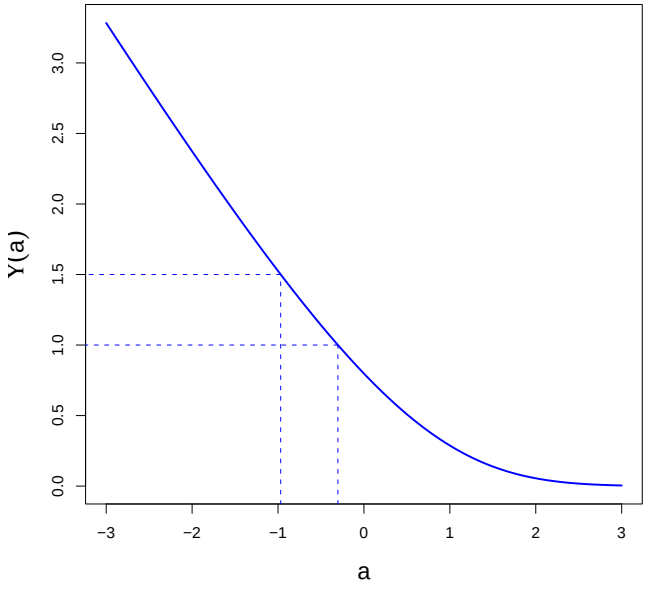


Figure 1: A plot of Υ function. The dashed lines correspond to the inverse mapping from the Mahalanobis distances of 2 and 3 to $a^* \approx -0.303$ and -0.969, respectively.

Once a^* is properly determined, we can express the solution β^* explicitly by (7) and (8):

$$\beta_0^* = -\frac{(\mu_f-\mu_g)^\top\Sigma^{-1}(\mu_f+\mu_g)}{2a^*d_\Sigma(\mu_f,\mu_g)+d_\Sigma(\mu_f,\mu_g)^2}$$

and

$$\beta_+^* = \frac{2\Sigma^{-1}(\mu_f - \mu_g)}{2a^* d_\Sigma(\mu_f,\mu_g) + d_\Sigma(\mu_f,\mu_g)^2}.$$

Thus the optimal hyperplane (3) is

$$\frac{2}{2a^* d_\Sigma(\mu_f,\mu_g) + d_\Sigma(\mu_f,\mu_g)^2}\left\{\Sigma^{-1}(\mu_f-\mu_g)\right\}^\top\left\{x - \frac{1}{2}(\mu_f+\mu_g)\right\} = 0,$$

which is equivalent to the Bayes decision boundary given by

$$\left\{\Sigma^{-1}(\mu_f-\mu_g)\right\}^\top\left\{x - \frac{1}{2}(\mu_f+\mu_g)\right\} = 0.$$

This shows that the linear SVM is equivalent to Fisher's linear discriminant analysis in this setting. In addition, $H(\beta^*)$ and $G(\beta^*)$ can be shown to be

$$G(\beta^*) = \frac{\Phi(a^*)}{2}\begin{bmatrix} 2 & (\mu_f+\mu_g)^\top \\ \mu_f+\mu_g & G_{22}(\beta^*) \end{bmatrix}$$

and

$$H(\beta^*) = \frac{\phi(a^*)}{4}(2a^* + d_\Sigma(\mu_f,\mu_g))\begin{bmatrix} 2 & (\mu_f+\mu_g)^\top \\ \mu_f+\mu_g & H_{22}(\beta^*) \end{bmatrix},$$

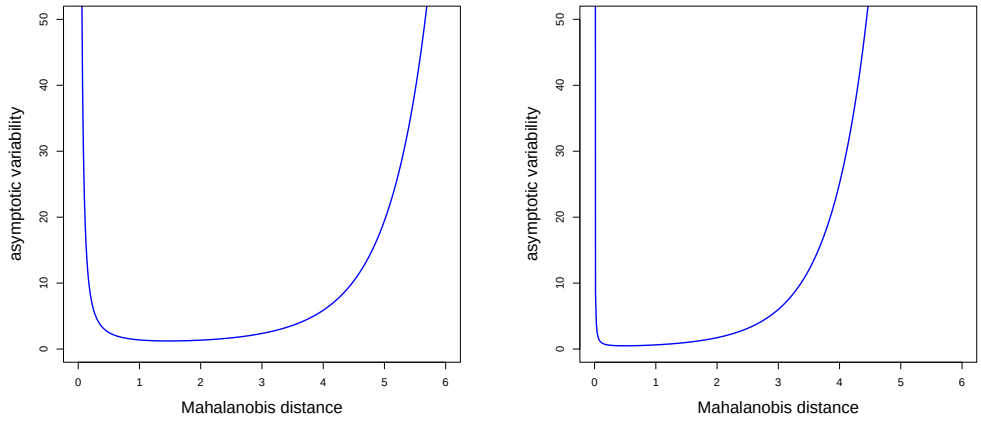
where

$$\begin{aligned} G_{22}(\beta^*) &= \mu_f\mu_f^\top + \mu_g\mu_g^\top + 2\Sigma - \left(\frac{a^*}{d_\Sigma(\mu_f,\mu_g)} + 1\right)(\mu_f-\mu_g)(\mu_f-\mu_g)^\top \text{ and} \\ H_{22}(\beta^*) &= \mu_f\mu_f^\top + \mu_g\mu_g^\top + 2\Sigma \\ &\quad + 2\left(\left(\frac{a^*}{d_\Sigma(\mu_f,\mu_g)}\right)^2 + \frac{a^*}{d_\Sigma(\mu_f,\mu_g)} - \frac{1}{d_\Sigma^2(\mu_f,\mu_g)}\right)(\mu_f-\mu_g)(\mu_f-\mu_g)^\top. \end{aligned}$$

For illustration, we consider the case when $d = 1$, $\mu_f + \mu_g = 0$, and $\sigma = 1$. The asymptotic variabilities of the intercept and the slope for the optimal decision boundary are calculated according to Theorem 2. Figure 2 shows the asymptotic variabilities as a function of the Mahalanobis distance between the two normal distributions, $|\mu_f - \mu_g|$ in this case. Also, it depicts the asymptotic variance of the estimated classification boundary value $(-\hat{\beta}_0/\hat{\beta}_1)$ by using the delta method. Although the Mahalanobis distance roughly in the range of 1 to 4 would be of practical interest, the plots show a notable trend in the asymptotic variances as the distance varies. When the two classes get very close, the variances shoot up due to the difficulty in discriminating them. On the other hand, as the Mahalanobis distance increases, that is, the two classes become more separated, the variances become increasingly large. A possible explanation for the trend is that the intercept and the slope of the optimal hyperplane are determined by only a small fraction of data falling into the margins in this case.

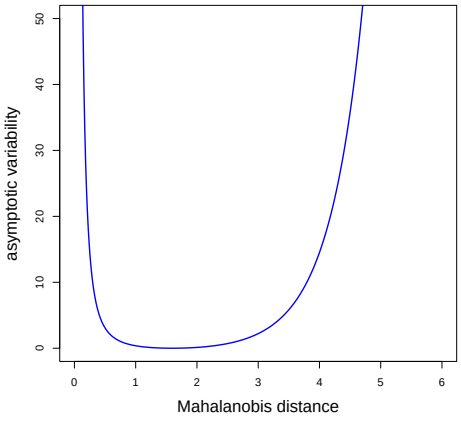
4. Simulation Studies

In this section, simulations are carried out to illustrate the asymptotic results and their potential for feature selection.



(a)

(b)



(c)

Figure 2: The asymptotic variabilities of estimates of (a) the intercept, (b) the slope, and (c) their ratio for the optimal hyperplane as a function of the Mahalanobis distance.

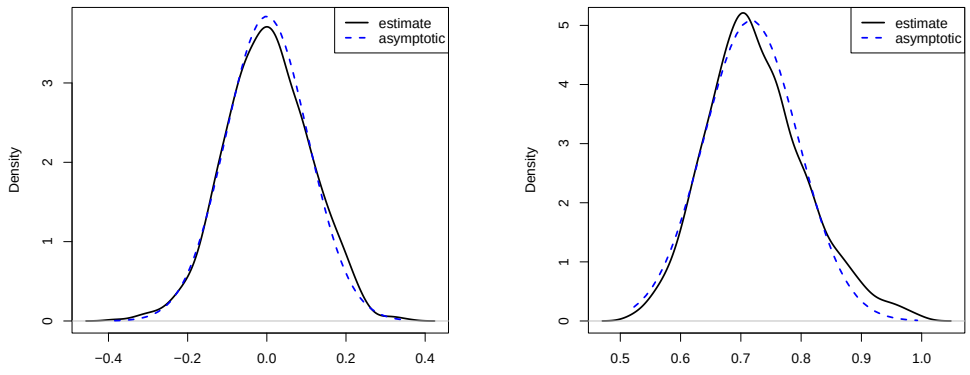
4.1 Bivariate Case

Theorem 2 is numerically illustrated with the multivariate normal setting in the previous section. Consider a bivariate case with mean vectors $\mu_f = (1,1)^\top$ and $\mu_g = (-1,-1)^\top$ and a common covariance matrix $\Sigma = I_2$. This example has $d_\Sigma(\mu_f, \mu_g) = 2\sqrt{2}$ and the corresponding Bayes error rate is 0.07865. Data were generated from the two normal distributions with an equal probability for each class. The total sample size was varied from 100 to 500. To see the direct effect of the hinge loss on the SVM coefficients without regularization as in the way the asymptotic properties in Section 2 are characterized ultimately, we estimated the coefficients of the linear SVM without the penalty term by linear programming. Such a simulation was repeated 1,000 times for each sample

size, and Table 1 summarizes the results by showing the averages of the estimated coefficients of the SVM over 1,000 replicates. As expected, the averages get closer to the theoretically optimal coefficients β^* as the sample size grows. Moreover, the sampling distributions of $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$ approximate their theoretical counterparts for a large sample size as shown in Figure 3. The solid lines are the estimated density functions of $\hat{\beta}_0$ and $\hat{\beta}_1$ for $n = 500$, and the dotted lines are the corresponding asymptotic normal densities in Theorem 2.

Coefficients	Sample size n			Optimal values
	100	200	500	
β_0	0.0006	-0.0013	0.0022	0
β_1	0.7709	0.7450	0.7254	0.7169
β_2	0.7749	0.7459	0.7283	0.7169

Table 1: Averages of estimated and optimal coefficients over 1,000 replicates.



(a) (b)

Figure 3: Estimated sampling distributions of (a) $\hat{\beta}_0$ and (b) $\hat{\beta}_1$ with the asymptotic normal densities overlaid.

4.2 Feature Selection

Clearly the results we have established have implications to statistical inferences on the SVM. Among others, feature selection is of particular interest. By using the asymptotic variability of estimated coefficients, one can derive a new feature selection criterion based on the standardized coefficients. Such a criterion will take inherent statistical variability into account. More generally, this consideration of new criteria opens the possibility of casting feature selection for the SVM formally in the framework of hypothesis testing and extending standard variable selection procedures in regression to classification.

We investigate the possibility of using the standardized coefficients of $\hat{\beta}$ for selection of variables. For practical applications, one needs to construct a reasonable nonparametric estimator of the asymptotic variance-covariance matrix, whose entries are defined through line integrals. A similar technical issue arises in quantile regression. See Koenker (2005) for some suggested variance estimators in the setting.

For the sake of simplicity in the second set of simulation, we used the theoretical asymptotic variance in standardizing $\hat{\beta}$ and selected those variables with the absolute standardized coefficient exceeding a certain critical value. And we mainly monitored the type I error rate of falsely declaring the significance of a variable when it is not, over various settings of a mixture of two multivariate normal distributions. Different combinations of the sample size (n) and the number of variables (d) were tried. For a fixed even d, we set $\mu_f = (\mathbf{1}_{d/2}, \mathbf{0}_{d/2})^\top$, $\mu_g = \mathbf{0}_d^\top$, and $\Sigma = I_d$, where $\mathbf{1}_p$ and $\mathbf{0}_p$ indicate p-vectors of ones and zeros, respectively. Thus only the first half of the d variables have nonzero coefficients in the optimal hyperplane of the linear SVM. Table 2 shows the minima, median, and maxima of such type I error rates in selection of relevant variables over 200 replicates when the critical value was $z_{0.025} \approx 1.96$ (5% level of significance). If the asymptotic distributions were accurate, the error rates would be close to the nominal level of 0.05. On the whole, the table suggests that when d is small, the error rates are very close to the nominal level even for small sample sizes, while for a large d, n has to be quite large for the asymptotic distributions to be valid. This pattern is clearly seen in Figure 4, which displays the median values of the type I error rates. In passing, we note that changing the proportion of relevant variables did not seem to affect the error rates, which are not shown here.

	Number of variables (d)			
n	6	12	18	24
250	[0.050, 0.060, 0.090]	[0.075, 0.108, 0.145]	[0.250, 0.295, 0.330]	[0.665, 0.698, 0.720]
500	[0.045, 0.080, 0.090]	[0.040, 0.068, 0.095]	[0.105, 0.130, 0.175]	[0.275, 0.293, 0.335]
750	[0.030, 0.055, 0.070]	[0.035, 0.065, 0.090]	[0.055, 0.095, 0.115]	[0.135, 0.185, 0.205]
1000	[0.050 ,0.065, 0.065]	[0.060, 0.068, 0.095]	[0.040, 0.075, 0.095]	[0.105, 0.135, 0.175]
1250	[0.065, 0.065, 0.070]	[0.035, 0.045, 0.050]	[0.055, 0.080, 0.105]	[0.070, 0.095, 0.125]
1500	[0.035, 0.050, 0.065]	[0.040, 0.058, 0.085]	[0.055, 0.075, 0.090]	[0.050, 0.095, 0.135]
1750	[0.030, 0.035, 0.060]	[0.035, 0.045, 0.075]	[0.040, 0.065, 0.095]	[0.055, 0.080, 0.120]
2000	[0.035, 0.040, 0.060]	[0.040, 0.065, 0.080]	[0.060, 0.070, 0.100]	[0.055, 0.075, 0.105]

Table 2: The minimum, median, and maximum values of the type I error rates of falsely flagging an irrelevant variable as relevant over 200 replicates by using the standardized SVM coefficients at 5% significance level.

We leave further development of asymptotic variance estimators for feature selection and comparison with risk based approaches such as the recursive feature elimination procedure as a future work.

5. Discussion

In this paper, we have investigated asymptotic properties of the coefficients of variables in the SVM solution for nonseparable linear classification. More specifically, we have established a Bahadur

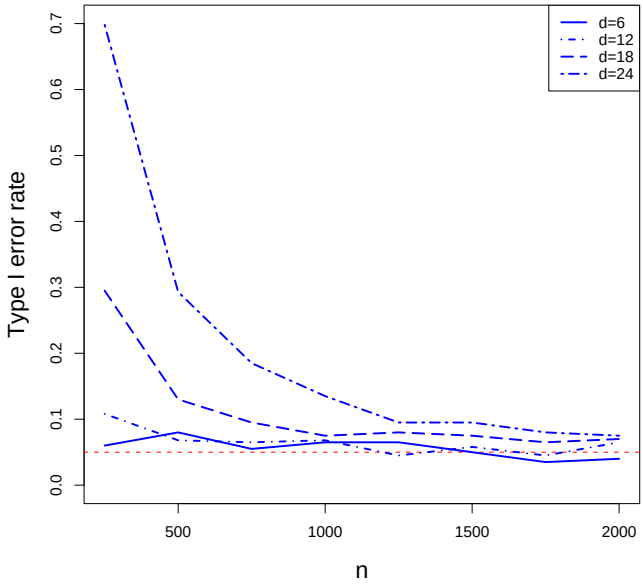


Figure 4: The median values of the type I error rates in variable selection depending on the sample size n and the number of variables d. The horizontal, dashed line indicates the nominal level of 0.05.

type representation of the coefficients and their asymptotic normality using Radon transformation of the second moments of the variables. The representation shows how the statistical behavior of the coefficients is determined by the margins of the optimal hyperplane and the underlying probability distribution. Shedding a new statistical light on the SVM, these results provide an insight into its asymptotic behavior and can be used to improve our statistical practice with the SVM in various aspects.

There are several issues yet to be investigated. The asymptotic results that we have obtained so far pertain only to the linear SVM in nonseparable cases. Although it may be of more theoretical consideration than practical, a similar analysis of the linear SVM in the separable case is anticipated, which will ultimately lead to a unified theory for separable as well as nonseparable cases. The separable case would require a slightly different treatment than the nonseparable case because the regularization parameter λ needs to remain positive in the limit to guarantee the uniqueness of the solution. An extension of the SVM asymptotics to the nonlinear case is another direction of interest. In this case, the minimizer defined by the SVM is not a vector of coefficients of a fixed length but a function in a reproducing kernel Hilbert space. So, the study of asymptotic properties of the minimizer in the function space essentially requires investigation of its pointwise behavior or its functionals in general as the sample size grows. A general theory in Shen (1997) on asymptotic normality and efficiency of substitution estimates for smooth functionals is relevant. In particular, Theorem 2 in Shen (1997) provides the asymptotic normality of the penalized sieve MLE, char-

acterization of which bears a close resemblance with function estimation for the nonlinear case. However, the theory was developed under the assumption of the differentiability of the loss, and it has to be modified for proper theoretical analysis of the SVM. As in the approach for the linear case presented in this paper, one may get around the non-differentiability issue of the hinge loss by imposing appropriate regularity conditions to ensure that the minimizer is unique and the expected loss is differentiable and locally quadratic around the minimizer.

Consideration of these extensions will lead to a more complete picture of the asymptotic behavior of the SVM solution.

6. Proofs

In this section, we present technical lemmas and prove the main results.

6.1 Technical Lemmas

Lemma 1 shows that there is a finite minimizer of $L(\beta)$, which is useful in proving the uniqueness of the minimizer in Lemma 6. In fact, the existence of the first moment of X is sufficient for Lemmas 1, 2, and 4. However, (A1) is needed for the existence and continuity of $H(\beta)$ in the proof of other lemmas and theorems.

Lemma 1 *Suppose that (A1) and (A2) are satisfied. Then $L(\beta) \to \infty$ as $\|\beta\| \to \infty$ and the existence of β^* is guaranteed.*

Proof. Without loss of generality, we may assume that $x_0 = 0$ in (A2) and $B(0,\delta_0) \subset \mathcal{X}$. For any $\varepsilon > 0$,

$$\begin{aligned}
L(\beta) &= \pi_+ \int_{\mathcal{X}} [1 - \tilde{x}^\top \beta]_+ f(x)dx + \pi_- \int_{\mathcal{X}} [1 + h(x;\beta)]_+ g(x)dx \\
&\geq \pi_+ \int_{\mathcal{X}} \{h(x;\beta) \leq 0\}(1 - h(x;\beta)) f(x)dx + \pi_- \int_{\mathcal{X}} \{h(x;\beta) \geq 0\}(1 + h(x;\beta)) g(x)dx \\
&\geq \pi_+ \int_{\mathcal{X}} \{h(x;\beta) \leq 0\}(-h(x;\beta)) f(x)dx + \pi_- \int_{\mathcal{X}} \{h(x;\beta) \geq 0\} h(x;\beta) g(x)dx \\
&\geq \int_{\mathcal{X}} |h(x;\beta)| \min(\pi_+ f(x), \pi_- g(x))\, dx \\
&\geq C_1 \min(\pi_+, \pi_-) \int_{B(0,\delta_0)} |h(x;\beta)| dx \\
&= C_1 \min(\pi_+, \pi_-) \|\beta\| \int_{B(0,\delta_0)} |h(x;w)| dx \\
&\geq C_1 \min(\pi_+, \pi_-) \|\beta\| \mathrm{vol}(\{|h(x;w)| \geq \varepsilon\} \cap B(0,\delta_0))\, \varepsilon,
\end{aligned}$$

where $w = \beta/\|\beta\|$ and $\mathrm{vol}(A)$ denotes the volume of a set A.

Note that $-1 \le w_0 \le 1$. For $0 \le w_0 < 1$ and $0 < \varepsilon < 1$,

$$\begin{aligned} & \mathrm{vol}\left(\{|h(x;w)| \ge \varepsilon\} \cap B(0,\delta_0)\right) \\ \ge\ & \mathrm{vol}\left(\{h(x;w) \ge \varepsilon\} \cap B(0,\delta_0)\right) \\ =\ & \mathrm{vol}\left(\left\{x^\top w_+ / \sqrt{1-w_0^2} \ge (\varepsilon - w_0)/\sqrt{1-w_0^2}\right\} \cap B(0,\delta_0)\right) \\ \ge\ & \mathrm{vol}\left(\left\{x^\top w_+ / \sqrt{1-w_0^2} \ge \varepsilon\right\} \cap B(0,\delta_0)\right) \equiv V(\delta_0,\varepsilon) \end{aligned}$$

since $(\varepsilon - w_0)/\sqrt{1-w_0^2} \le \varepsilon$. When $-1 < w_0 < 0$, we obtain

$$\mathrm{vol}_B(h(x;w) \le -\varepsilon) \ge V(\delta_0,\varepsilon)$$

in a similar way. Note that $V(\delta_0,\varepsilon)$ is independent of β and $V(\delta_0,\varepsilon) > 0$ for some $\varepsilon < \delta_0$. Consequently, $L(\beta) \ge C_1 \min(\pi_+,\pi_-)\|\beta\| V(\delta_0,\varepsilon)\varepsilon \to \infty$ as $\|\beta\| \to \infty$. The case $w_0 = \pm 1$ is trivial.

Since the hinge loss is convex, $L(\beta)$ is convex in β. Since $L(\beta) \to \infty$ as $\|\beta\| \to \infty$, the set, denoted by $\mathcal{M}$, of minimizers of $L(\beta)$ forms a bounded connected set. The existence of the solution β^* of $L(\beta)$ easily follows from this. ■

In Lemmas 2 and 3, we obtain explicit forms of $S(\beta)$ and $H(\beta)$ for non-constant decision functions.

Lemma 2 *Assume that (A1) is satisfied. If $\beta_+ \ne 0$, then we have*

$$\frac{\partial L(\beta)}{\partial \beta_j} = S(\beta)_j$$

for $0 \le j \le d$.

Proof. It suffices to show that

$$\frac{\partial}{\partial \beta_j} \int_{\mathcal{X}} [1 - h(x;\beta)]_+ f(x)dx = -\int_{\mathcal{X}} \{h(x;\beta) \le 1\} \widetilde{x}_j f(x)dx.$$

Define $\Delta(t) = [1 - h(x;\beta) - t\widetilde{x}_j]_+ - [1 - h(x;\beta)]_+$. Let $t > 0$.

First, consider the case $\widetilde{x}_j > 0$. Then,

$$\Delta(t) = \begin{cases} 0 & \text{if} \quad h(x;\beta) > 1 \\ h(x;\beta) - 1 & \text{if} \quad 1 - t\widetilde{x}_j < h(x;\beta) \le 1 \\ -t\widetilde{x}_j & \text{if} \quad h(x;\beta) \le 1 - t\widetilde{x}_j. \end{cases}$$

Observe that

$$\begin{aligned} \int_{\mathcal{X}} \Delta(t)\{\widetilde{x}_j > 0\} f(x)dx \ =\ & \int_{\mathcal{X}} \{1 - t\widetilde{x}_j < h(x;\beta) \le 1\}(h(x;\beta) - 1) f(x)dx \\ & -t\int_{\mathcal{X}} \{h(x;\beta) \le 1 - t\widetilde{x}_j, \widetilde{x}_j > 0\} \widetilde{x}_j f(x)dx \end{aligned}$$

and that

$$\left| \frac{1}{t} \int_{\mathcal{X}} \{1 - t\widetilde{x}_j < h(x;\beta) \le 1\}(h(x;\beta) - 1) f(x)dx \right| \le \int_{\mathcal{X}} \{1 - t\widetilde{x}_j < h(x;\beta) \le 1\} \widetilde{x}_j f(x)dx.$$

By Dominated Convergence Theorem,

$$\lim_{t\downarrow 0}\int_{\mathcal{X}}\{1-t\widetilde{x}_j < h(x;\beta)\le 1\}\widetilde{x}_j f(x)dx = \int_{\mathcal{X}}\{h(x;\beta)=1\}\widetilde{x}_j f(x)dx = 0$$

and

$$\lim_{t\downarrow 0}\int_{\mathcal{X}}\{h(x;\beta)\le 1-t\widetilde{x}_j,\widetilde{x}_j>0\}\widetilde{x}_j f(x)dx = \int_{\mathcal{X}}\{h(x;\beta)\le 1,\widetilde{x}_j>0\}\widetilde{x}_j f(x)dx.$$

Hence

$$\lim_{t\downarrow 0}\frac{1}{t}\int_{\mathcal{X}}\Delta(t)\{\widetilde{x}_j>0\}f(x)dx = -\int_{\mathcal{X}}\{h(x;\beta)\le 1,\widetilde{x}_j>0\}\widetilde{x}_j f(x)dx. \tag{9}$$

Now assume that $\widetilde{x}_j<0$. Then,

$$\Delta(t)=\begin{cases} 0 & \text{if} \quad h(x;\beta)>1-t\widetilde{x}_j \\ 1-h(x;\beta)-t\widetilde{x}_j & \text{if} \quad 1<h(x;\beta)\le 1-t\widetilde{x}_j \\ -t\widetilde{x}_j & \text{if} \quad h(x;\beta)\le 1. \end{cases}$$

In a similar fashion, one can show that

$$\lim_{t\downarrow 0}\frac{1}{t}\int_{\mathcal{X}}\Delta(t)\{\widetilde{x}_j<0\}f(x)dx = -\int_{\mathcal{X}}\{h(x;\beta)\le 1,\widetilde{x}_j<0\}\widetilde{x}_j f(x)dx. \tag{10}$$

Combining (9) and (10), we have shown that

$$\lim_{t\downarrow 0}\frac{1}{t}\int_{\mathcal{X}}\Delta(t)f(x)dx = -\int_{\mathcal{X}}\{h(x;\beta)\le 1\}\widetilde{x}_j f(x)dx.$$

The proof for the case $t<0$ is similar. ■

The proof of Lemma 3 is based on the following identity

$$\int\delta(Dt+E)T(t)dt = \frac{1}{|D|}T(-E/D) \tag{11}$$

for constants D and E. This identity follows from the fact that $\delta(at)=\delta(t)/|a|$ and $\int\delta(t-a)T(t)dt = T(a)$ for a constant a.

Lemma 3 *Suppose that (A1) is satisfied. Under the condition that $\beta_+\neq 0$, we have*

$$\frac{\partial^2 L(\beta)}{\partial\beta_j\partial\beta_k} = H(\beta)_{jk},$$

for $0\le j,k\le d$.

Proof. Define

$$\Psi(\beta)=\int_{\mathcal{X}}\{x^\top\beta_+<1-\beta_0\}s(x)dx$$

for a continuous and integrable function s defined on $\mathcal{X}$. Without loss of generality, we may assume that $\beta_1\neq 0$. It is sufficient to show that for $0\le j,k\le d$,

$$\frac{\partial^2}{\partial\beta_j\partial\beta_k}\int_{\mathcal{X}}[1-h(x;\beta)]_+ f(x)dx = \int_{\mathcal{X}}\delta(1-h(x;\beta))\widetilde{x}_j\widetilde{x}_k f(x)dx.$$

Define $\mathcal{X}_{-j} = \{(x_1,\ldots,x_{j-1},x_{j+1},\ldots,x_d) \,:\, (x_1,\ldots,x_d) \in \mathcal{X}\}$ and $\mathcal{X}_j = \{x_j \,:\, (x_1,\ldots,x_d) \in \mathcal{X}\}$. Observe that

$$\frac{\partial \Psi(\beta)}{\partial \beta_0} = -\frac{1}{|\beta_1|}\int_{\mathcal{X}_{-1}} s\left(\frac{1-h(x;\beta)+\beta_1 x_1}{\beta_1}, x_2,\ldots,x_d\right) dx_{-1} \tag{12}$$

and that for $k \neq 1$,

$$\frac{\partial \Psi(\beta)}{\partial \beta_k} = -\frac{1}{|\beta_1|}\int_{\mathcal{X}_{-1}} x_k s\left(\frac{1-h(x;\beta)+\beta_1 x_1}{\beta_1}, x_2,\ldots,x_d\right) dx_{-1}. \tag{13}$$

If $\beta_p = 0$ for any $p \neq 1$, then

$$\frac{\partial \Psi(\beta)}{\partial \beta_1} = -\frac{1-\beta_0}{\beta_1|\beta_1|}\int_{\mathcal{X}_{-1}} s\left(\frac{1-\beta_0}{\beta_1}, x_2,\ldots,x_d\right) dx_{-1}. \tag{14}$$

If there is $p \neq 1$ with $\beta_p \neq 0$, then we have

$$\frac{\partial \Psi(\beta)}{\partial \beta_1} = -\frac{1}{|\beta_p|}\int_{\mathcal{X}_{-p}} x_1 s\left(x_1,\ldots,x_{p-1}, \frac{1-h(x;\beta)+\beta_p x_p}{\beta_p}, x_{p+1},\ldots,x_d\right) dx_{-p}. \tag{15}$$

Choose $D = \beta_1$, $E = \widetilde{x}^\top\beta - \beta_1 x_1 - 1$, $t = x_1$ in (11). It follows from (13) and (15) that

$$\begin{aligned}
\frac{\partial \Psi(\beta)}{\partial \beta_k} &= -\frac{1}{|\beta_1|}\int_{\mathcal{X}_{-1}} x_k s\left(\frac{1-h(x;\beta)+\beta_1 x_1}{\beta_1}, x_2,\ldots,x_d\right) dx_{-1} \\
&= -\int_{\mathcal{X}_{-1}}\int_{\mathcal{X}_1} x_k s(x)\delta(h(x;\beta)-1)dx_1 dx_{-1} \\
&= -\int_{\mathcal{X}} \delta(1-h(x;\beta)) x_k s(x) dx.
\end{aligned} \tag{16}$$

Similarly, we have

$$\frac{\partial \Psi(\beta)}{\partial \beta_0} = -\int_{\mathcal{X}} \delta(1-h(x;\beta)) s(x) dx \tag{17}$$

by (12).

Choosing $D = \beta_1$, $E = \beta_0 - 1$, $t = x_1$ in (11), we have

$$\int_{\mathcal{X}_1} \delta(\beta_1 x_1 - 1 + \beta_0) x_1 s(x) dx_1 = \frac{1}{|\beta_1|}\left(\frac{1-\beta_0}{\beta_1}\right) s\left(\frac{1-\beta_0}{\beta_1}, x_2,\ldots,x_d\right)$$

by (14). This implies (16) for $k = 1$. The desired result now follows from (16) and (17). ■

The following lemma asserts that the optimal decision function is not a constant under the condition that the centers of two classes are separated.

Lemma 4 *Suppose that (A1) is satisfied. Then (A3) implies that* $\beta_+^* \neq 0$.

Proof. Suppose that

$$\int_{\mathcal{X}} \{x_{i^*} \geq G_{i^*}^-\} x_{i^*} g(x) dx < \int_{\mathcal{X}} \{x_{i^*} \leq F_{i^*}^+\} x_{i^*} f(x) dx \tag{18}$$

in (A3). We will show that

$$\min_{\beta_0} L(\beta_0, 0, \ldots 0) > \min_{\beta_0, \beta_{i^*} > 0} L(\beta_0, 0, \ldots, 0, \beta_{i^*}, 0, \ldots, 0), \tag{19}$$

implying that $\beta_+^* \neq 0$. Henceforth, we will suppress β's that are equal to zero in $L(\beta)$. The population minimizer $(\beta_0^*, \beta_{i^*}^*)$ is given by the minimizer of

$$L(\beta_0, \beta_{i^*}) = \pi_+ \int_{\mathcal{X}} [1 - \beta_0 - \beta_{i^*} x_{i^*}]_+ f(x)dx + \pi_- \int_{\mathcal{X}} [1 + \beta_0 + \beta_{i^*} x_{i^*}]_+ g(x)dx.$$

First, consider the case $\beta_{i^*} = 0$.

$$L(\beta_0) = \begin{cases} \pi_-(1+\beta_0), & \beta_0 > 1 \\ 1 + (\pi_- - \pi_+)\beta_0, & -1 \leq \beta_0 \leq 1 \\ \pi_+(1-\beta_0), & \beta_0 < -1 \end{cases}$$

with its minimum

$$\min_{\beta_0} L(\beta_0) = 2\min(\pi_+, \pi_-). \tag{20}$$

Now consider the case $\beta_{i^*} > 0$, where

$$\begin{aligned} L(\beta_0, \beta_{i^*}) = {} & \pi_+ \int_{\mathcal{X}} \left\{ x_{i^*} \leq \frac{1-\beta_0}{\beta_{i^*}} \right\} (1 - \beta_0 - \beta_{i^*} x_{i^*}) f(x)dx \\ & + \pi_- \int_{\mathcal{X}} \left\{ x_{i^*} \geq \frac{-1-\beta_0}{\beta_{i^*}} \right\} (1 + \beta_0 + \beta_{i^*} x_{i^*}) g(x)dx. \end{aligned}$$

Let $\widetilde{\beta}_0$ denote the minimizer of $L(\beta_0, \beta_{i^*})$ for a given β_{i^*}. Note that $\partial L(\beta_0, \beta_{i^*})/\partial \beta_0$ is given as

$$\begin{aligned} & \frac{\partial L(\beta_0, \beta_{i^*})}{\partial \beta_0} \\ = {} & -\pi_+ \int_{\mathcal{X}} \left\{ x_{i^*} \leq \frac{1-\beta_0}{\beta_{i^*}} \right\} f(x)dx + \pi_- \int_{\mathcal{X}} \left\{ x_{i^*} \geq \frac{-1-\beta_0}{\beta_{i^*}} \right\} g(x)dx, \end{aligned} \tag{21}$$

which is monotonic increasing in β_0 with $\lim_{\beta_0 \to -\infty} \frac{\partial L(\beta_0, \beta_{i^*})}{\partial \beta_0} \to -\pi_+$ and $\lim_{\beta_0 \to \infty} \frac{\partial L(\beta_0, \beta_{i^*})}{\partial \beta_0} \to \pi_-$. Hence $\widetilde{\beta}_0$ exists for a given β_{i^*}.

When $\pi_- < \pi_+$, we can easily check that $F_{i^*}^+ < \infty$ and $G_{i^*}^- = -\infty$. $F_{i^*}^+$ and $G_{i^*}^-$ may not be determined uniquely, meaning that there may exist an interval with probability zero. There is no significant change in the proof under the assumption that $F_{i^*}^+$ and $G_{i^*}^-$ are unique. Note that

$$\frac{1 - \widetilde{\beta}_0}{\beta_{i^*}} \leq F_{i^*}^+$$

by definition of $F_{i^*}^+$ and (21). Then,

$$\frac{-1 - \widetilde{\beta}_0}{\beta_{i^*}} \leq F_{i^*}^+ - \frac{2}{\beta_{i^*}} \to -\infty \text{ as } \beta_{i^*} \to 0,$$

and

$$\frac{1 - \widetilde{\beta}_0}{\beta_{i^*}} \to F_{i^*}^+ \text{ as } \beta_{i^*} \to 0.$$

¿From (18),

$$\pi_- \int_{\mathcal{X}} x_{i^*} g(x)dx < \pi_+ \int_{\mathcal{X}} \{x_{i^*} \leq F_{i^*}^+\} x_{i^*} f(x)dx. \tag{22}$$

Now consider the minimum of $L(\widetilde{\beta}_0, \beta_{i^*})$ with respect to $\beta_{i^*} > 0$. From (21),

$$\begin{aligned} & L(\widetilde{\beta}_0, \beta_{i^*}) \qquad (23) \\ = \; & \pi_+ \int_{\mathcal{X}} \left\{ x_{i^*} \le \frac{1-\widetilde{\beta}_0}{\beta_{i^*}} \right\} (1 - \widetilde{\beta}_0 - \beta_{i^*} x_{i^*}) f(x) dx \\ & + \pi_- \int_{\mathcal{X}} \left\{ x_{i^*} \ge \frac{-1-\widetilde{\beta}_0}{\beta_{i^*}} \right\} (1 + \widetilde{\beta}_0 + \beta_{i^*} x_{i^*}) g(x) dx \\ = \; & 2\pi_- \int_{\mathcal{X}} \left\{ x_{i^*} \ge \frac{-1-\widetilde{\beta}_0}{\beta_{i^*}} \right\} g(x) dx \\ & + \beta_{i^*} \left(\pi_- \int_{\mathcal{X}} \left\{ x_{i^*} \ge \frac{-1-\widetilde{\beta}_0}{\beta_{i^*}} \right\} x_{i^*} g(x) dx - \pi_+ \int_{\mathcal{X}} \left\{ x_{i^*} \le \frac{1-\widetilde{\beta}_0}{\beta_{i^*}} \right\} x_{i^*} f(x) dx \right) \end{aligned}$$

By (22), it can be easily seen that the second term in (23) is negative for sufficiently small $\beta_{i^*} > 0$, implying

$$L(\widetilde{\beta}_0, \beta_{i^*}) < 2\pi_- \quad \text{for some } \beta_{i^*} > 0.$$

If $\pi_- > \pi_+$, then $F_{i^*}^+ = \infty$ and $G_{i^*}^- > -\infty$. Similarly, it can be checked that

$$L(\widetilde{\beta}_0, \beta_{i^*}) < 2\pi_+ \quad \text{for some } \beta_{i^*} > 0.$$

Suppose that $\pi_- = \pi_+$. Then it can be verified that

$$\frac{1-\widetilde{\beta}_0}{\beta_{i^*}} \to \infty \text{ as } \beta_{i^*} \to 0,$$

and

$$\frac{-1-\widetilde{\beta}_0}{\beta_{i^*}} \to -\infty \text{ as } \beta_{i^*} \to 0.$$

In this case, $L(\widetilde{\beta}_0, \beta_{i^*}) < 1$.

Hence, under (18), we have shown that

$$L(\widetilde{\beta}_0, \beta_{i^*}) < 1 \quad \text{for some } \beta_{i^*} > 0.$$

This, together with (20), implies (19). For the second case in (A3), the same arguments hold with $\beta_{i^*} < 0$. ■

Note that (A1) implies that H is well-defined and continuous in its argument. (A4) ensures that $H(\beta)$ is positive definite around β^* and thus we have a lower bound result in Lemma 5.

Lemma 5 *Under (A1), (A3) and (A4),*

$$\beta^\top H(\beta^*)\beta \ge C_4 \|\beta\|^2,$$

where C_4 may depend on β^*.

Proof. Since the proof for the case $d = 1$ is trivial, we consider the case $d \geq 2$ only. Observe that

$$\begin{aligned}
\beta^\top H(\beta^*)\beta &= \mathbb{E}\Big(\delta(1 - Yh(X;\beta^*))h^2(X;\beta)\Big) \\
&= \pi_+ \int_{\mathcal{X}} \delta(1 - h(x;\beta^*))h^2(x;\beta) f(x)dx + \pi_- \int_{\mathcal{X}} \delta(1 + h(x;\beta^*))h^2(x;\beta)g(x)dx \\
&= \pi_+(\mathcal{R}h^2 f)(1 - \beta_0^*, \beta_+^*) + \pi_-(\mathcal{R}h^2 g)(1 + \beta_0^*, -\beta_+^*) \\
&= \pi_+(\mathcal{R}h^2 f)(1 - \beta_0^*, \beta_+^*) + \pi_-(\mathcal{R}h^2 g)(-1 - \beta_0^*, \beta_+^*).
\end{aligned}$$

The last equality follows from the homogeneity of Radon transform.

Recall that $\beta_+^* \neq 0$ by Lemma 4. Without loss of generality, we assume that $j^* = 1$ in (A4). Let $A_1\beta_+ = a = (a_1, \ldots, a_d)$ and $z = (A_1 x)/\|\beta_+^*\|$. Given $u = (u_2, \ldots, u_d)$, let $u^* = \Big((1 - \beta_0^*)/\|\beta_+^*\|, u\Big)$. Define $\mathcal{Z} = \{z = (A_1 x)/\|\beta_+^*\| : x \in \mathcal{X}\}$ and $\mathcal{U} = \{u : u_j = \|\beta_+^*\| z_j \text{ for } j = 2, \ldots, d, \text{ and } z \in \mathcal{Z}\}$. Note that $\det A_1 = 1$, $du = \|\beta_+^*\|^{d-1} dz_2 \ldots dz_d$,

$$\|\beta_+^*\| A_1^\top z\Big|_{z_1 = (1-\beta_0^*)/\|\beta_+^*\|^2} = A_1^\top \begin{pmatrix} (1 - \beta_0^*)/\|\beta_+^*\| \\ \|\beta_+^*\| z_2 \\ \vdots \\ \|\beta_+^*\| z_d \end{pmatrix} = A_1^\top u^*$$

and

$$\begin{aligned}
\beta_0 + \|\beta_+^*\| a^\top z\Big|_{z_1 = (1-\beta_0^*)/\|\beta_+^*\|^2} &= \beta_0 + \|\beta_+^*\| \Big(a_1(1 - \beta_0^*)/\|\beta_+^*\|^2 + \sum_{j=2}^d a_j z_j\Big) \\
&= \beta_0 + a_1(1 - \beta_0^*)/\|\beta_+^*\| + \|\beta_+^*\| \sum_{j=2}^d a_j z_j.
\end{aligned}$$

Using the transformation A_1, we have

$$\begin{aligned}
& (\mathcal{R}h^2 f)(1 - \beta_0^*, \beta_+^*) \\
= & \int_{\mathcal{Z}} \delta\left(1 - \beta_0^* - \|\beta_+^*\|(A_1\beta_+^*)^\top z\right) h^2\left(\|\beta_+^*\| A_1^\top z; \beta\right) f\left(\|\beta_+^*\| A_1^\top z\right) \|\beta_+^*\|^d dz \\
= & \int_{\mathcal{Z}} \delta\left(1 - \beta_0^* - \|\beta_+^*\|^2 e_1^\top z\right) \left(\beta_0 + \|\beta_+^*\|(A_1\beta_+)^\top z\right)^2 f\left(\|\beta_+^*\| A_1^\top z\right) \|\beta_+^*\|^d dz \\
= & \int_{\mathcal{Z}} \delta\left(1 - \beta_0^* - \|\beta_+^*\|^2 z_1\right) \left(\beta_0 + \|\beta_+^*\| a^\top z\right)^2 f\left(\|\beta_+^*\| A_1^\top z\right) \|\beta_+^*\|^d dz \\
= & \frac{1}{\|\beta_+^*\|} \int_{\mathcal{U}} \Big(\beta_0 + a_1(1 - \beta_0^*)/\|\beta_+^*\| + \sum_{j=2}^d a_j u_j\Big)^2 f(A_1^\top u^*) du.
\end{aligned}$$

The last equality follows from the identity (11). Let $\mathcal{D}_*^+ = \{u : A_1^\top u^* \in \mathcal{D}^+\}$. By (A4), there exists a constant $C_2 > 0$ and a rectangle $\mathcal{D}_*^+$ on which $f(A_1^\top u^*) \geq C_2$ for $u \in \mathcal{D}_*^+$. Then

$$\begin{aligned}
& (\mathcal{R}h^2 f)(1-\beta_0^*, \beta_+^*) \\
\geq\ & \frac{1}{\|\beta_+^*\|} \int_{\mathcal{D}_*^+} \Big(\beta_0 + a_1(1-\beta_0^*)/\|\beta_+^*\| + \sum_{j=2}^d a_j u_j\Big)^2 f(A_1^\top u^*) du \\
\geq\ & \frac{1}{\|\beta_+^*\|} \cdot C_2 \int_{\mathcal{D}_*^+} \Big(\beta_0 + a_1(1-\beta_0^*)/\|\beta_+^*\| + \sum_{j=2}^d a_j u_j\Big)^2 du \\
=\ & \frac{1}{\|\beta_+^*\|} \cdot C_2 \cdot \mathrm{vol}(\mathcal{D}_*^+) \mathbb{E}^u \Big(\beta_0 + a_1(1-\beta_0^*)/\|\beta_+^*\| + \sum_{j=2}^d a_j U_j\Big)^2 \\
=\ & \frac{1}{\|\beta_+^*\|} \cdot C_2 \cdot \mathrm{vol}(\mathcal{D}_*^+) \Big\{ \Big(\beta_0 + a_1(1-\beta_0^*)/\|\beta_+^*\| + \mathbb{E}^u \sum_{j=2}^d a_j U_j\Big)^2 + \mathbb{V}^u\Big(\sum_{j=2}^d a_j U_j\Big)\Big\},
\end{aligned}$$

where U_j for $j = 2, \ldots, d$ are independent and uniform random variables defined on $\mathcal{D}_*^+$, and $\mathbb{E}^u$ and $\mathbb{V}^u$ denote the expectation and variance with respect to the uniform distribution.

Letting $m_i = (l_i + v_i)/2$, we have

$$\begin{aligned}
& (\mathcal{R}h^2 f)(1-\beta_0^*, \beta_+^*) \qquad (24) \\
\geq\ & \frac{1}{\|\beta_+^*\|} \cdot C_2 \cdot \mathrm{vol}(\mathcal{D}_*^+) \Big\{ \Big(\beta_0 + a_1(1-\beta_0^*)/\|\beta_+^*\| + \sum_{j=2}^d a_j m_j\Big)^2 + \min_{2\leq j\leq d} \mathbb{V}^u(U_j) \sum_{j=2}^d a_j^2 \Big\}.
\end{aligned}$$

Similarly, it can be shown that

$$\begin{aligned}
& (\mathcal{R}h^2 g)(-1-\beta_0^*, \beta_+^*) \qquad (25) \\
\geq\ & \frac{1}{\|\beta_+^*\|} \cdot C_3 \cdot \mathrm{vol}(\mathcal{D}_*^-) \Big\{ \Big(\beta_0 - a_1(1+\beta_0^*)/\|\beta_+^*\| + \sum_{j=2}^d a_j m_j\Big)^2 + \min_{2\leq j\leq d} \mathbb{V}^u(U_j) \sum_{j=2}^d a_j^2 \Big\},
\end{aligned}$$

where $\mathcal{D}_*^- = \Big\{u : A_1^\top \Big((-1-\beta_0^*)/\|\beta_+^*\|, u\Big) \in \mathcal{D}^-\Big\}$.

Combining (24)-(25) and letting $C_5 = 2/\|\beta_+^*\| \min\Big(\pi_+ C_2 \cdot \mathrm{vol}(\mathcal{D}_*^+), \pi_- C_3 \cdot \mathrm{vol}(\mathcal{D}_*^-)\Big)$ and $C_6 = \min\Big(1, \min_{2\le j\le d} \mathbb{V}^u(U_j)\Big)$, we have

$$\begin{aligned}
& \beta^\top H(\beta^*)\beta \\
\geq\ & \frac{\pi_+}{\|\beta_+^*\|} \cdot C_2 \cdot \mathrm{vol}(\mathcal{D}_*^+)\Big\{\Big(\beta_0 + a_1(1-\beta_0^*)/\|\beta_+^*\| + \sum_{j=2}^d a_j m_j\Big)^2 + \min_{2\le j\le d} \mathbb{V}^u(U_j) \sum_{j=2}^d a_j^2\Big\} \\
& + \frac{\pi_-}{\|\beta_+^*\|} \cdot C_3 \cdot \mathrm{vol}(\mathcal{D}_*^-)\Big\{\Big(\beta_0 - a_1(1+\beta_0^*)/\|\beta_+^*\| + \sum_{j=2}^d a_j m_j\Big)^2 + \min_{2\le j\le d} \mathbb{V}^u(U_j) \sum_{j=2}^d a_j^2\Big\} \\
\geq\ & C_5 C_6\Big\{\Big(\beta_0 + a_1(1-\beta_0^*)/\|\beta_+^*\| + \sum_{j=2}^d a_j m_j\Big)^2 \\
& + \Big(\beta_0 - a_1(1+\beta_0^*)/\|\beta_+^*\| + \sum_{j=2}^d a_j m_j\Big)^2 + 2\sum_{j=2}^d a_j^2\Big\}/2 \\
=\ & C_5 C_6\Big\{\Big(\beta_0 + a_1(1-\beta_0^*)/\|\beta_+^*\|\Big)^2 + \Big(\beta_0 - a_1(1+\beta_0^*)/\|\beta_+^*\|\Big)^2 \\
& + 4\Big(\beta_0 - a_1\beta_0^*/\|\beta_+^*\|\Big)\sum_{j=2}^d a_j m_j + 2\Big(\sum_{j=2}^d a_j m_j\Big)^2 + 2\sum_{j=2}^d a_j^2\Big\}/2.
\end{aligned}$$

Note that

$$\begin{aligned}
& \Big(\sum_{j=2}^d a_j m_j\Big)^2 + 2\Big(\beta_0 - a_1\beta_0^*/\|\beta_+^*\|\Big)\sum_{j=2}^d a_j m_j \\
=\ & \Big(\sum_{j=2}^d a_j m_j + \beta_0 - a_1\beta_0^*/\|\beta_+^*\|\Big)^2 - \Big(\beta_0 - a_1\beta_0^*/\|\beta_+^*\|\Big)^2
\end{aligned}$$

and

$$\Big(\beta_0 + a_1(1-\beta_0^*)/\|\beta_+^*\|\Big)^2 + \Big(\beta_0 - a_1(1+\beta_0^*)/\|\beta_+^*\|\Big)^2 - 2\Big(\beta_0 - a_1\beta_0^*/\|\beta_+^*\|\Big)^2 = 2a_1^2/\|\beta_+^*\|^2.$$

Thus, the lower bound of $\beta^\top H(\beta^*)\beta$ except for the constant C_5C_6 allows the following quadratic form in terms of $\beta_0, a_1, \ldots, a_d$. Let

$$Q(\beta_0, a_1, \ldots, a_d) = a_1^2/\|\beta_+^*\|^2 + \Big(\sum_{j=2}^d a_j m_j + \beta_0 - a_1\beta_0^*/\|\beta_+^*\|\Big)^2 + \sum_{j=2}^d a_j^2.$$

Obviously $Q(\beta_0, a_1, \ldots, a_d) \ge 0$ and $Q(\beta_0, a_1, \ldots, a_d) = 0$ implies that $a_1 = \ldots = a_d = 0$ and $\beta_0 = 0$. Therefore Q is positive definite. Letting $\nu_1 > 0$ be the smallest eigenvalue of the matrix corresponding to Q, we have proved

$$\beta^\top H(\beta^*)\beta \quad \geq \quad C_5C_6\nu_1(\beta_0^2 + \sum_{j=1}^d a_j^2) = C_5C_6\nu_1(\beta_0^2 + \sum_{j=1}^d \beta_j^2).$$

The last equality follows from the fact that $A_1\beta_+ = a$ and the transformation A_1 preserves the norm. With the choice of $C_4 = C_5C_6\nu_1 > 0$, the result follows. ■

Lemma 6 *Suppose that (A1)-(A4) are met. Then $L(\beta)$ has a unique minimizer.*

Proof. By Lemma 1, we may choose any minimizer $\beta^* \in \mathcal{M}$. By Lemma 4 and 5, $H(\beta)$ is positive definite at β^*. Then $L(\beta)$ is locally strictly convex at β^*, so that $L(\beta)$ has a local minimum at β^*. Hence the minimizer of $L(\beta)$ is unique. ■

6.2 Proof of Theorems 1 and 2

For fixed $\theta \in \mathbb{R}^{d+1}$, define

$$\Lambda_n(\theta) = n\Big(l_{\lambda,n}(\beta^* + \theta/\sqrt{n}) - l_{\lambda,n}(\beta^*)\Big)$$

and

$$\Gamma_n(\theta) = \mathbb{E}\Lambda_n(\theta).$$

Observe that

$$\Gamma_n(\theta) = n\left(L(\beta^* + \theta/\sqrt{n}) - L(\beta^*)\right) + \frac{\lambda}{2}\left(\|\theta_+\|^2 + 2\sqrt{n}{\beta^*_+}^\top \theta_+\right).$$

By Taylor series expansion of L around β^*, we have

$$\Gamma_n(\theta) = \frac{1}{2}\theta^\top H(\widetilde{\beta})\theta + \frac{\lambda}{2}\left(\|\theta_+\|^2 + 2\sqrt{n}{\beta^*_+}^\top \theta_+\right),$$

where $\widetilde{\beta} = \beta^* + (t/\sqrt{n})\theta$ for some $0 < t < 1$. Define $D_{jk}(\alpha) = H(\beta^* + \alpha)_{jk} - H(\beta^*)_{jk}$ for $0 \le j,k \le d$. Since $H(\beta)$ is continuous in β, there exists $\delta_1 > 0$ such that $|D_{jk}(\alpha)| < \varepsilon_1$ if $\|\alpha\| < \delta_1$ for any $\varepsilon_1 > 0$ and all $0 \le j,k \le d$. Then, as $n \to \infty$,

$$\frac{1}{2}\theta^\top H(\widetilde{\beta})\theta = \frac{1}{2}\theta^\top H(\beta^*)\theta + o(1).$$

It is because for sufficiently large n such that $\|(t/\sqrt{n})\theta\| < \delta_1$,

$$\begin{aligned}\left|\theta^\top\Big(H(\widetilde{\beta}) - H(\beta^*)\Big)\theta\right| &\le \sum_{j,k}|\theta_j||\theta_k|\left|D_{jk}\left(\frac{t}{\sqrt{n}}\theta\right)\right| \\ &\le \varepsilon_1\sum_{j,k}|\theta_j||\theta_k| \le 2\varepsilon_1\|\theta\|^2.\end{aligned}$$

Together with the assumption that $\lambda = o(n^{-1/2})$, we have

$$\Gamma_n(\theta) = \frac{1}{2}\theta^\top H(\beta^*)\theta + o(1).$$

Define $W_n = -\sum_{i=1}^n \zeta^i Y^i \widetilde{X}^i$ where $\zeta^i = \Big\{Y^i h(X^i;\beta^*) \le 1\Big\}$. Then $\frac{1}{\sqrt{n}}W_n$ follows asymptotically $N\Big(0, nG(\beta^*)\Big)$ by central limit theorem. Note that

$$\mathbb{E}\left(\zeta^i Y^i \widetilde{X}^i\right) = 0 \quad \text{and} \quad \mathbb{E}\left(\zeta^i Y^i \widetilde{X}^i (\zeta^i Y^i \widetilde{X}^i)^\top\right) = \mathbb{E}\left(\zeta^i \widetilde{X}^i (\widetilde{X}^i)^\top\right). \tag{26}$$

Recall that β^* is characterized by $S(\beta^*)=0$ implying the first part of (26). If we define

$$R_{i,n}(\theta)=\Big[1-Y^ih(X^i;\beta^*+\theta/\sqrt{n})\Big]_+-\Big[1-Y^ih(X^i;\beta^*)\Big]_+-\zeta^iY^ih(X^i;\theta/\sqrt{n}),$$

then we see that

$$\Lambda_n(\theta)=\Gamma_n(\theta)+W_n^\top\theta/\sqrt{n}+\sum_{i=1}^n\Big(R_{i,n}(\theta)-\mathbb{E}R_{i,n}(\theta)\Big)$$

and

$$\Big|R_{i,n}(\theta)\Big|\le\Big|h(X^i;\theta)/\sqrt{n}\Big|U\left(\Big|h(X^i;\theta)/\sqrt{n}\Big|\right), \tag{27}$$

where

$$U(t)=\Big\{\Big|1-Y^ih(X^i;\beta^*)\Big|\le t\Big\}\qquad\text{for}\quad t\in\mathbb{R}.$$

To verify (27), let $\zeta=\{a\le 1\}$ and $R=[1-z]_+-[1-a]_++\zeta(z-a)$. If $a>1$, then $R=(1-z)\{z\le 1\}$; otherwise, $R=(z-1)\{z>1\}$. Hence,

$$\begin{aligned}R&=(1-z)\{a>1,z\le 1\}+(z-1)\{a<1,z>1\}\\&\le|z-a|\{a>1,z\le 1\}+|z-a|\{a<1,z>1\}\\&=|z-a|\Big(\{a>1,z\le 1\}+\{a<1,z>1\}\Big)\\&\le|z-a|\{|1-a|\le|z-a|\}.\end{aligned}\tag{28}$$

Choosing $z=Y^ih(X^i;\beta^*+\theta/\sqrt{n})$ and $a=Y^ih(X^i;\beta^*)$ in (28) yields (27).

Since cross-product terms in $\mathbb{E}(\sum_i(R_{i,n}-\mathbb{E}R_{i,n}))^2$ cancel out, we obtain from (27) that for each fixed θ,

$$\begin{aligned}\sum_{i=1}^n\mathbb{E}\left(|R_{i,n}(\theta)-\mathbb{E}R_{i,n}(\theta)|^2\right)&\le\sum_{i=1}^n\mathbb{E}\left(R_{i,n}(\theta)^2\right)\\&\le\sum_{i=1}^n\mathbb{E}\left(\left(h(X^i;\theta)/\sqrt{n}\right)^2U\left(|h(X^i;\theta)/\sqrt{n}|\right)\right)\\&\le\sum_{i=1}^n\mathbb{E}\left((1+\|X^i\|^2)\|\theta\|^2/n\,U\left(\sqrt{1+\|X^i\|^2}\|\theta\|/\sqrt{n}\right)\right)\\&=\|\theta\|^2\mathbb{E}\left((1+\|X\|^2)\,U\left(\sqrt{1+\|X\|^2}\|\theta\|/\sqrt{n}\right)\right).\end{aligned}$$

(A1) implies that $\mathbb{E}\left(\|X\|^2\right)<\infty$. Hence, for any $\varepsilon>0$, choose C_7 such that $\mathbb{E}\Big((1+\|X\|^2)\{\|X\|>C_7\}\Big)<\varepsilon/2$. Then

$$\begin{aligned}&\mathbb{E}\left((1+\|X\|^2)\,U\left(\sqrt{1+\|X\|^2}\|\theta\|/\sqrt{n}\right)\right)\\\le\;&\mathbb{E}\Big((1+\|X\|^2)\{\|X\|>C_7\}\Big)+(1+C_7^2)\mathbb{P}\left(U\left(\sqrt{1+C_7^2}\|\theta\|/\sqrt{n}\right)\right)\end{aligned}$$

By (A1), the distribution of X is not degenerate, which in turn implies that $\lim_{t\downarrow 0}\mathbb{P}(U(t))=0$. We can take a large N such that $\mathbb{P}\left(U\left(\sqrt{1+C_7^2}\|\theta\|/\sqrt{n}\right)\right)<\varepsilon/(2(1+C_7^2))$ for $n\geq N$. This proves that

$$\sum_{i=1}^{n}\mathbb{E}\left(|R_{i,n}(\theta)-\mathbb{E}R_{i,n}(\theta)|^2\right)\to 0$$

as $n\to\infty$. Thus, for each fixed θ,

$$\Lambda_n(\theta)=\frac{1}{2}\theta^\top H(\beta^*)\theta+W_n^\top\theta/\sqrt{n}+o_{\mathbb{P}}(1).$$

Let $\eta_n=-H(\beta^*)^{-1}W_n/\sqrt{n}$. By Convexity Lemma in Pollard (1991), we have

$$\Lambda_n(\theta)=\frac{1}{2}(\theta-\eta_n)^\top H(\beta^*)(\theta-\eta_n)-\frac{1}{2}\eta_n^\top H(\beta^*)\eta_n+r_n(\theta),$$

where, for each compact set K in $\mathbb{R}^{d+1}$,

$$\sup_{\theta\in K}|r_n(\theta)|\to 0\qquad\text{in probability.}$$

Because η_n converges in distribution, there exists a compact set K containing B_ε, where B_ε is a closed ball with center η_n and radius ε with probability arbitrarily close to one. Hence we have

$$\Delta_n=\sup_{\theta\in B_\varepsilon}|r_n(\theta)|\to 0\qquad\text{in probability.}\tag{29}$$

For examination of the behavior of $\Lambda_n(\theta)$ outside B_ε, consider $\theta=\eta_n+\gamma v$, with $\gamma>\varepsilon$ and v, a unit vector and a boundary point $\theta^*=\eta_n+\varepsilon v$. By Lemma 5, convexity of Λ_n, and the definition of Δ_n, we have

$$\begin{aligned}\frac{\varepsilon}{\gamma}\Lambda_n(\theta)+\left(1-\frac{\varepsilon}{\gamma}\right)\Lambda_n(\eta_n) &\geq \Lambda_n(\theta^*)\\ &\geq \frac{1}{2}(\theta^*-\eta_n)^\top H(\beta^*)(\theta^*-\eta_n)-\frac{1}{2}\eta_n^\top H(\beta^*)\eta_n-\Delta_n\\ &\geq \frac{C_4}{2}\varepsilon^2+\Lambda_n(\eta_n)-2\Delta_n,\end{aligned}$$

implying that

$$\inf_{\|\theta-\eta_n\|>\varepsilon}\Lambda_n(\theta)\geq\Lambda_n(\eta_n)+\left(\frac{C_4}{2}\varepsilon^2-2\Delta_n\right).$$

By (29), we can take Δ_n so that $2\Delta_n<C_4\varepsilon^2/4$ with probability tending to one. So the minimum of Λ_n cannot occur at any θ with $\|\theta-\eta_n\|>\varepsilon$. Hence, for each $\varepsilon>0$ and $\widehat{\theta}_{\lambda,n}=\sqrt{n}(\widehat{\beta}_{\lambda,n}-\beta^*)$,

$$\mathbb{P}\Big(\|\widehat{\theta}_{\lambda,n}-\eta_n\|>\varepsilon\Big)\to 0.$$

This completes the proof of Theorems 1 and 2. ■

Acknowledgments

The authors are grateful to Wolodymyr Madych for personal communications on Radon transform. This research was supported by a Korea Research Foundation Grant funded by the Korean government (MOEHRD, Basic Research Promotion Fund) (KRF-2005-070-C00020).

References

R. R. Bahadur. A note on quantiles in large samples. *Annals of Mathematical Statistics*, 37:577–580, 1966.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statististical Association*, 101:138–156, 2006.

G. Blanchard, O. Bousquet, and P. Massart. Statistical performance of support vector machines. *The Annals of Statistics*, 36:489–531, 2008.

P. Chaudhuri. Nonparametric estimates of regression quantiles and their local Bahadur representation. *The Annals of Statistics*, 19:760–777, 1991.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, 2000.

S. R. Deans. *The Radon Transform and Some of Its Applications*. Krieger Publishing Company, Florida, 1993.

I. Guyon and A. Elisseeff. Introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

A. B. Ishak and B. Ghattas. An efficient method for variable selection using svm-based criteria. Preprint, Institut de Mathématiques de Luminy, 2005.

R. Koenker. *Quantile Regression*. Econometric Society Monographs. Cambridge University Press, 2005.

Y. Lin. Some asymptotic properties of the support vector machine. Technical report 1029, Department of Statistics, University of Wisconsin-Madison, 2000.

Y. Lin. A note on margin-based loss functions in classification. *Statististics and Probability Letters*, 68:73–82, 2002.

F. Natterer. *The Mathematics of Computerized Tomography*. Wiley & Sons, New York, 1986.

D. Pollard. Asymptotics for least absolute deviation regression estimators. *Econometric Theory*, 7: 186–199, 1991.

A. G. Ramm and A. I. Katsevich. *The Radon Transform and Local Tomography*. CRC Press, Boca Raton, 1996.

B. Schölkopf and A. Smola. *Learning with Kernels - Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.

J. C. Scovel and I. Steinwart. Fast rates for support vector machines using gaussian kernels. *The Annals of Statistics*, 35:575–607, 2007.

X. Shen. On methods of sieves and penalization. *The Annals of Statistics*, 25:2555–2591, 1997.

I. Steinwart. Consistency of support vector machines and other regularized kernel machines. *IEEE Transactions on Information Theory*, 51:128–142, 2005.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1996.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32:56–84, 2004.

Journal of Machine Learning Research 9 (2008) 1369-1398 Submitted 1/08; Revised 4/08; Published 7/08

Coordinate Descent Method for Large-scale L2-loss Linear Support Vector Machines

Kai-Wei Chang B92084@CSIE.NTU.EDU.TW
Cho-Jui Hsieh B92085@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science
National Taiwan University
Taipei 106, Taiwan

Editor: Leon Bottou

Abstract

Linear support vector machines (SVM) are useful for classifying large-scale sparse data. Problems with sparse features are common in applications such as document classification and natural language processing. In this paper, we propose a novel coordinate descent algorithm for training linear SVM with the L2-loss function. At each step, the proposed method minimizes a one-variable sub-problem while fixing other variables. The sub-problem is solved by Newton steps with the line search technique. The procedure globally converges at the linear rate. As each sub-problem involves only values of a corresponding feature, the proposed approach is suitable when accessing a feature is more convenient than accessing an instance. Experiments show that our method is more efficient and stable than state of the art methods such as Pegasos and TRON.

Keywords: linear support vector machines, document classification, coordinate descent

1. Introduction

Support vector machines (SVM) (Boser et al., 1992) are a popular data classification tool. Given a set of instance-label pairs $(\mathbf{x}_j, y_j), j = 1, \ldots, l,\ \mathbf{x}_j \in R^n,\ y_j \in \{-1, +1\}$, SVM solves the following unconstrained optimization problem:

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{j=1}^{l} \xi(\mathbf{w}; \mathbf{x}_j, y_j), \tag{1}$$

where $\xi(\mathbf{w}; \mathbf{x}_j, y_j)$ is a loss function, and $C \in R$ is a penalty parameter. There are two common loss functions. L1-SVM uses the sum of losses and minimizes the following optimization problem:

$$f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{j=1}^{l} \max(1 - y_j\mathbf{w}^T\mathbf{x}_j, 0), \tag{2}$$

while L2-SVM uses the sum of squared losses, and minimizes

$$f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{j=1}^{l} \max(1 - y_j\mathbf{w}^T\mathbf{x}_j, 0)^2. \tag{3}$$

SVM is related to regularized logistic regression (LR), which solves the following problem:

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{j=1}^{l}\log(1+e^{-y_j\mathbf{w}^T\mathbf{x}_j}). \tag{4}$$

In some applications, we include a bias term b in SVM problems. For convenience, one may extend each instance with an additional dimension to eliminate this term:

$$\mathbf{x}_j^T \leftarrow [\mathbf{x}_j^T, 1] \qquad \mathbf{w}^T \leftarrow [\mathbf{w}^T, b].$$

SVM usually maps training vectors into a high-dimensional (and possibly infinite dimensional) space, and solves the dual problem of (1) with a nonlinear kernel. In some applications, data appear in a rich dimensional feature space, so that with/without nonlinear mapping obtain similar performances. If data are not mapped, we call such cases linear SVM, which are often encountered in applications such as document classification. While one can still solve the dual problem for linear SVM, directly solving (2) or (3) is possible. The objective function of L1-SVM (2) is nondifferentiable, so typical optimization methods cannot be directly applied. In contrast, L2-SVM (3) is a piecewise quadratic and strongly convex function, which is differentiable but not twice differentiable (Mangasarian, 2002). We focus on studying L2-SVM in this paper because of its differentiability.

In recent years, several optimization methods are applied to solve linear SVM in large-scale scenarios. For example, Keerthi and DeCoste (2005); Mangasarian (2002) propose modified Newton methods to train L2-SVM. As (3) is not twice differentiable, to obtain the Newton direction, they use the generalized Hessian matrix (i.e., generalized second derivative). A trust region Newton method (TRON) (Lin et al.) is proposed to solve logistic regression and L2-SVM. For large-scale L1-SVM, SVM$^{\text{perf}}$ (Joachims, 2006) uses a cutting plane technique to obtain the solution of (2). Smola et al. (2008) apply bundle methods, and view SVM$^{\text{perf}}$ as a special case. Zhang (2004) proposes a stochastic gradient method; Pegasos (Shalev-Shwartz et al., 2007) extends Zhang's work and develops an algorithm which alternates between stochastic gradient descent steps and projection steps. The performance is reported to be better than SVM$^{\text{perf}}$. Another stochastic gradient implementation similar to Pegasos is by Bottou (2007). All the above algorithms are iterative procedures, which update $\mathbf{w}$ at each iteration and generate a sequence $\{\mathbf{w}^k\}_{k=0}^{\infty}$. To distinguish these approaches, we consider the two extremes of optimization methods mentioned in the paper (Lin et al.):

$$\begin{array}{l}\text{Low cost per iteration;}\\ \text{slow convergence.}\end{array} \longleftrightarrow \begin{array}{l}\text{High cost per iteration;}\\ \text{fast convergence.}\end{array}$$

Among methods discussed above, Pegasos randomly subsamples a few instances at a time, so the cost per iteration is low, but the number of iterations is high. In contrast, Newton methods such as TRON take significant efforts at each iteration, but converge at fast rates. In large-scale scenarios, usually an approximate solution of the optimization problem is enough to produce a good model. Thus, methods with a low-cost iteration may be preferred as they can quickly generate a reasonable model. However, if one specifies an unsuitable stopping condition, such methods may fall into the situation of lengthy iterations. A recent overview on the tradeoff between learning accuracy and optimization cost is by Bottou and Bousquet (2008).

Coordinate descent is a common unconstrained optimization technique, but its use for large linear SVM has not been exploited much.[1] In this paper, we aim at applying it to L2-SVM. A coor-

1. For SVM with kernels, decomposition methods are popular, and they are related to coordinate descent methods. Since we focus on linear SVM, we do not discuss decomposition methods in this paper.

dinate descent method updates one component of $\mathbf{w}$ at a time by solving a one-variable sub-problem. It is competitive if one can exploit efficient ways to solve the sub-problem. For L2-SVM, the sub-problem is to minimize a single-variable piecewise quadratic function, which is differentiable but not twice differentiable. An earlier paper using coordinate descents for L2-SVM is by Zhang and Oles (2001). The algorithm, called CMLS, applies a modified Newton method to approximately solve the one-variable sub-problem. Here, we propose another modified Newton method, which obtains an approximate solution by line searches. Two key properties differentiate our method and CMLS:

1. Our proposed method attempts to use the full Newton step if possible, while CMLS takes a more conservative step. Our setting thus leads to faster convergence.

2. CMLS maintains the strict decrease of the function value, but does not prove the convergence. We prove that our method globally converges to the unique minimum.

We say $\hat{\mathbf{w}}$ is an ε-accurate solution if

$$f(\hat{\mathbf{w}}) \leq \min_{\mathbf{w}} f(\mathbf{w}) + \varepsilon.$$

We prove that our process obtains an ε-accurate solution in $O\left(nC^3P^6(\#\text{nz})^3\log(1/\varepsilon)\right)$ iterations, where the definitions of #nz and P can be found in the end of this section. Experiments show that our proposed method is more efficient and stable than existing algorithms.

Subsequent to this work, we and some collaborators propose a dual coordinate descent method for linear SVM (Hsieh et al., 2008). The method performs very well on document data (generally better than the primal-based method here). However, the dual method is not be stable for some non-document data with a small number of features. Clearly, if the number of features is much smaller than the number of instances, one should solve the primal form, which has less variables. In addition, the primal method uses the column format to store data (see Section 3.1). It is thus suitable for data stored as some form of inverted index in a very large database.

The organization of this paper is as follows. In Section 2, we describe and analyze our algorithm. Several implementation issues are discussed in Section 3. In Sections 4 and 5, we describe existing methods such as Pegasos, TRON and CMLS, and compare them with our approach. Results show that the proposed method is efficient and stable. Finally, we give discussions and conclusions in Section 6.

All sources used in this paper are available at
`http://www.csie.ntu.edu.tw/~cjlin/liblinear/exp.html`.

Notation The following notations are used in this paper. The input vectors are $\{\mathbf{x}_j\}_{j=1,\ldots,l}$, and x_{ji} is the ith feature of $\mathbf{x}_j$. For the problem size, l is the number of instances, n is number of features, and #nz is total number of nonzero values of training data.

$$m = \frac{\#\text{nz}}{n} \tag{5}$$

is the average number of nonzero values per feature, and

$$P = \max_{ji} |x_{ji}| \tag{6}$$

represents the upper bound of x_{ji}. We use $\|\cdot\|$ to represent the 2-norm of a vector.

Algorithm 1 Coordinate descent algorithm for L2-SVM

1. Start with any initial $\mathbf{w}^0$.
2. For $k = 0, 1, \ldots$ (outer iterations)
   (a) For $i = 1, 2, \ldots, n$ (inter iterations)
      i. Fix $w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_{i+1}^k, \ldots, w_n^k$ and approximately solve the sub-problem (7) to obtain w_i^{k+1}.

2. Solving Linear SVM via Coordinate Descent

In this section, we describe our coordinate descent method for solving L2-SVM given in (3). The algorithm starts from an initial point $\mathbf{w}^0$, and produces a sequence $\{\mathbf{w}^k\}_{k=0}^{\infty}$. At each iteration, $\mathbf{w}^{k+1}$ is constructed by sequentially updating each component of $\mathbf{w}^k$. This process generates vectors $\mathbf{w}^{k,i} \in R^n$, $i = 1, \ldots, n$, such that $\mathbf{w}^{k,1} = \mathbf{w}^k$, $\mathbf{w}^{k,n+1} = \mathbf{w}^{k+1}$, and

$$\mathbf{w}^{k,i} = [w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_i^k, \ldots, w_n^k]^T \text{ for } i = 2, \ldots, n.$$

For updating $\mathbf{w}^{k,i}$ to $\mathbf{w}^{k,i+1}$, we solve the following one-variable sub-problem:

$$\begin{aligned} &\min_z f(w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_i^k + z, w_{i+1}^k, \ldots, w_n^k) \\ \equiv &\min_z f(\mathbf{w}^{k,i} + z\mathbf{e}_i), \end{aligned} \tag{7}$$

where $\mathbf{e}_i = [\underbrace{0, \ldots, 0}_{i-1}, 1, 0, \ldots, 0]^T$. A description of the coordinate descent algorithm is in Algorithm 1. The function in (7) can be rewritten as

$$\begin{aligned} D_i(z) &= f(\mathbf{w}^{k,i} + z\mathbf{e}_i) \\ &= \frac{1}{2}(\mathbf{w}^{k,i} + z\mathbf{e}_i)^T(\mathbf{w}^{k,i} + z\mathbf{e}_i) + C \sum_{j \in I(\mathbf{w}^{k,i} + z\mathbf{e}_i)} (b_j(\mathbf{w}^{k,i} + z\mathbf{e}_i))^2, \end{aligned} \tag{8}$$

where

$$b_j(\mathbf{w}) = 1 - y_j \mathbf{w}^T \mathbf{x}_j \text{ and } I(\mathbf{w}) = \{j \mid b_j(\mathbf{w}) > 0\}.$$

In any interval of z where the set $I(\mathbf{w}^{k,i} + z\mathbf{e}_i)$ does not change, $D_i(z)$ is quadratic. Therefore, $D_i(z), z \in R$, is a piecewise quadratic function. As Newton method is suitable for quadratic optimization, here we apply it for minimizing $D_i(z)$. If $D_i(z)$ is twice differentiable, then the Newton direction at a given $\bar{z}$ would be

$$\frac{-D_i'(\bar{z})}{D_i''(\bar{z})}.$$

The first derivative of $D_i(z)$ is:

$$D_i'(z) = w_i^{k,i} + z - 2C \sum_{j \in I(\mathbf{w}^{k,i} + z\mathbf{e}_i)} y_j x_{ji} (b_j(\mathbf{w}^{k,i} + z\mathbf{e}_i)). \tag{9}$$

Unfortunately, $D_i(z)$ is not twice differentiable as the last term of $D_i'(z)$ is not differentiable at $\{z \mid b_j(\mathbf{w}^{k,i}+z\mathbf{e}_i)=0 \text{ for some } j\}$. We follow Mangasarian (2002) to define the generalized second derivative:

$$\begin{aligned} D_i''(z) &= 1+2C \sum_{j\in I(\mathbf{w}^{k,i}+z\mathbf{e}_i)} y_j^2 x_{ji}^2 \\ &= 1+2C \sum_{j\in I(\mathbf{w}^{k,i}+z\mathbf{e}_i)} x_{ji}^2. \end{aligned} \tag{10}$$

A simple Newton method to solve (7) begins with $z^0=0$ and iteratively updates z by the following way until $D_i'(z)=0$:

$$z^{t+1} = z^t - D_i'(z^t)/D_i''(z^t) \text{ for } t=0,1,\ldots. \tag{11}$$

Mangasarian (2002) proved that under an assumption, this procedure terminates in finite steps and solves (7). Coordinate descent methods are known to converge if at each inner iteration we uniquely attain the minimum of the sub-problem (Bertsekas, 1999, Proposition 2.7.1). Unfortunately, the assumption by Mangasarian (2002) may not hold in real cases, so taking the full Newton step (11) may not decrease the function $D_i(z)$. Furthermore, solving the sub-problem exactly is too expensive.

An earlier approach of using coordinate descents for L2-SVM without exactly solving the sub-problem is by Zhang and Oles (2001). In their algorithm CMLS, the approximate solution is restricted within a region. By evaluating the upper bound of generalized second-order derivatives in this region, one replaces the denominator of the Newton step (11) with that upper bound. This setting guarantees the decrease of $D_i(z)$. However, there are two problems. First, function decreasing does not imply that $\{\mathbf{w}^k\}$ converges to the global optimum. Secondly, the step size generated by evaluating the upper bound of generalized second derivatives may be too conservative. We describe details of CMLS in Section 4.3.

While coordinate descent methods have been well studied in optimization, most convergence analyses assume that the one-variable sub-problem is exactly solved. We consider the result by Grippo and Sciandrone (1999), which establishes the convergence by requiring only the following sufficient decrease condition:

$$D_i(z) - D_i(0) \le -\sigma z^2, \tag{12}$$

where z is the step taken and σ is any constant in (0, 1/2). Since we intend to take the Newton direction

$$d = \frac{-D_i'(0)}{D_i''(0)}, \tag{13}$$

it is important to check if $z=d$ satisfies (12). The discussion below shows that in general the condition hold. If the function $D_i(z)$ is quadratic around 0, then

$$D_i(z) - D_i(0) = D_i'(0)z + \frac{1}{2}D_i''(0)z^2.$$

Using $D_i''(0) > 1$ in (10), $z=d=-D_i'(0)/D_i''(0)$ leads to

$$-\frac{D_i'(0)^2}{2D_i''(0)} \le -\sigma \frac{D_i'(0)^2}{D_i''(0)^2},$$

so (12) holds. As $D_i(z)$ is only piecewise quadratic, (12) may not hold using $z=d$. However, we can conduct a simple line search. The following theorem shows that there is a $\lambda \in (0,1)$ such that $z=\lambda d$ satisfies the sufficient decrease condition:

Algorithm 2 Solving the sub-problem using Newton direction with the line search.

1. Given $\mathbf{w}^{k,i}$. Choose $\beta \in (0,1)$ (e.g., $\beta = 0.5$).
2. Calculate the Newton direction $d = -D_i'(0)/D_i''(0)$.
3. Compute $\lambda = \max\{1, \beta, \beta^2, \ldots\}$ such that $z = \lambda d$ satisfies (12).

Theorem 1 *Given the Newton direction d as in* (13)*. Then $z = \lambda d$ satisfies* (12) *for all $0 \le \lambda \le \bar{\lambda}$, where*

$$\bar{\lambda} = \frac{D_i''(0)}{H_i/2 + \sigma} \text{ and } H_i = 1 + 2C \sum_{j=1}^{l} x_{ji}^2. \tag{14}$$

The proof is in Appendix A.1. Therefore, at each inner iteration of Algorithm 1, we take the Newton direction d as in (13), and then sequentially check $\lambda = 1, \beta, \beta^2, \ldots$, where $\beta \in (0,1)$, until λd satisfies (12). Algorithm 2 lists the details of a line search procedure. We did not specify how to approximately solve sub-problems in Algorithm 1. From now on, we assume that it uses Algorithm 2.

Calculating $D_i(\lambda d)$ is the main cost of checking (12). We can use a trick to reduce the number of $D_i(\lambda d)$ calculations. Theorem 1 indicates that if

$$0 \le \lambda \le \bar{\lambda} = \frac{D_i''(0)}{H_i/2 + \sigma}, \tag{15}$$

then $z = \lambda d$ satisfies the sufficient decrease condition (12). H_i is independent of $\mathbf{w}$, so it can be precomputed before training. Furthermore, we already evaluate $D_i''(0)$ in computing the Newton step, so it takes only constant time to check (15). At Step 3 of Algorithm 2, we sequentially use $\lambda = 1, \beta, \beta^2, \ldots$, etc. Before calculating (12) using a smaller λ, we check if λ satisfies (15). If it does, then there is no need to evaluate the new $D_i(\lambda d)$. If $\lambda = 1$ already satisfies (15), the line search procedure is essentially waived. Thus the computational time is effectively reduced.

We discuss parameters in our algorithm. First, as $\lambda = 1$ is often successful, our algorithm is insensitive to β. We choose β as 0.5. Secondly, there is a parameter σ in (12). The smaller value of σ leads to a looser sufficient decrease condition, which reduces the time of line search, but increases the number of outer iterations. A common choice of σ is 0.01 in unconstrained optimization algorithms.

It is important to study the convergence properties of Algorithm 1. An excellent study on the convergence rate of coordinate descent methods is by Luo and Tseng (1992). They assume that each sub-problem is exactly solved, so we cannot apply their results here. The following theorem proves the convergence results of Algorithm 1.

Theorem 2 *The sequence $\{\mathbf{w}^k\}$ generated by Algorithm 1 linearly converges. That is, there is a constant $\mu \in (0,1)$ such that*

$$f(\mathbf{w}^{k+1}) - f(\mathbf{w}^*) \le (1-\mu)(f(\mathbf{w}^k) - f(\mathbf{w}^*)), \forall k.$$

Moreover, the sequence $\{\mathbf{w}^k\}$ globally converges to $\mathbf{w}^*$. The algorithm obtains an ε-accurate solution in

$$O\left(nC^3P^6(\#nz)^3 \log(1/\varepsilon)\right) \tag{16}$$

iterations.

The proof is in Appendix A.2. Note that as data are usually scaled before training, $P \leq 1$ in most practical cases.

Next, we investigate the computational complexity per outer iteration of Algorithm 1. The main cost comes from solving the sub-problem by Algorithm 2. At Step 2 of Algorithm 2, to evaluate $D_i'(0)$ and $D_i''(0)$, we need $b_j(\mathbf{w}^{k,i})$ for all j. Here we consider sparse data instances. Calculating $b_j(\mathbf{w}), j = 1,\ldots,l$ takes $O(\#\text{nz})$ operations, which are large. However, one can use the following trick to save the time:

$$b_j(\mathbf{w}+z\mathbf{e}_i) = b_j(\mathbf{w}) - zy_jx_{ji}, \tag{17}$$

If $b_j(\mathbf{w}), j = 1,\ldots,l$ are available, then obtaining $b_j(\mathbf{w}+z\mathbf{e}_i)$ involves only nonzero x_{ji}'s of the ith feature. Using (17), obtaining all $b_j(\mathbf{w}+z\mathbf{e}_i)$ costs $O(m)$, where m, the average number of nonzero values per feature, is defined in (5). To have $b_j(\mathbf{w}^0)$, we can start with $\mathbf{w}^0 = \mathbf{0}$, so $b_j(\mathbf{w}^0) = 1, \forall j$. With $b_j(\mathbf{w}^{k,i})$ available, the cost of evaluating $D_i'(0)$ and $D_i''(0)$ is $O(m)$. At Step 3 of Algorithm 2, we need several line search steps using $\lambda = 1, \beta, \beta^2, \ldots$, etc. For each λ, the main cost is on calculating

$$\begin{aligned} D_i(\lambda d) - D_i(0) = &\frac{1}{2}(w_i^{k,i}+\lambda d)^2 - \frac{1}{2}(w_i^{k,i})^2 \\ &+ C\Big(\sum_{j\in I(\mathbf{w}^{k,i}+\lambda d\mathbf{e}_i)} (b_j(\mathbf{w}^{k,i}+\lambda d\mathbf{e}_i))^2 - \sum_{j\in I(\mathbf{w}^{k,i})} (b_j(\mathbf{w}^{k,i}))^2\Big). \end{aligned} \tag{18}$$

Note that from (17), if $x_{ji} = 0$,

$$b_j(\mathbf{w}^{k,i}+\lambda d\mathbf{e}_i) = b_j(\mathbf{w}^{k,i}).$$

Hence, (18) involves no more than $O(m)$ operations. In summary, Algorithm 2 costs

$$\begin{aligned} & O(m) \text{ for evaluating } D_i'(0) \text{ and } D_i''(0) \\ + \quad & O(m) \times \#\text{ line search steps.} \end{aligned}$$

From the explanation earlier and our experiments, in general the sufficient decrease condition holds when $\lambda = 1$. Then the cost of Algorithm 2 is about $O(m)$. Therefore, in general the complexity per outer iteration is:

$$O(nm) = O(\#\text{nz}). \tag{19}$$

3. Implementation Issues

In this section, we discuss some techniques for a fast implementation of Algorithm 1. First, we aim at suitable data representations. Secondly, we show that the order of sub-problems at each iteration can be any permutation of $\{1,\ldots,n\}$. Experiments in Section 5 indicate that the performance of using a random permutation is superior to that of using the fixed order $1,\ldots,n$. Finally, we present an online version of our algorithm.

3.1 Data Representation

For sparse data, we use a sparse matrix

$$X = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_l^T \end{bmatrix} \tag{20}$$

to store the training instances. There are several ways to implement a sparse matrix. Two common ones are "row format" and "column format" (Duff et al., 1989). For data classification, using column (row) format allows us to easily access any particular feature (instance). In our case, as we decompose the problem (3) into sub-problems over features, the column format is more suitable.

3.2 Random Permutation of Sub-problems

In Section 2, we propose a coordinate descent algorithm which solves the one-variable sub-problems in the order of $w_1, \ldots, w_n$. As the features may be correlated, the order of features may affect the training speed. One can even use an arbitrary order of sub-problems. To prove the convergence, we require that each sub-problem is solved once at one outer iteration. Therefore, at the kth iteration, we construct a random permutation π_k of $\{1, \ldots, n\}$, and sequentially minimize with respect to variables $w_{\pi(1)}, w_{\pi(2)}, \ldots, w_{\pi(n)}$. Similar to Algorithm 1, the algorithm generates a sequence $\{\mathbf{w}^{k,i}\}$ such that $\mathbf{w}^{k,1} = \mathbf{w}^k$, $\mathbf{w}^{k,n+1} = \mathbf{w}^{k+1,1}$ and

$$w_t^{k,i} = \begin{cases} w_t^{k+1} & \text{if } \pi_k^{-1}(t) < i, \\ w_t^k & \text{if } \pi_k^{-1}(t) \geq i. \end{cases}$$

The update from $\mathbf{w}^{k,i}$ to $\mathbf{w}^{k,i+1}$ is by

$$w_t^{k,i+1} = w_t^{k,i} + \arg\min_z f(\mathbf{w}^{k,i} + z\mathbf{e}_{\pi_k(i)}) \quad \text{if } \pi_k^{-1}(t) = i.$$

We can prove the same convergence result:

Theorem 3 *Results in Theorem 2 hold for Algorithm 1 with random permutations π_k.*

The proof is in Appendix A.3. Experiments in Section 5 show that a random permutation of sub-problems leads to faster training.

3.3 An Online Algorithm

If the number of features is very large, we may not need to go through all $\{w_1, \ldots, w_n\}$ at each iteration. Instead, one can have an online setting by arbitrarily choosing a feature at a time. That is, from $\mathbf{w}^k$ to $\mathbf{w}^{k+1}$ we only modify one component. A description is in Algorithm 3. The following theorem indicates the convergence rate in expectation:

Theorem 4 *Let $\delta \in (0,1)$. Algorithm 3 requires $O\left(nl^2C^3P^6(\#nz)\log(\frac{1}{\delta\varepsilon})\right)$ iterations to obtain an ε-accurate solution with confidence $1-\delta$.*

The proof is in Appendix A.4.

4. Related Methods

In this section, we discuss three existing schemes for large-scale linear SVM. They will be compared in Section 5. The first one is Pegasos (Shalev-Shwartz et al., 2007), which is notable for its efficiency in training linear L1-SVM. The second one is a trust region Newton method (Lin et al.). It is one of the fastest implementations for L2-SVM. The last one is CMLS, which is a coordinate descent method proposed by Zhang and Oles (2001).

Algorithm 3 An online coordinate descent algorithm

1. Start with any initial $\mathbf{w}^0$.
2. For $k = 0, 1, \ldots$
   (a) Randomly choose $i_k \in \{1, 2, \ldots, n\}$.
   (b) Fix $w_1^k, \ldots, w_{i_k-1}^k, w_{i_k+1}^k, \ldots, w_n^k$ and approximately solve the sub-problem (7) to obtain $w_{i_k}^{k+1}$.

Coordinate descent methods have been used in other machine learning problems. For example, Rätsch et al. (2002) discuss the connection between boosting/logistic regression and coordinate descent methods. Their strategies for selecting coordinates at each outer iteration are different from ours. We do not discuss details here.

4.1 Pegasos for L1-SVM

We briefly introduce the Pegasos algorithm (Shalev-Shwartz et al., 2007). It is an efficient method to solve the following L1-SVM problem:

$$\min_{\mathbf{w}} g(\mathbf{w}) = \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{l}\sum_{j=1}^{l} \max(1 - y_j \mathbf{w}^T \mathbf{x}_j, 0). \tag{21}$$

By setting $\lambda = \frac{1}{Cl}$, we have

$$g(\mathbf{w}) = f(\mathbf{w})/Cl, \tag{22}$$

where $f(\mathbf{w})$ is the objective function of (2). Thus (21) and (2) are equivalent. Pegasos has two parameters. One is the subsample size K, and the other is the penalty parameter λ. It begins with an initial $\mathbf{w}^0$ whose norm is at most $1/\sqrt{\lambda}$. At each iteration k, it randomly selects a set $A_k \subset \{\mathbf{x}_j, y_j\}_{j=1,\ldots,l}$ of size K as the subsamples of training instances and sets a learning rate

$$\eta_k = \frac{1}{\lambda k}. \tag{23}$$

Then it updates $\mathbf{w}^k$ with the following rules:

$$\begin{aligned}
\mathbf{w}^{k+1} &= \min\left(1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}^{k+\frac{1}{2}}\|}\right)\mathbf{w}^{k+\frac{1}{2}}, \\
\mathbf{w}^{k+\frac{1}{2}} &= \mathbf{w}^k - \eta_k \nabla_k, \\
\nabla_k &= \lambda \mathbf{w}^k - \frac{1}{K}\sum_{j \in A_k^+(\mathbf{w}^k)} y_j \mathbf{x}_j, \\
A_k^+(\mathbf{w}) &= \{j \in A_k \mid 1 - y_j \mathbf{w}^T \mathbf{x}_j > 0\},
\end{aligned} \tag{24}$$

where ∇_k is considered as a sub-gradient of the approximate objective function:

$$\frac{\lambda}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{K}\sum_{j \in A_k} \max(1 - y_j \mathbf{w}^T \mathbf{x}_j, 0).$$

Algorithm 4 Pegasos algorithm for solving L1-SVM.

1. Given λ, K, and $\mathbf{w}^0$ with $\|\mathbf{w}^0\| \leq 1/\sqrt{\lambda}$.
2. For $k = 0, 1, \ldots$
   (a) Select a set $A_k \in \{\mathbf{x}_j, y_j \mid j = 1 \ldots l\}$, and the learning rate η by (23).
   (b) Obtain $\mathbf{w}^{k+1}$ by (24).

Here $\mathbf{w}^{k+1/2}$ is a vector obtained by the stochastic gradient descent step, and $\mathbf{w}^{k+1}$ is the projection of $\mathbf{w}^{k+1/2}$ to the set $\{\mathbf{w} \mid \|\mathbf{w}\| \leq 1/\sqrt{\lambda}\}$. Algorithm 4 lists the detail of Pegasos. The parameter K decides the number of training instances involved at an iteration. If $K = l$, Pegasos considers all examples at each iteration, and becomes a subgradient projection method. In this case the cost per iteration is $O(\#\text{nz})$. If $K < l$, Pegasos is a randomized algorithm. For the extreme case of $K = 1$, Pegasos chooses only one training instance for updating. Thus the average cost per iteration is $O(\#\text{nz}/l)$. In subsequent experiments, we set the subsample size K to one as Shalev-Shwartz et al. (2007) suggested.

Regarding the complexity of Pegasos, we first compare Algorithm 1 with Pegasos ($K = l$). Both algorithms are deterministic and cost $O(\#\text{nz})$ per iteration. Shalev-Shwartz et al. (2007) prove that Pegasos with $K = l$ needs $\tilde{O}(R^2/(\varepsilon_g \lambda))$ iterations to achieve an ε_g-accurate solution, where $R = \max_j \|\mathbf{x}_j\|$, and $\tilde{O}(h(n))$ is shorthand for $O(h(n)\log^k h(n))$, for some $k \geq 0$. We use ε_g as Pegasos considers $g(\mathbf{w})$ in (22), a scaled form of $f(\mathbf{w})$. From (1), an ε_g-accurate solution for $g(\mathbf{w})$ is equivalent to an (ε/Cl)-accurate solution for $f(\mathbf{w})$. With $\lambda = 1/Cl$ and $R^2 = O(P^2(\#\text{nz})/l)$, where P is defined in (6), Pegasos takes

$$\tilde{O}\left(\frac{C^2P^2l(\#\text{nz})}{\varepsilon}\right)$$

iterations to achieve an ε-accurate solution. One can compare this value with (16), the number of iterations by Algorithm 1.

Next, we compare two random algorithms: Pegasos with $K = 1$ and our Algorithm 3. Shalev-Shwartz et al. (2007) prove that Pegasos takes $\tilde{O}(\frac{R^2}{\lambda\delta\varepsilon_g})$ iterations to obtain an ε_g-accurate solution with confidence $1 - \delta$. Using a similar derivation in the last paragraph, we can show that this is equivalent to $\tilde{O}(C^2P^2l(\#\text{nz})/\delta\varepsilon)$. As the cost per iteration is $O(\#\text{nz}/l)$, the overall complexity is

$$\tilde{O}\left(\frac{C^2P^2(\#\text{nz})^2}{\delta\varepsilon}\right).$$

For our Algorithm 3, each iteration costs $O(m)$, so following Theorem 4 the overall complexity is $O\left(l^2C^3P^6(\#\text{nz})^2\log(\frac{1}{\delta\varepsilon})\right)$.

Based on the above analysis, the number of iterations required for our algorithm is proportional to $O(\log(1/\varepsilon))$, while that for Pegasos is $O(1/\varepsilon)$. Therefore, our algorithm tends to have better final convergence than Pegasos for both deterministic and random settings. However, for the dependence on the size of data (number of instances and features), our algorithm is worse.

Regarding the stopping condition, as at each iteration Pegasos only takes one sample for updating $\mathbf{w}$, neither function nor gradient information is available. This keeps Pegasos from designing a suitable stopping condition. Shalev-Shwartz et al. (2007) suggest to set a maximal number of

iterations. However, deciding a suitable value may be difficult. We will discuss stopping conditions of Pegasos and other methods in Section 5.3.

4.2 Trust Region Newton Method (TRON) for L2-SVM

Recently, Lin et al. introduced a trust region Newton method for logistic regression. Their proposed method can be extended to L2-SVM. In this section, we briefly discuss their approach. For convenience, in subsequent sections, we use TRON to indicate the trust region Newton method for L2-SVM, and TRON-LR for logistic regression

The optimization procedure of TRON has two layers of iterations. At each outer iteration k, TRON sets a size Δ_k of the trust region, and builds a quadratic model

$$q_k(\mathbf{s}) = \nabla f(\mathbf{w}^k)^T\mathbf{s} + \frac{1}{2}\mathbf{s}^T\nabla^2 f(\mathbf{w}^k)\mathbf{s}$$

as the approximation of the value $f(\mathbf{w}^k+\mathbf{s}) - f(\mathbf{w}^k)$, where $f(\mathbf{w})$ is the objective function in (3) and $\nabla^2 f(\mathbf{w})$ is the generalized Hessian (Mangasarian, 2002) of $f(\mathbf{w})$. Then an inner conjugate gradient procedure approximately finds the Newton direction by minimizing the following optimization problem:

$$\begin{aligned} \min_{\mathbf{s}} \quad & q_k(\mathbf{s}) \\ \text{subject to} \quad & \|\mathbf{s}\| \le \Delta_k. \end{aligned} \tag{25}$$

TRON updates $\mathbf{w}^k$ and Δ_k by the following rules:

$$\begin{aligned} \mathbf{w}^{k+1} &= \begin{cases} \mathbf{w}^k + \mathbf{s}^k & \text{if } \rho_k > \eta_0, \\ \mathbf{w}^k & \text{if } \rho_k \le \eta_0, \end{cases} \\ \Delta_{k+1} &\in \begin{cases} [\sigma_1 \min\{\|\mathbf{s}^k\|, \Delta_k\}, \sigma_2\Delta_k] & \text{if } \rho_k \le \eta_1, \\ [\sigma_1\Delta_k, \sigma_3\Delta_k] & \text{if } \rho_k \in (\eta_1, \eta_2), \\ [\Delta_k, \sigma_3\Delta_k] & \text{if } \rho_k \ge \eta_2, \end{cases} \\ \rho_k &= \frac{f(\mathbf{w}^k + \mathbf{s}^k) - f(\mathbf{w}^k)}{q_k(\mathbf{s}^k)}, \end{aligned} \tag{26}$$

where ρ_k is the ratio of the actual reduction in the objective function to the approximation model $q_k(\mathbf{s})$. Users pre-specify parameters $\eta_0 > 0$, $1 > \eta_2 > \eta_1 > 0$, and $\sigma_3 > 1 > \sigma_2 > \sigma_1 > 0$. We use

$$\begin{aligned} \eta_0 &= 10^{-4}, \eta_1 = 0.25, \eta_2 = 0.75, \\ \sigma_1 &= 0.25, \sigma_2 = 0.5, \sigma_3 = 4, \end{aligned}$$

as suggested by Lin et al.. The procedure is listed in Algorithm 5.

For the computational complexity, the main cost per TRON iteration is

$$O(\#\text{nz}) \times (\#\text{ conjugate gradient iterations}). \tag{27}$$

Compared to our approach or Pegasos, the cost per TRON iteration is high. It keeps TRON from quickly obtaining a usable model. However, when $\mathbf{w}$ gets close to the minimum, TRON takes the Newton step to achieve fast convergence. We give more observations in the experiment section.

Algorithm 5 Trust region Newton method for L2-SVM.

1. Given $\mathbf{w}^0$.
2. For $k = 0, 1, \ldots$
   (a) Find an approximate solution $\mathbf{s}^k$ of the trust region sub-problem (25).
   (b) Update $\mathbf{w}^k$ and Δ_k according to (26).

4.3 CMLS: A Coordinate Descent Method for L2-SVM

In Sections 2 and 3, we introduced our coordinate descent method for solving L2-SVM. Here, we discuss the previous work (Zhang and Oles, 2001), which also applies the coordinate descent technique. Zhang and Oles refer to their method as CMLS. At each outer iteration k, it sequentially minimizes sub-problems (8) by updating one variable of (3) at a time. In solving the sub-problem, Zhang and Oles (2001) mention that using line searches may result in small step sizes. Hence, CMLS applies a technique similar to the trust region method. It sets a size $\Delta^{k,i}$ of the trust region, evaluates the first derivative (9) of (8), and calculates the upper bound of the generalized second derivative subject to $|z| \le \Delta^{k,i}$:

$$U_i(z) = 1 + \sum_{j=1}^{l} \beta_j(\mathbf{w}^{k,i} + z\mathbf{e}_i),$$

$$\beta_j(\mathbf{w}) = \begin{cases} 2C & \text{if } y_j \mathbf{w}^T \mathbf{x}_j \le 1 + |\Delta^{k,i} x_{ij}|, \\ 0 & \text{otherwise.} \end{cases}$$

Then we obtain the step z as:

$$z = \min(\max(-\frac{D_i'(z)}{U_i(z)}, -\Delta^{k,i}), \Delta^{k,i}). \tag{28}$$

The updating rule of Δ is:

$$\Delta^{k+1,i} = 2|z| + \varepsilon, \tag{29}$$

where ε is a small constant.

In order to speed up the process, Zhang and Oles (2001) smooth the objective function of sub-problems with a parameter $c_k \in [0, 1]$:

$$D_i(z) = \frac{1}{2}(\mathbf{w}^{k,i} + z\mathbf{e}_i)^T(\mathbf{w}^{k,i} + z\mathbf{e}_i) + C\sum_{j=1}^{l}(b_j(\mathbf{w}^{k,i} + z\mathbf{e}_i))^2,$$

where

$$b_j(\mathbf{w}) = \begin{cases} 1 - y_j \mathbf{w}^T \mathbf{x}_j & \text{if } 1 - y_j \mathbf{w}^T \mathbf{x}_j > 0, \\ c_k(1 - y_j \mathbf{w}^T \mathbf{x}_j) & \text{otherwise.} \end{cases}$$

Following the setting by (Zhang and Oles, 2001), we choose

$$c_k = \max(0, 1 - k/50), \tag{30}$$

Algorithm 6 CMLS algorithm for solving L2-SVM.

1. Given $\mathbf{w}^0$ and set initial $\Delta^{0,i} = 10, \forall i$.
2. For $k = 0, 1, \ldots$
   (a) Set c_k by (30). Let $\mathbf{w}^{k,1} = \mathbf{w}^k$.
   (b) For $i = 1, 2, \ldots, n$
      i. Evaluate z by (28).
      ii. $\mathbf{w}^{k,i+1} = \mathbf{w}^{k,i} + z\mathbf{e}_i$.
      iii. Update Δ by (29).
   (c) Let $\mathbf{w}^{k+1} = \mathbf{w}^{k,n+1}$.

Problem	l	n	#nz
astro-physic	62,369	99,757	4,834,550
real-sim	72,309	20,958	3,709,083
news20	19,996	1,355,191	9,097,916
yahoo-japan	176,203	832,026	23,506,415
rcv1	677,399	47,236	49,556,258
yahoo-korea	460,554	3,052,939	156,436,656

Table 1: Data set statistics: l is the number of instances and n is the number of features.

and set the initial $\mathbf{w} = \mathbf{0}$ and $\Delta^{0,i} = 10, \forall i$. We find that the result is insensitive to these parameters. The detail of CMLS algorithm is listed in Algorithm 6.

Zhang and Oles (2001) prove that if $c_k = 0, \forall k$, then the objective function of (3) is decreasing after each inner iteration. However, such a property may not imply that Algorithm 6 converges to the minimum. In addition, CMLS updates $\mathbf{w}$ by (28), which is more conservative than Newton steps. In Section 5, we show that CMLS takes more time and iterations than ours to obtain a solution.

5. Experiments and Analysis

In this section, we conduct two experiments to investigate the performance of our proposed coordinate descent algorithm. The first experiment compares our method with other L2-SVM solvers in terms of the speed to reduce function/gradient values. The second experiment evaluates various state of the art linear classifiers for L1-SVM, L2-SVM, and logistic regression. We also discuss the stopping condition of these methods.

5.1 Data Sets

Table 1 lists the number of instances (l), features (n), and non-zero elements (#nz) of six data sets. All sets are from document classification. Past studies show that linear SVM performs as good as kernelized ones for such data. Details of astro-physic are mentioned in Joachims (2006), while others are in Lin et al.. Three data sets real-sim, news20 and rcv1 are publicly available

Data set	CDPER	CD	TRON	CMLS
astro-physic	**0.5**	1.2	1.2	2.6
real-sim	**0.2**	0.3	0.9	2.0
news20	2.4	**1.0**	5.2	5.3
yahoo-japan	**2.9**	9.3	38.2	13.5
rcv1	**5.1**	10.8	18.6	54.8
yahoo-korea	**18.4**	58.1	286.1	146.3

Table 2: The training time for an L2-SVM solver to reduce the objective value to within 1% of the optimal value. Time is in seconds. We use $C = 1$. The approach with the shortest running time is boldfaced.

at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`. A brief reminder for each data set can be found below.

- astro-physic: This set is a classification problem of scientific papers from Physics ArXiv.
- real-sim: This set includes some Usenet articles.
- news20: This is a collection of news documents, and was preprocessed by Keerthi and DeCoste (2005).
- yahoo-japan: We use binary term frequencies and normalize each instance to unit length.
- rcv1: This set (Lewis et al., 2004) is an archive of manually categorized newswire stories from Reuters Ltd. Each vector is a cosine normalization of a log transformed TF-IDF (term frequency, inverse document frequency) feature vector.
- yahoo-korea: Similar to yahoo-japan, we use binary term frequencies and normalize each instance to unit length.

To examine the testing accuracy, we use a stratified selection to split each set to $4/5$ training and $1/5$ testing.

5.2 Comparisons

We compare the following six implementations. TRON-LR is for logistic regression, Pegasos is for L1-SVM, and all others are for L2-SVM.

1. CD: the coordinate descent method described in Section 2. We choose σ in (12) as 0.01.
2. CDPER: the method modified from CD by permuting sub-problems at each outer step. See the discussion in Section 3.2.
3. CMLS: a coordinate descent method for L2-SVM (Zhang and Oles, 2001, Algorithm 3). It is discussed in Section 4.3.
4. TRON: the trust region Newton method (Lin et al.) for L2-SVM. See the discussion in Section 4.2. We use the L2-loss linear SVM implementation in the software LIBLINEAR (version 1.21 with option `-s 2`; `http://www.csie.ntu.edu.tw/~cjlin/liblinear`).

Data set	L2-SVM		L1-SVM		LR	
	C	Accuracy	C	Accuracy	C	Accuracy
astro-physic	0.5	97.14	1.0	97.09	8.0	97.03
real-sim	1.0	97.59	1.0	97.52	8.0	97.57
news20	4.0	96.85	2.0	96.70	64.0	96.17
yahoo-japan	0.5	92.91	1.0	92.97	4.0	92.76
rcv1	0.5	97.77	1.0	97.77	8.0	97.76
yahoo-korea	2.0	87.51	4.0	87.42	64.0	87.31

Table 3: The best parameter C and the corresponding testing accuracy of L1-SVM, L2-SVM and logistic regression (LR). We conduct five-fold cross validation to select C.

5. TRON-LR: the trust region Newton method for logistic regression introduced by Lin et al.. Similar to TRON, we use the implementation in the software LIBLINEAR with option `-s 0`.

6. Pegasos: the primal estimated sub-gradient solver for L1-SVM (Shalev-Shwartz et al., 2007). See the discussion in Section 4.1. The source code is available online at `http://ttic.uchicago.edu/~shai/code`.

We do not include the bias term in all the solvers. All the above algorithms are implemented in C++ with double-precision floating-point numbers. Using single precision (e.g., Bottou, 2007) may reduce the computational time in some situations, but this setting may cause numerical inaccuracy. We conduct experiments on an Intel 2.66GHz processor with 8GB of main memory under Linux.

In our first experiment, we compare L2-SVM solvers (with $C = 1$) in term of the speed to reduce function/gradient values. In Table 2, we check their CPU time of reducing the relative difference of the function value to the optimum,

$$\frac{f(\mathbf{w}^k) - f(\mathbf{w}^*)}{|f(\mathbf{w}^*)|}, \tag{31}$$

to within 0.01. We run TRON with the stopping condition $\|\nabla f(\mathbf{w}^k)\| \leq 0.01$ to obtain the reference solutions. Since objective values are stable under such strict stopping conditions, these solutions are seen to be very close to the optima. Overall, our proposed algorithms CDPER and CD perform well on all the data sets. For the large data sets (rcv1, yahoo-japan, yahoo-korea), CD is significantly better than TRON and CMLS. With the permutation of sub-problems, CDPER is even better than CD. To show more detailed comparisons, Figure 1 presents time versus relative difference (31). As a reference, we draw a horizontal dotted line to indicate the relative difference 0.01. Consistent with the observation in Table 2, CDPER is more efficient and stable than others.

In addition, we are interested in how fast these methods decrease the norm of gradients. Figure 2 shows the result. Overall, CDPER converges faster in the beginning, while TRON is the best for final convergence.

The second experiment is to check the relationship between training time and testing accuracy using our implementation and other solvers: CMLS, TRON (L2-SVM and logistic regression), and Pegasos. That is, we investigate which method achieves reasonable testing accuracy more quickly. To have a fair evaluation, we conduct five-fold cross validation to select the best parameter C for each learning method. Using the selected C, we then train the whole training set and predict the

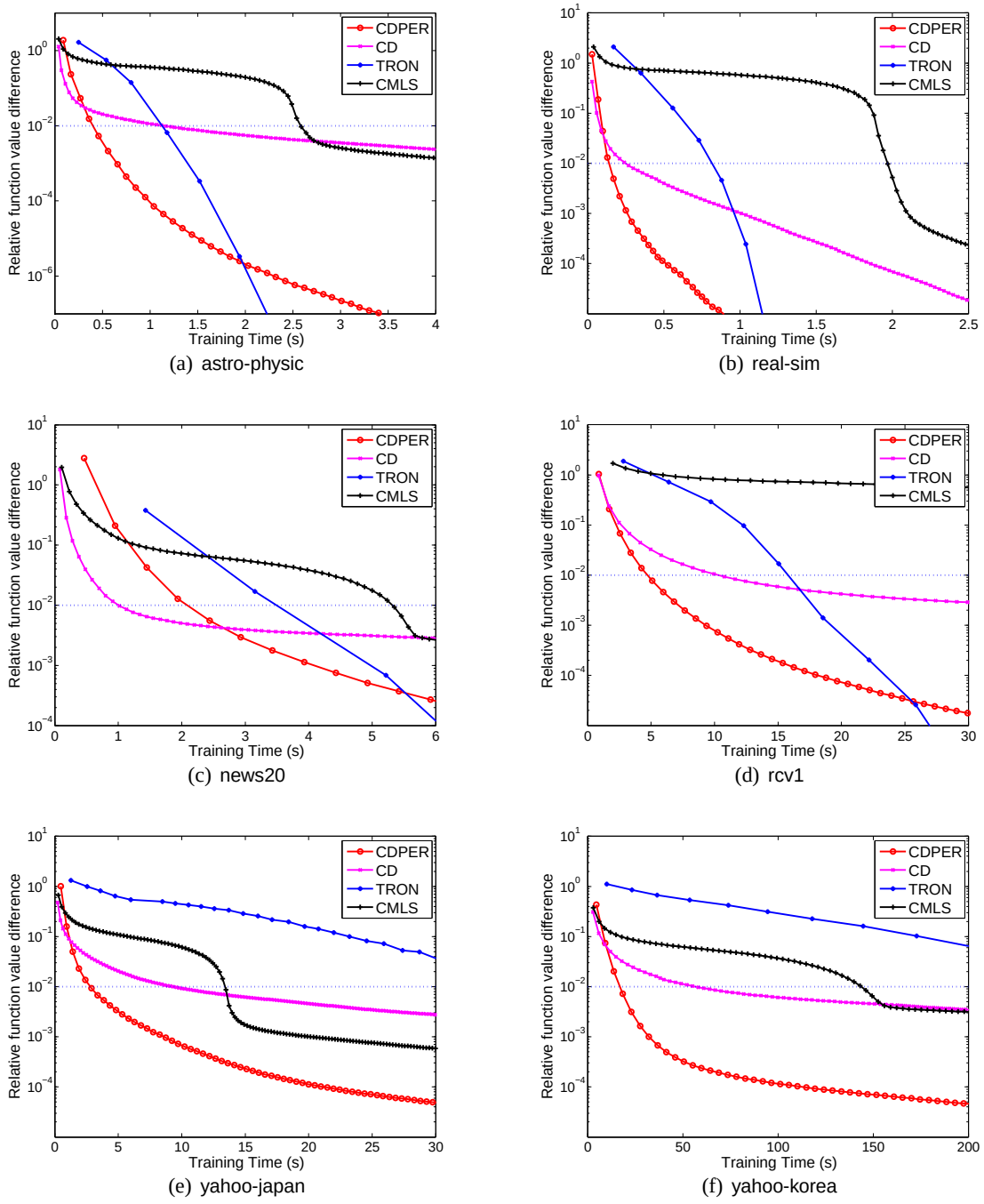


Figure 1: Time versus the relative difference of the objective value to the minimum. The dotted line indicates the relative difference 0.01. We show the training time for each solver to reach this ratio in Table 2. Time is in seconds. $C = 1$ is used.

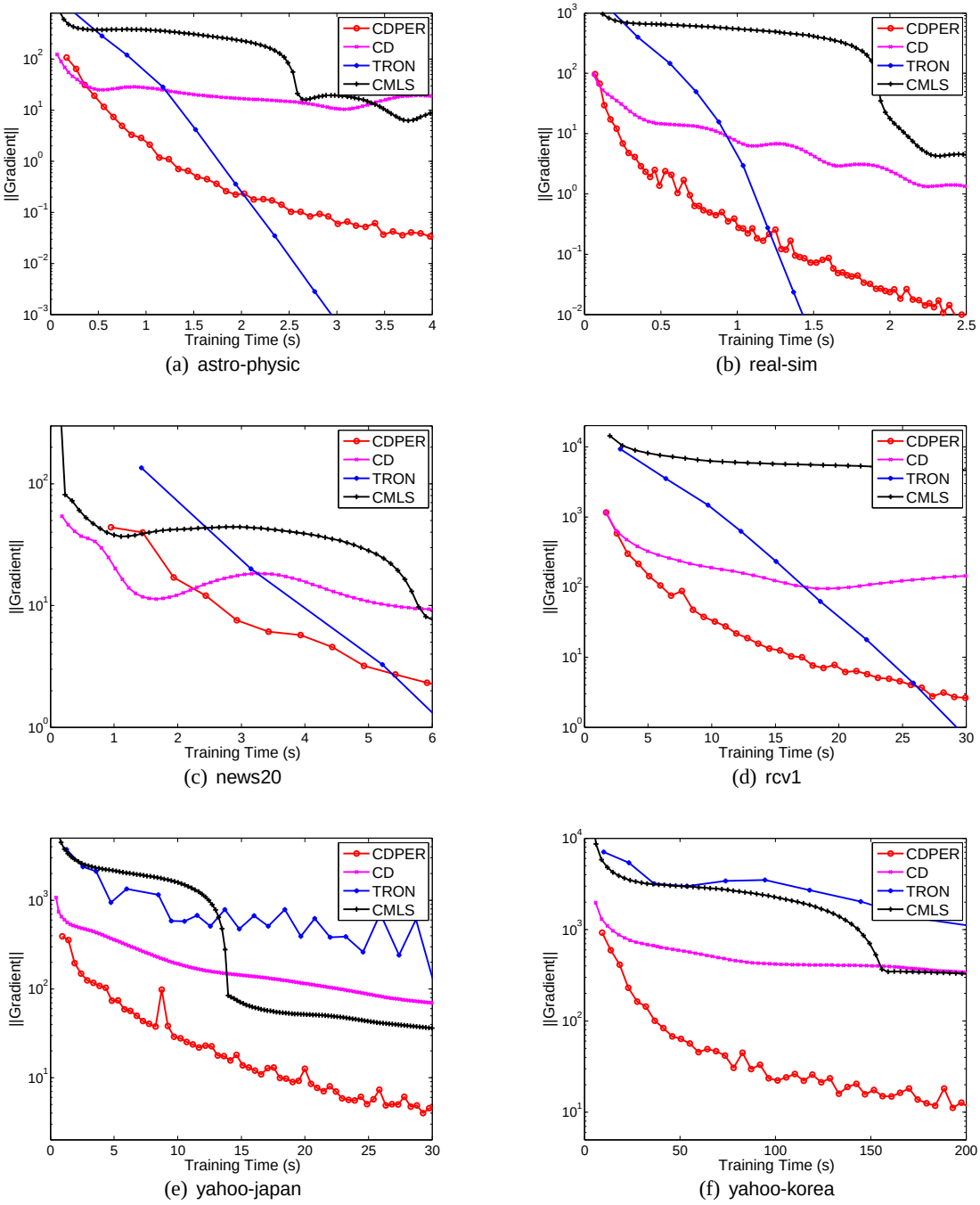


Figure 2: The two-norm of gradient versus the training time. Time is in seconds. $C = 1$ is used.

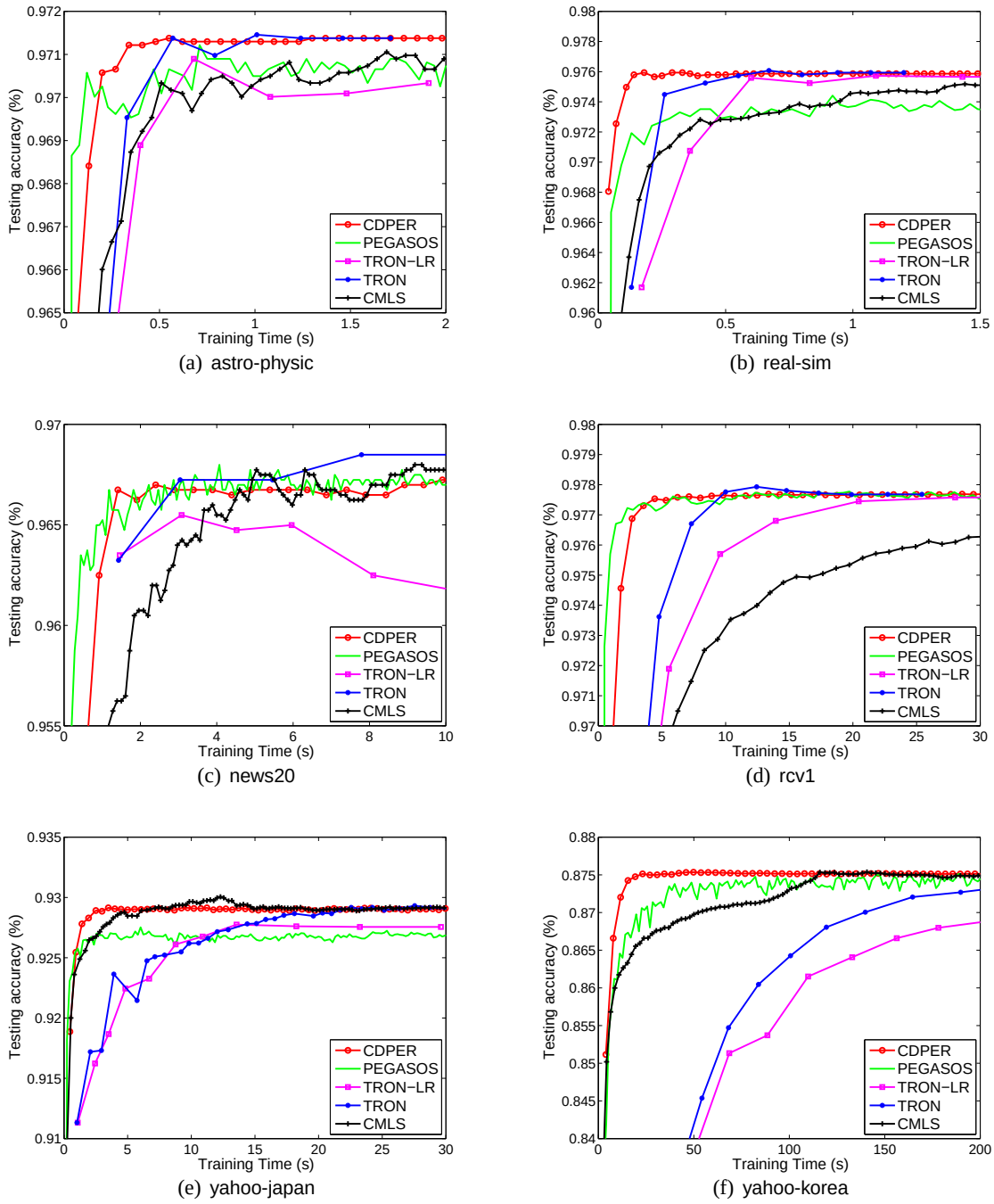


Figure 3: Testing accuracy versus the training time. Time is in seconds. We train each data set using the best C from cross validation. (see Table 3 for details.)

testing set. Table 3 presents the testing accuracy. Notice that some solvers scale the SVM formulation, so we adjust their regularization parameter C accordingly.[2] With the best parameter setting, SVM (L1 and L2) and logistic regression give comparable generalization performances. In Figure 3, we present the testing accuracy along the training time. As an accurate solution of the SVM optimization problem does not imply the best testing accuracy, some implementations achieve higher accuracy before reaching the minimal function value. Below we give some observations of the experiments.

We do not include CD in Figure 3, because CDPER is better than it in almost all situations. One may ask if simply shuffling features once in the beginning can give similar performances to CDPER. Moreover, we can apply permutation schemes to CMLS as well. In Section 6.1, we give a detailed discussion on the issue of feature permutations.

Regarding the online setting of randomly selecting only one feature at each step (Algorithm 3), we find that results are similar to those of CDPER.

From the experimental results, CDPER converges faster than CMLS. Both are coordinate descent methods, and the cost per iteration is similar. However, CMLS suffers from lengthy iterations because its modified Newton method takes a conservative step size. In Figure 3(d), the testing accuracy even does not reach a reasonable value after 30 seconds. Conversely, CDPER usually uses full Newton steps, so it converges faster. For example, CDPER takes the full Newton step in 99.997% inner iterations for solving rcv1 (we check up to 5.96 seconds).

Though Pegasos is efficient for several data sets, the testing accuracy is sometimes unstable (see Figure 3(c)). As Pegasos only subsamples one training data to update $\mathbf{w}^k$, it is influenced more by noisy data. We also observe slow final convergence on the function value. This slow convergence may make the selection of stopping conditions (maximal number of iterations for Pegasos) more difficult.

Finally, compared to TRON and TRON-LR, CDPER is more efficient to yield good testing accuracy (See Table 2 and Figure 3); however, if we check the value $||\nabla f(\mathbf{w}^k)||$, Figure 2 shows that TRON converges faster in the end. This result is consistent with what we discussed in Section 1 on distinguishing various optimization methods. We indicated that a Newton method (where TRON is) has fast final convergence. Unfortunately, since the cost per TRON iteration is high, and the Newton direction is not effective in the beginning, TRON is less efficient in the early stage of the optimization procedure.

5.3 Stopping Conditions

In this section, we discuss stopping conditions of our algorithm and other existing methods. In solving a strictly convex optimization problem, the norm of gradients is often considered in the stopping condition. The reason is that

$$\|\nabla f(\mathbf{w})\| = 0 \iff \mathbf{w} \text{ is the global minimum.}$$

For example, TRON checks whether the norm of gradient is small enough for stopping. However, as our coordinate descent method updates one component of $\mathbf{w}$ at each inner iteration, we have only $D_i'(0) = \nabla f(\mathbf{w}^{k,i})_i, i = 1,\ldots,n$. Theorem 2 shows that $\mathbf{w}^{k,i} \to \mathbf{w}^*$, so we have

$$D_i'(0) = \nabla f(\mathbf{w}^{k,i})_i \to 0, \forall i.$$

2. The objective function of Pegasos and (2) are equivalent by setting $\lambda = 1/(C \times \text{number of instances})$, where λ is the penalty parameter used by Shalev-Shwartz et al. (2007).

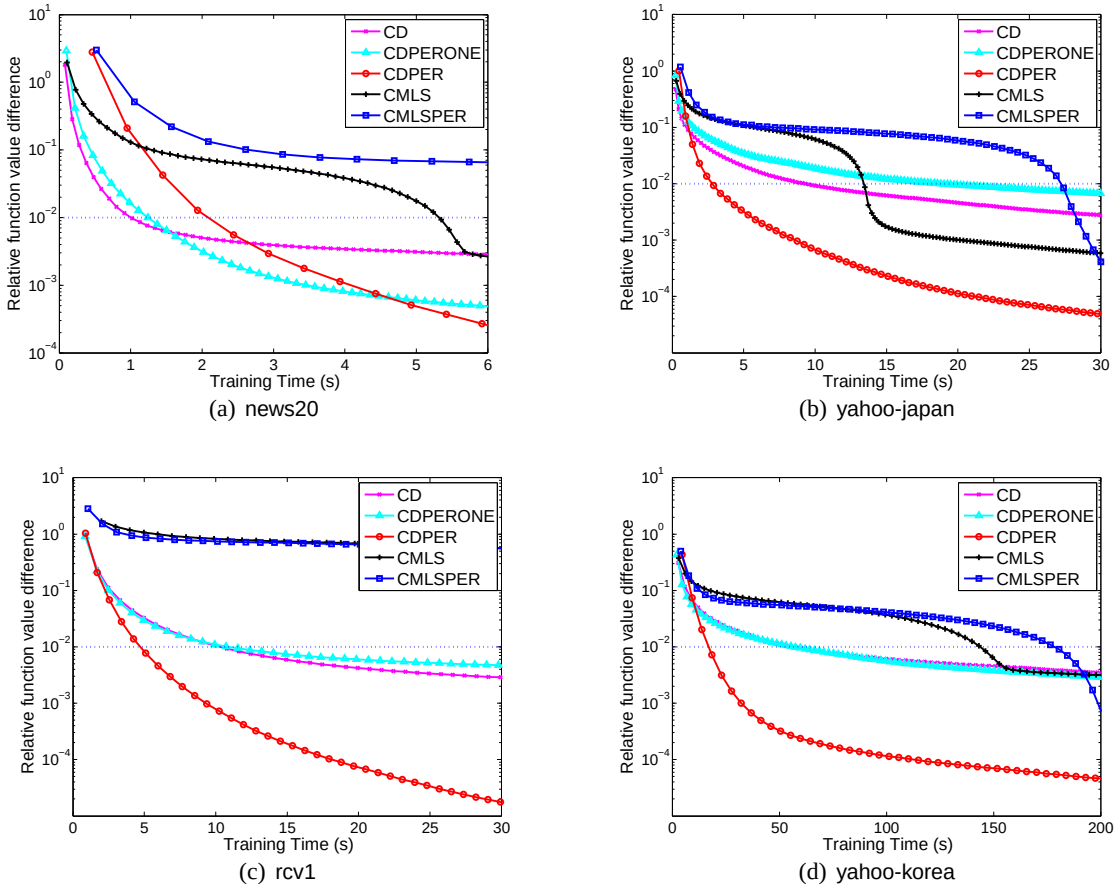


Figure 4: Results of different orders of sub-problems at each outer iteration. We present time versus the relative difference of the objective value to the minimum. Time is in second.

Therefore, by storing $D_i(0), \forall i$, at the end of the kth iteration, one can check if $\sum_{i=1}^{n} D_i'(0)^2$ or $\max_i |D_i'(0)|$ is small enough. For Pegasos, we mentioned in Section 4.1 that one may need a maximal number of iterations as the stopping condition due to the lack of function/gradient information. Another possible condition is to check the validation accuracy. That is, the training procedure terminates after reaching a stable validation accuracy value.

6. Discussion and Conclusions

In this section, we discuss some related issues and give conclusions.

6.1 Order of Sub-problems at Each Outer Iteration

In Section 5.2, we show that a random order of the sub-problems helps our coordinate descent method to converge faster in most cases. In this section, we give detailed experiments. Following the same setting in Figure 1, we compare our coordinate descent method with/without permutation

of sub-problems (CDPER and CD), with permutation only once before training (CDPERONE), and CMLS with/without permuting sub-problems (CMLS and CMLSPER). Figure 4 shows the relative difference of the objective value to the minimum along time. Overall, CDPER converges faster than CDPERONE and CD, but CMLSPER does not improve over CMLS much.

With the permutation of features at each iteration, the cost per CDPER iteration is slightly higher than CD, but CDPER requires much fewer iterations to achieve a similar accuracy value. This result seems to indicate that if the sub-problem order is fixed, the update of variables becomes slower. However, as CD sequentially accesses features, it has better data locality in the computer memory hierarchy. An example is news20 in Figure 4(a). As the number of features is much larger than the number of instances, two adjacent sub-problems of CDPER may access two very far away features. Then the cost per CD iteration is only 1/5 of CDPER, so CD is better in the beginning. CDPER catches up in the end due to its faster convergence.

For CMLS and CMLSPER, the latter is only faster in the final stage (see the right end of Figures 4(b) and 4(d)). Since the function reduction of CMLS (or CMLSPER) is slow, the advantage of doing permutations appears after long training time.

The main difference between CDPERONE and CDPER is that the former only permutes features once in the beginning. Figure 4 clearly shows that CDPER is better than CDPERONE, so permuting features only once is not enough. If we compare CD and CDPERONE, there is no definitive winner. This result seems to indicate that feature ordering affects the performance of the coordinate descent method. By using various feature orders, CDPER avoids taking a bad one throughout all iterations.

6.2 Coordinate Descents for Logistic Regression

We can apply the proposed coordinate descent method to solve logistic regression, which is twice differentiable. An earlier study of using coordinate decent methods for logistic regression/maximum entropy is by Miroslav et al. (2004). We compare an implementation with TRON-LR. Surprisingly, our method is not better in most cases. Experiments show that for training rcv1, our coordinate descent method takes 93.1 seconds to reduce the objective value to within 1% of the optimal value, while TRON-LR takes 27.9 seconds. Only for yahoo-japan and yahoo-korea, where TRON-LR is slow (see Figure 3), the coordinate descent method is competitive. This result is contrast to earlier experiments for L2-SVM, where the coordinate descent method more quickly obtains a useful model than TRON. We give some explanations below.

With the logistic loss, the objective function is (4). The single-variable function $D_i(z)$ is non-linear, so we use Algorithm 2 to obtain an approximate minimum. To use the Newton direction, similar to $D_i'(z)$ and $D_i''(z)$ in (9) and (10), we need

$$D_i'(0) = \nabla_i f(\mathbf{w}) = w_i + C \sum_{j:x_{ji}\neq 0} \frac{-y_j x_{ji} e^{-y_j \mathbf{w}^T \mathbf{x}_j}}{1+e^{-y_j \mathbf{w}^T \mathbf{x}_j}},$$

$$D_i''(0) = \nabla_{ii}^2 f(\mathbf{w}) = 1 + C \sum_{j:x_{ji}\neq 0} \frac{x_{ji}^2 e^{-y_j \mathbf{w}^T \mathbf{x}_j}}{(1+e^{-y_j \mathbf{w}^T \mathbf{x}_j})^2},$$

where we abbreviate $\mathbf{w}^{k,i}$ to $\mathbf{w}$, and use $y_j = \pm 1$. Then $|\{j \mid x_{ji} \neq 0\}|$ exponential operations are conducted. If we assume that $\lambda = 1$ satisfies the sufficient decrease condition (12), then the cost per outer iteration is

$$O(\#\text{nz}) + (\#\text{nz exponential operations}). \tag{32}$$

This complexity is the same as (19) for L2-SVM. However, since each exponential operation is expensive (equivalent to tens of multiplications/divisions), in practice (32) is much more time consuming. For TRON-LR, a trust region Newton method, it calculates $\nabla f(\mathbf{w})$ and $\nabla^2 f(\mathbf{w})$ at the beginning of each iteration. Hence l exponential operations are needed for $\exp(-y_j \mathbf{w}^T \mathbf{x}_j), j = 1, \ldots, l$. From (27), the cost per iteration is

$$O(\#\text{nz}) \times (\#\text{ conjugate gradient iterations}) + (l \text{ exponential operations}). \tag{33}$$

Since $l \ll \#\text{nz}$, exponential operations are not significant in (33). Therefore, the cost per iteration of applying trust region Newton methods to L2-SVM and logistic regression does not differ much. In contrast, (32) shows that coordinate descent methods are less suitable for logistic regression than L2-SVM. However, we may avoid expensive exponential operations if all the elements of $\mathbf{x}_j$ are either 0 or the same constant. By storing $\exp(-y_j \mathbf{w}^T \mathbf{x}_j), j = 1, \ldots, l$, one updates $\exp(-y_j (\mathbf{w}^{k,i})^T \mathbf{x}_j)$ by multiplying it by $\exp(-z y_j x_{ji})$. Using $y_j = \pm 1$, $\exp(-z y_j x_{ji}) = (\exp(-z x_{ji}))^{y_j}$. As x_{ji} is zero or a constant for all j, the number of exponential operations per inner iteration is reduced to one. In addition, applying fast approximations of exponential operations such as Schraudolph (1999) may speed up the coordinate descent method for logistic regression.

6.3 Conclusions

In summary, we propose and analyze a coordinate descent method for large-scale linear L2-SVM. The new method possesses sound optimization properties. The method is suitable for data with an easy access of any feature. Experiments indicate that our method is more stable and efficient than most existing algorithms. We plan to extend our work to other challenging problems such as training large data which can not fit into memory.

Acknowledgments

This work was supported in part by the National Science Council of Taiwan grant 95-2221-E-002-205-MY3. The authors thank Associate Editor and reviewers for helpful comments.

Appendix A. Proofs

In this section, we prove theorems appeared in the paper. First, we discuss some properties of our objective function $f(\mathbf{w})$. Consider the following piecewise quadratic strongly convex function:

$$g(\mathbf{s}) = \frac{1}{2}\mathbf{s}^T \mathbf{s} + C\|(A\mathbf{s} - \mathbf{h})_+\|^2, \tag{34}$$

where $(\cdot)_+$ is the operator that replaces negative components of a vector with zeros. Mangasarian (2002) proves the following inequalities for all $\mathbf{s}, \mathbf{v} \in R^n$:

$$(\nabla g(\mathbf{s}) - \nabla g(\mathbf{v}))^T (\mathbf{s} - \mathbf{v}) \geq \|\mathbf{s} - \mathbf{v}\|^2, \tag{35}$$

$$|g(\mathbf{v}) - g(\mathbf{s}) - \nabla g(\mathbf{s})^T (\mathbf{v} - \mathbf{s})| \leq \frac{K}{2}\|\mathbf{v} - \mathbf{s}\|^2, \tag{36}$$

where

$$K = 1 + 2C\|A\|_2^2.$$

Our objective function $f(\mathbf{w})$ is a special case of (34) with $A = YX$ (X is defined in Eq. 20) and $\mathbf{h} = -\mathbf{1}$, where Y is a diagonal matrix with $Y_{jj} = y_j$, $j = 1,\ldots,l$ and $\mathbf{1}$ is the vector of all ones. With $y_i = \pm 1$, $f(\mathbf{w})$ satisfies (35) and (36) with

$$K = 1 + 2C\|X\|_2^2. \tag{37}$$

To derive properties of the subproblem $D_i(z)$ (defined in Eq. 8), we set

$$A = -\begin{bmatrix} y_1 x_{1i} \\ \vdots \\ y_l x_{li} \end{bmatrix} \text{ and } \mathbf{h} = \begin{bmatrix} y_1 \mathbf{w}^T \mathbf{x}_1 - y_1 w_i x_{1i} - 1 \\ \vdots \\ y_l \mathbf{w}^T \mathbf{x}_l - y_l w_i x_{li} - 1 \end{bmatrix},$$

where we abbreviate $\mathbf{w}^{k,i}$ to $\mathbf{w}$. Since

$$D_i(z) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \frac{1}{2}w_i^2 + \left(\frac{1}{2}(z+w_i)^2 + \|(A(z+w_i) - \mathbf{h})_+\|^2\right),$$

the first and second terms of the above form are constants. Hence, $D_i(z)$ satisfies (35) and (36) with

$$K = 1 + 2C\sum_{j=1}^{l} x_{ji}^2.$$

We use H_i to denote $D_i(z)$'s corresponding K. This definition of H_i is the same as the one in (14). We then derive several lemmas.

Lemma 5 *For any $i \in \{1,\ldots,n\}$, and any $z \in R$,*

$$D_i'(0)z + \frac{1}{2}H_i z^2 \geq D_i(z) - D_i(0) \geq D_i'(0)z + \frac{1}{2}z^2. \tag{38}$$

Proof There are two inequalities in (38). The first inequality directly comes from (36) using $D_i(z)$ as $g(\mathbf{s})$ and H_i as K. To derive the second inequality, if $z < 0$, using $D_i(z)$ as $g(\mathbf{s})$ in (35) yields

$$D_i'(z) \leq D_i'(0) + z.$$

Then,

$$D_i(z) - D_i(0) = -\int_{t=z}^{0} D_i'(t)dt \geq D_i'(0)z + \frac{1}{2}z^2.$$

The situation for $z \geq 0$ is similar.

■

Lemma 6 *There exists a unique optimum solution for* (3).

Proof From Weierstrass' Theorem, any continuous function on a compact set attains its minimum. We consider the level set $A = \{\mathbf{w} \mid f(\mathbf{w}) \leq f(\mathbf{0})\}$. If A is not bounded, there is a sub-sequence $\{\mathbf{w}^k\} \subset A$ such that $\|\mathbf{w}^k\| \to \infty$. Then

$$f(\mathbf{w}^k) \geq \frac{1}{2}\|\mathbf{w}^k\|^2 \to \infty.$$

This contradicts $f(\mathbf{w}^k) \leq f(\mathbf{0})$, so A is bounded. Thus there is at least one optimal solution for (3). Combining with the strict convexity of (3), a unique global optimum exists. ■

A.1 Proof of Theorem 1

Proof By Lemma 5, we let $z = \lambda d$ and have

$$\begin{aligned} & D_i(\lambda d) - D_i(0) + \sigma\lambda^2 d^2 \\ \leq\ & D_i'(0)\lambda d + \frac{1}{2}H_i\lambda^2 d^2 + \sigma\lambda^2 d^2 \\ =\ & -\lambda\frac{D_i'(0)^2}{D_i''(0)} + \frac{1}{2}H_i\lambda^2\frac{D_i'(0)^2}{D_i''(0)^2} + \sigma\lambda^2\frac{D_i'(0)^2}{D_i''(0)^2} \\ =\ & \lambda\frac{D_i'(0)^2}{D_i''(0)}\left(\lambda(\frac{H_i/2+\sigma}{D_i''(0)}) - 1\right). \end{aligned} \tag{39}$$

If we choose $\bar{\lambda} = \frac{D_i''(0)}{H_i/2+\sigma}$, then for $\lambda \leq \bar{\lambda}$, (39) is non-positive. Therefore, (12) is satisfied for all $0 \leq \lambda \leq \bar{\lambda}$. ■

A.2 Proof of Theorem 2 (convergence of Algorithm 1)

Proof By setting $\pi_k(i) = i$, this theorem is a special case of Theorem 3. ■

A.3 Proof of Theorem 3 (Convergence of Generalized Algorithm 1)

Proof To begin, we define 1-norm and 2-norm of a vector $\mathbf{w} \in R^n$:

$$\|\mathbf{w}\|_1 = \sum_{i=1}^{n}|w_i|, \quad \|\mathbf{w}\|_2 = \sqrt{\sum_{i=1}^{n} w_i^2}.$$

The following inequality is useful:

$$\|\mathbf{w}\|_2 \leq \|\mathbf{w}\|_1 \leq \sqrt{n}\|\mathbf{w}\|_2, \quad \forall \mathbf{w} \in R^n. \tag{40}$$

By Theorem 1, any $\lambda \in [\beta\bar{\lambda}, \bar{\lambda}]$ satisfies the sufficient decrease condition (12), where $\beta \in (0,1)$ and $\bar{\lambda}$ is defined in (14). Since Algorithm 2 selects λ by trying $\{1, \beta, \beta^2, \dots\}$, the value λ selected by Algorithm 2 satisfies

$$\lambda \geq \beta\bar{\lambda} = \frac{\beta}{H_i/2+\sigma}D''_{\pi_k(i)}(0).$$

This and (13) suggest that the step size $z = \lambda d$ in Algorithm 2 satisfies

$$|z| = \lambda\left|\frac{-D'_{\pi_k(i)}(0)}{D''_{\pi_k(i)}(0)}\right| \geq \frac{\beta}{H_i/2+\sigma}|D'_{\pi_k(i)}(0)|. \tag{41}$$

Assume

$$H = \max(H_1, \dots, H_n) \text{ and } \gamma = \frac{\beta}{H/2+\sigma}. \tag{42}$$

We use $\mathbf{w}$ and $f(\mathbf{w})$ to rewrite (41):

$$|w^{k,i+1}_{\pi_k(i)} - w^{k,i}_{\pi_k(i)}| \geq \gamma |\nabla f(\mathbf{w}^{k,i})_{\pi_k(i)}|, \tag{43}$$

where we use the fact

$$D'_{\pi_k(i)}(0) = \nabla f(\mathbf{w}^{k,i})_{\pi_k(i)}.$$

Taking the summation of (43) from $i = 1$ to n, we have

$$\begin{aligned}
\|\mathbf{w}^{k+1} - \mathbf{w}^k\|_1 &\geq \gamma \sum_{i=1}^{n} |\nabla f(\mathbf{w}^{k,i})_{\pi_k(i)}| \\
&\geq \gamma \sum_{i=1}^{n} (|\nabla f(\mathbf{w}^{k,1})_{\pi_k(i)}| - |\nabla f(\mathbf{w}^{k,i})_{\pi_k(i)} - \nabla f(\mathbf{w}^{k,1})_{\pi_k(i)}|) \\
&= \gamma \left(\|\nabla f(\mathbf{w}^{k,1})\|_1 - \sum_{i=1}^{n} |\nabla f(\mathbf{w}^{k,i})_{\pi_k(i)} - \nabla f(\mathbf{w}^{k,1})_{\pi_k(i)}| \right).
\end{aligned} \tag{44}$$

By the definition of $f(\mathbf{w})$ in (3),

$$\nabla f(\mathbf{w}) = \mathbf{w} - 2C \sum_{j=1}^{l} y_j \mathbf{x}_j \max(1 - y_j \mathbf{w}^T \mathbf{x}_j, 0).$$

With $y_j = \pm 1$,

$$\begin{aligned}
&\sum_{i=1}^{n} |\nabla f(\mathbf{w}^{k,i})_{\pi_k(i)} - \nabla f(\mathbf{w}^{k,1})_{\pi_k(i)}| \\
\leq &\sum_{i=1}^{n} \left(|w^{k,i}_{\pi_k(i)} - w^{k,1}_{\pi_k(i)}| + 2C \sum_{j=1}^{l} |x_{j\pi_k(i)}| \, |(\mathbf{w}^{k,i})^T \mathbf{x}_j - (\mathbf{w}^{k,1})^T \mathbf{x}_j| \right) \\
\leq &\sum_{i=1}^{n} \left(|w^{k+1}_{\pi_k(i)} - w^{k}_{\pi_k(i)}| + 2C \sum_{j=1}^{l} |x_{j\pi_k(i)}| \sum_{q=1}^{n} |x_{jq}| \, |w^{k,i}_q - w^{k,1}_q| \right) \\
= &\|\mathbf{w}^{k+1} - \mathbf{w}^k\|_1 + 2C \sum_{i=1}^{n} \sum_{j=1}^{l} \sum_{q=1}^{n} |x_{j\pi_k(i)}| \, |x_{jq}| \, |w^{k,i}_q - w^{k,1}_q| \\
\leq &\|\mathbf{w}^{k+1} - \mathbf{w}^k\|_1 + 2C \sum_{q=1}^{n} |w^{k+1}_q - w^k_q| \sum_{i,j: x_{j\pi_k(i)} \neq 0} P^2 \\
= &(1 + 2CP^2(\#\text{nz})) \|\mathbf{w}^{k+1} - \mathbf{w}^k\|_1,
\end{aligned} \tag{45}$$

where P is defined in (6). From (44) and (45), we have

$$\|\mathbf{w}^{k+1} - \mathbf{w}^k\|_1 \geq \frac{\gamma}{1 + \gamma + 2\gamma CP^2(\#\text{nz})} \|\nabla f(\mathbf{w}^{k,1})\|_1.$$

With (40),

$$\begin{aligned}
&\|\mathbf{w}^{k+1} - \mathbf{w}^k\|_2 \geq \frac{1}{\sqrt{n}} \|\mathbf{w}^{k+1} - \mathbf{w}^k\|_1 \\
\geq &\frac{\gamma}{\sqrt{n}(1 + \gamma + 2\gamma CP^2(\#\text{nz}))} \|\nabla f(\mathbf{w}^k)\|_1 \geq \frac{\gamma}{\sqrt{n}(1 + \gamma + 2\gamma CP^2(\#\text{nz}))} \|\nabla f(\mathbf{w}^k)\|_2.
\end{aligned} \tag{46}$$

From Lemma 6, there is a unique global optimum $\mathbf{w}^*$ for (3). The optimality condition shows that

$$\nabla f(\mathbf{w}^*) = \mathbf{0}. \tag{47}$$

From (35) and (47),

$$\|\mathbf{w}^k - \mathbf{w}^*\|_2 \le \|\nabla f(\mathbf{w}^k) - \nabla f(\mathbf{w}^*)\|_2 = \|\nabla f(\mathbf{w}^k)\|_2. \tag{48}$$

With (46),

$$\|\mathbf{w}^{k+1} - \mathbf{w}^k\|_2 \ge \tau \|\mathbf{w}^k - \mathbf{w}^*\|_2, \quad \text{where } \tau = \frac{\gamma}{\sqrt{n}(1+\gamma+2\gamma CP^2(\#\text{nz}))}. \tag{49}$$

From (12) and (49),

$$\begin{aligned} f(\mathbf{w}^k) - f(\mathbf{w}^{k+1}) &= \sum_{i=1}^{n} (f(\mathbf{w}^{k,i}) - f(\mathbf{w}^{k,i+1})) \\ &\ge \sum_{i=1}^{n} \sigma (w^{k,i+1}_{\pi_k(i)} - w^{k,i}_{\pi_k(i)})^2 = \sigma \|\mathbf{w}^{k+1} - \mathbf{w}^k\|_2^2 \ge \sigma\tau^2 \|\mathbf{w}^k - \mathbf{w}^*\|_2^2. \end{aligned}$$

By (36) and (47),

$$f(\mathbf{w}^k) - f(\mathbf{w}^*) \le \frac{K}{2} \|\mathbf{w}^k - \mathbf{w}^*\|_2^2, \tag{50}$$

where K is defined in (37). Therefore, we have

$$f(\mathbf{w}^k) - f(\mathbf{w}^{k+1}) \ge \frac{2\sigma\tau^2}{K} (f(\mathbf{w}^k) - f(\mathbf{w}^*)).$$

This is equivalent to

$$(f(\mathbf{w}^k) - f(\mathbf{w}^*)) + (f(\mathbf{w}^*) - f(\mathbf{w}^{k+1})) \ge \frac{2\sigma\tau^2}{K} (f(\mathbf{w}^k) - f(\mathbf{w}^*)).$$

Finally, we have

$$f(\mathbf{w}^{k+1}) - f(\mathbf{w}^*) \le (1 - \frac{2\sigma\tau^2}{K})(f(\mathbf{w}^k) - f(\mathbf{w}^*)). \tag{51}$$

With $\tau \le 1$ from (49), $K \ge 1$ from (37) and $\sigma < 1/2$, we have $2\sigma\tau^2/K < 1$. Hence, (51) ensures that $f(\mathbf{w}^k)$ approaches $f(\mathbf{w}^*)$.

From (51), $\{f(\mathbf{w}^k)\}$ converges to $f(\mathbf{w}^*)$. We can then prove that $\{\mathbf{w}^k\}$ globally converges to $\mathbf{w}^*$. If this result does not hold, there is a sub-sequence $\{\mathbf{w}^k\}_M$ converging to a point $\bar{\mathbf{w}} \ne \mathbf{w}^*$. However, Lemma 6 shows that $f(\bar{\mathbf{w}}) > f(\mathbf{w}^*)$, so $\lim_{k \in M} f(\mathbf{w}^k) > f(\mathbf{w}^*)$, a contradiction.

Let $\mu = 2\sigma\tau^2/K$, (51) implies

$$f(\mathbf{w}^k) - f(\mathbf{w}^*) \le (1-\mu)^k (f(\mathbf{w}^0) - f(\mathbf{w}^*)), \ \forall k.$$

To achieve an ε-accurate solution, we need the right-hand side to be smaller than ε. Thus,

$$k \ge \frac{\log(f(\mathbf{w}^0) - f(\mathbf{w}^*)) + \log(1/\varepsilon)}{-\log(1-\mu)}.$$

From the inequality

$$\log(1-x) \leq -x \text{ if } x < 1,$$

we have

$$k \geq \frac{\log(f(\mathbf{w}^0) - f(\mathbf{w}^*)) + \log(1/\varepsilon)}{\mu}. \tag{52}$$

In following we discuss the order of $\mu^{-1} = \frac{K}{2\sigma\tau^2}$. From (37),

$$K = 2C\|X\|_2^2 + 1 \leq 2C\|X\|_F^2 + 1 \leq 2CP^2(\#\text{nz}) + 1, \tag{53}$$

where $\|\cdot\|_F$ is the Frobenious norm. From (49),

$$\tau^{-1} = \sqrt{n}\left(\frac{1}{\gamma} + 2CP^2(\#\text{nz}) + 1\right).$$

Since $\gamma = \frac{\beta}{\sigma + H/2}$, from (14) and (42) we have

$$\gamma^{-1} = O(lCP^2 + 1) \leq O(CP^2(\#\text{nz}) + 1). \tag{54}$$

As #nz is usually large, we omit the constant term $O(1)$ in the following discussion. Then $\tau^{-1} = O(\sqrt{n}CP^2(\#\text{nz}))$. Thus,

$$\mu^{-1} = \frac{K}{2\sigma\tau^2} = O(nC^3P^6(\#\text{nz})^3). \tag{55}$$

From (52) and (55), Algorithm 1 obtains an ε-accurate solution in

$$O\left(nC^3P^6(\#\text{nz})^3\log(1/\varepsilon)\right)$$

iterations. ■

A.4 Proof of Theorem 4 (Linear Convergence of the Online Setting)

Proof To begin, we denote the expectation value of a function g of a random variable y to be

$$E_y(g(y)) = \sum_y P(y)g(y).$$

Then for any vector $\mathbf{s} \in R^n$ and a random variable I where

$$P(I = i) = \frac{1}{n},\ \forall i \in \{1, \ldots, n\},$$

we have

$$E_I(s_I^2) = \sum_{i=1}^{n} \frac{s_i^2}{n} = \frac{1}{n}\|\mathbf{s}\|_2^2. \tag{56}$$

At each iteration k ($k = 0, 1, \ldots$) of Algorithm 3, we randomly choose one index i_k and update $\mathbf{w}^k$ to $\mathbf{w}^{k+1}$. The expected function value after iteration k can be represented as

$$E_{i_0,\ldots,i_{k-1},i_k}(f(\mathbf{w}^{k+1})).$$

From (12), (43), (56), (48), and (50), we have

$$\begin{aligned}
& E_{i_0,\ldots,i_{k-1}} E_{i_k}(f(\mathbf{w}^k) - f(\mathbf{w}^{k+1})) \\
\geq\ & \sigma E_{i_0,\ldots,i_{k-1}} E_{i_k}(|w_{i_k}^{k+1} - w_{i_k}^k|^2) \\
\geq\ & \sigma\gamma^2 E_{i_0,\ldots,i_{k-1}} E_{i_k}(|\nabla f(\mathbf{w}^k)_{i_k}|^2) \\
=\ & \frac{\sigma\gamma^2}{n} E_{i_0,\ldots,i_{k-1}}(\|\nabla f(\mathbf{w}^k)\|_2^2) \\
\geq\ & \frac{\sigma\gamma^2}{n} E_{i_0,\ldots,i_{k-1}}(\|\mathbf{w}^k - \mathbf{w}^*\|_2^2) \\
\geq\ & \frac{2\sigma\gamma^2}{nK} E_{i_0,\ldots,i_{k-1}}(f(\mathbf{w}^k) - f(\mathbf{w}^*)).
\end{aligned}$$

This is equivalent to

$$E_{i_0,\ldots,i_k}(f(\mathbf{w}^{k+1})) - f(\mathbf{w}^*) \leq \left(1 - \frac{2\sigma\gamma^2}{nK}\right)\left(E_{i_0,\ldots,i_{k-1}}(f(\mathbf{w}^k)) - f(\mathbf{w}^*)\right).$$

From Markov inequality $P(|Z| \geq a) \leq E(|Z|)/a$ for any random variable Z and the fact $f(\mathbf{w}^{k+1}) \geq f(\mathbf{w}^*)$, we have

$$P\left(f(\mathbf{w}^k) - f(\mathbf{w}^*) \geq \varepsilon\right) \leq E\left(f(\mathbf{w}^k) - f(\mathbf{w}^*)\right)/\varepsilon. \tag{57}$$

To achieve an ε-accurate solution with confidence $1-\delta$, we need the right-hand side of (57) to be less than δ. This indicates the iteration number k must satisfy

$$E_{i_0,\ldots,i_{k-1}}(f(\mathbf{w}^k)) - f(\mathbf{w}^*) \leq (f(\mathbf{w}^0) - f(\mathbf{w}^*))\left(1 - \frac{2\sigma\gamma^2}{nK}\right)^k \leq \varepsilon\delta.$$

By a derivation similar to Theorem 3, we can show that after $O(\frac{nK}{\sigma\gamma^2}\log(\frac{1}{\delta\varepsilon}))$ iterations, with confidence $1-\delta$, we obtain an ε-accurate solution. From (53) and (54), we have

$$K \leq 2CP^2(\#\text{nz}) + 1 \text{ and } \gamma^{-1} = O(lCP^2 + 1).$$

Therefore,

$$O\left(\frac{nK}{\sigma\gamma^2}\log(\frac{1}{\delta\varepsilon})\right) = O\left(nl^2C^3P^6(\#\text{nz})\log(\frac{1}{\delta\varepsilon})\right).$$

■

References

Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA 02178-9998, second edition, 1999.

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.

Leon Bottou. Stochastic gradient descent examples, 2007. http://leon.bottou.org/projects/sgd.

Leon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

Iain S. Duff, Roger G. Grimes, and John G. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15:1–14, 1989.

Luigi Grippo and Marco Sciandrone. Globally convergent block-coordinate techniques for unconstrained optimization. *Optimization Methods and Software*, 10:587–637, 1999.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*, 2008. URL http://www.csie.ntu.edu.tw/~cjlin/papers/cddual.pdf. Software available at http://www.csie.ntu.edu.tw/~cjlin/liblinear.

Thorsten Joachims. Training linear SVMs in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2006.

S. Sathiya Keerthi and Dennis DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650. URL http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf. Software available at http://www.csie.ntu.edu.tw/~cjlin/liblinear.

Zhi-Quan Luo and Paul Tseng. On the convergence of coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.

Olvi L. Mangasarian. A finite Newton method for classification. *Optimization Methods and Software*, 17(5):913–929, 2002.

Dukík Miroslav, Steven J. Phillips, and Robert E. Schapire. Performance guarantees for regularized maximum entropy density estimation. In *Proceedings of the 17th Annual Conference on Computational Learning Theory*, pages 655–662, New York, 2004. ACM press.

Gunnar Rätsch, Sebastian Mika, and Manfred K. Warmuth. On the convergence of leveraging. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 487–494. MIT Press, Cambridge, MA, 2002.

Nicol N. Schraudolph. A fast, compact approximation of the exponential function. *Neural Computation*, 11:853–862, 1999.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007.

Alex J. Smola, S V N Vishwanathan, and Quoc Le. Bundle methods for machine learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the 21th International Conference on Machine Learning (ICML)*, 2004.

Tong Zhang and Frank J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31, 2001.

Journal of Machine Learning Research 9 (2008) 1399-1435 Submitted 7/07; Revised 5/08; Published 7/08

Online Learning of Complex Prediction Problems Using Simultaneous Projections

Yonatan Amit MITMIT@CS.HUJI.AC.IL
The Hebrew University
School of Computer Science and Engineering, Givat Ram, Jerusalem, 91904, Israel

Shai Shalev-Shwartz SHAI@TTI-C.ORG
Toyota Technological Institute
E. 60th Street, Chicago, IL 60637, USA

Yoram Singer SINGER@GOOGLE.COM
Google Inc.
1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA

Editor: Manfred K. Warmuth

Abstract

We describe and analyze an algorithmic framework for online classification where each online trial consists of *multiple* prediction tasks that are tied together. We tackle the problem of updating the online predictor by defining a projection problem in which each prediction task corresponds to a single linear constraint. These constraints are tied together through a single slack parameter. We then introduce a general method for approximately solving the problem by projecting *simultaneously* and independently on each constraint which corresponds to a prediction sub-problem, and then averaging the individual solutions. We show that this approach constitutes a feasible, albeit not necessarily optimal, solution of the original projection problem. We derive concrete simultaneous projection schemes and analyze them in the mistake bound model. We demonstrate the power of the proposed algorithm in experiments with synthetic data and with multiclass text categorization tasks.

Keywords: online learning, parallel computation, mistake bounds, structured prediction

1. Introduction

We discuss and analyze an algorithmic framework for complex prediction problems in the online learning model. Our construction unifies various complex prediction tasks by considering a setting in which at each trial the learning algorithm should make multiple binary decisions. We present a simultaneous online update rule that uses the entire set of binary examples received at each trial while retaining the simplicity of algorithms whose update is based on a single binary example.

Online learning is performed in a sequence of consecutive trials. At the beginning of each trial, the algorithm first receives an instance and is required to make a prediction in some complex domain. The prediction is generated using an hypothesis constructed by the algorithm. Once the algorithm makes a prediction it receives the correct target and is allowed to update its hypothesis. In this paper we consider an online learning model in which the complex prediction task can be cast as multiple binary decisions. Such a view is common in Multiclass categorization tasks, for example in Crammer et al. (2006), Crammer and Singer (2003), and Allwein et al. (2000). There,

the complex problem of predicting which label out of k possible labels is the correct label is cast as a set of binary prediction problems, each of which focuses on two labels.

Previous approaches to this construction can be roughly divided into two paradigms. The first paradigm, which we term the *max update*, tackles the problem by selecting a *single* binary problem and updating the algorithm's hypothesis based on that problem solely. While this approach is sub-optimal, it is often very simple to implement and quite effective in practice. The second approach considers all the binary problems and incorporates the entire information contained for updating the hypothesis. The second approach thus performs an optimal update at the price of often incurring higher computational costs.

We introduce a third approach, which enjoys the simplicity and performance of the max-update approach, while incorporating information expressed in *all* binary problems. Our family of algorithms achieves this goal by considering each instance *separately* and acting *simultaneously*. An update is constructed for each binary sub-problem, and then all the updates are combine together to form the new online hypothesis. As we show in the sequel, the update rule due to each binary example amounts to a projection operation. We thus denote our approach as the *simultaneous projections* approach.

We propose a simple, general, and efficient framework for online learning of a wide variety of complex problems. We do so by casting the online update task as an optimization problem in which the newly devised hypothesis is required to be close to the current hypothesis while attaining a small loss on *multiple* binary prediction problems. Casting the online learning task as a sequence of instantaneous optimization problems was first suggested and analyzed by Kivinen and Warmuth (1997) for binary classification and regression problems. In our optimization-based approach, the complex decision problem is cast as an optimization problem that consists of *multiple* linear constraints each of which represents a single binary example. These constraints are tied through a *single* slack variable whose role is to assess the *overall* prediction quality for the complex problem.

The max-update approach described above selects a single binary example, which translates into a single constraint. Performing the update thus becomes a simple projection task, where an analytical solution can often be easily devised. In contrast, the optimal update seeks the optimal solution of the instantaneous optimization problem. However, in the general case no analytical solution can be found, and the algorithm is required to resort to a full scale numeric solver.

We describe and analyze a family of two-phase algorithms. In the first phase, the algorithms solve *simultaneously* multiple sub-problems. Each sub-problem distills to an optimization problem with a *single* linear constraint from the original multiple-constraints problem. The simple structure of each single-constraint problem results in an analytical solution, which is efficiently computable. In the second phase, the algorithms take a convex combination of the independent solutions to obtain a solution for the multiple-constraints problem. We further explore the structure of our problem and attain an update form that combines the two phases while maintaining the simplicity of the simultaneous projection scheme. The end result is an approach whose time complexity and mistake bounds are equivalent to approaches which solely deal with the worst-violating constraint (Crammer et al., 2006). In practice, though, the performance of the simultaneous projection framework is much better than update schemes that are based on a single-constraint .

We introduce an additive and multiplicative variants of our framework. The additive framework extends additive algorithms such as the Perceptron (Rosenblatt, 1958) and the family of Passive-Aggressive algorithms (Crammer et al., 2006) to our settings. We then present a multiplicative family of simultaneous algorithms that extends the Winnow family of algorithms (Littlestone, 1988).

We further extend our model showing its applicability when working with a larger family of loss functions. Finally we present a unified analysis in the mistake bound model, based on the primal-dual analysis presented in Shalev-Shwartz and Singer (2006a). Our results are on par with the best known mistake bounds for multiclass algorithms.

1.1 Related Work

The task of multiclass categorization can be thought of as a specific case of our construction. In multiclass categorization the task is to predict a *single* label out of k possible outcomes. Our simultaneous projection approach is based on the fact that we can retrospectively (after receiving the correct label) cast the problem as the task of making $k-1$ binary decisions, each of which involves the correct label and one of the competing labels. Our framework then performs an update on each of the problems separately and then combines the updates to form a new hypothesis. The performance of the $k-1$ predictions is measured through a single loss function. Our approach stands in contrast to previously studied methods which can be roughly be partitioned into three paradigms. The first paradigm follows the max update paradigm presented above. For example, the algorithms by Crammer and Singer (2003) and Crammer et al. (2006) focus on the single, worst performing, derived sub-problem. While this approach adheres with the original structure of the problem, the resulting update mechanism is by construction sub-optimal as it oversees almost all of the constraints imposed by the complex prediction problem. (See also Shalev-Shwartz and Singer, 2006a, for analysis and explanation of the sub-optimality of this approach).

Since applying full scale numeric solvers in each online trial is usually prohibitive due to the high computational cost, the optimal paradigm for dealing with complex problems is to tailor a specific efficient solution for the problem on hand. While this approach yielded highly efficient learning algorithms for multiclass categorization problems (Crammer and Singer, 2003; Shalev-Shwartz and Singer, 2006b) and aesthetic solutions for structured output problems (Taskar et al., 2003; Tsochantaridis et al., 2004), devising these algorithms required dedicated efforts. Moreover, tailored solutions typically impose rather restrictive assumptions on the representation of the data in order to yield efficient algorithmic solutions.

The third (and probably the simplest) previously studied approach is to break the problem into multiple *decoupled* problems that are solved *independently*. Such translation effectively changes the problem definition. Thus, the simplicity of this approach also underscores its deficiency as it is detached from the original loss of the complex decision problem. Such an approach was used for instance for batch learning of multiclass support vector machines (Weston and Watkins, 1999) and boosting algorithms (Schapire and Singer, 1999). Decoupling approaches have further been extended to various ways. Hastie and Tibshirani (1998) considered construction of a binary problem for each pair of classes. In Dietterich and Bakiri (1995), Allwein et al. (2000) and Crammer and Singer (2002) error correcting output codes are applied to solve the multiclass problem as separate binary problems.

It is interesting to note that the methods for performing multiple projections simultaneously have been studied in a different context in the optimization community. Similar ideas which can be broadly characterized as row-action methods date back more than 50 years, see for example Hildreth (1957), Bregman (1967), and Pierro and Iusem (1986). These methods are used to find the optimal solution of a convex function subject to a very large number of constraints. The core idea behind row-action methods is to consider and isolate a few number of constraints and repeatedly perform

a projection on a small subset (typically a single constrain) until convergence. Pierro and Iusem (1986) introduced the concept of averaging to the general family of Bregmans methods, where the projection step is relaxed and the new solution is the *average* of the previous solution and the result of the projection. Censor and Zenios (1997) introduces a parallel version of the row-action methods. The parallel algorithms perform the projection step on each constraint separately and update the new solution to the average of all these projections. For further extensive description of row-action methods see Censor and Zenios (1997). Row-actions methods have recently received attention in the learning community, for problems such as finding the optimal solution of SVM. For instance the SMO technique of Platt (1998) can be viewed as a row-action optimization method that manipulates two constraints at a time. In this paper we take a different approach, and *decompose* a single complex constraint into multiple projections problem which are tied together through a single slack variable.

The rest of the paper is organized as follows. We start with a description of the problem setting in Sec. 2. In Sec. 3 we describe two complex decision tasks that can be tackled by our approach. A template algorithm for additive simultaneous projection in an online learning setting with multiple instances is described in Sec. 4. We propose concrete schemes for selecting an update form in Sec. 5 and analyze our algorithms within the mistake bound model in Sec. 6. We extend our algorithm to a large family of losses in Sec. 7 and derive family of multiplicative algorithms in Sec. 8. We demonstrate the merits of our approach in a series of experiments with synthetic and real data sets in Sec. 9 and conclude in Sec. 10.

2. Problem Setting

In this section we introduce the notation used throughout the paper and formally describe our problem setting. We denote vectors by lower case bold face letters (e.g., $\mathbf{x}$ and $\boldsymbol{\omega}$) where the j'th element of $\mathbf{x}$ is denoted by x_j. We denote matrices by upper case bold face letters (e.g., $\mathbf{X}$), where the j'th row of $\mathbf{X}$ is denoted by $\mathbf{x}_j$. The set of integers $\{1,\ldots,k\}$ is denoted by $[k]$. Finally, we use the hinge function $[a]_+ = \max\{0,a\}$.

Online learning is performed in a sequence of trials. At trial t the algorithm receives a matrix $\mathbf{X}^t$ of size $k_t \times n$, where each row of $\mathbf{X}^t$ is an instance, and is required to make a prediction on the label associated with each instance. We denote the vector of predicted labels by $\hat{\mathbf{y}}^t$. We allow $\hat{y}^t_j$ to take any value in $\mathbb{R}$, where the actual label being predicted is $\mathrm{sign}(\hat{y}^t_j)$ and $|\hat{y}^t_j|$ is the confidence in the prediction. After making a prediction $\hat{\mathbf{y}}^t$ the algorithm receives the correct labels $\mathbf{y}^t$ where $y^t_j \in \{-1,1\}$ for all $j \in [k_t]$. In this paper we assume that the predictions in each trial are formed by calculating the inner product between a weight vector $\boldsymbol{\omega}^t \in \mathbb{R}^n$ with each instance in $\mathbf{X}^t$, thus $\hat{\mathbf{y}}^t = \mathbf{X}^t\boldsymbol{\omega}^t$. Our goal is to *perfectly* predict the entire vector $\mathbf{y}^t$. We thus say that the vector $\mathbf{y}^t$ was *imperfectly* predicted if there exists an outcome j such that $y^t_j \neq \mathrm{sign}(\hat{y}^t_j)$. That is, we suffer a unit loss on trial t if there exists j, such that $\mathrm{sign}(\hat{y}^t_j) \neq y^t_j$. Directly minimizing this combinatorial error is a computationally difficult task. Therefore, we use an adaptation of the *hinge-loss*, defined $\ell(\hat{\mathbf{y}}^t,\mathbf{y}^t) = \max_j \left[1 - y^t_j \hat{y}^t_j\right]_+$, as a proxy for the combinatorial error. The quantity $y^t_j \hat{y}^t_j$ is often referred to as the (signed) *margin* of the prediction and ties the correctness and the confidence in the prediction. We use $\ell(\boldsymbol{\omega}^t;(\mathbf{X}^t,\mathbf{y}^t))$ to denote $\ell(\hat{\mathbf{y}}^t,\mathbf{y}^t)$ where $\hat{\mathbf{y}}^t = \mathbf{X}^t\boldsymbol{\omega}^t$. We also denote the set of instances whose labels were predicted incorrectly by $\mathcal{M}^t = \{j \,|\, \mathrm{sign}(\hat{y}^t_j) \neq y^t_j\}$, and similarly the set of instances whose hinge-losses are greater than zero by $\Gamma^t = \{j \,|\, [1 - y^t_j \hat{y}^t_j]_+ > 0\}$.

3. Derived Problems

In this section we further explore the motivation for our problem setting by describing two different complex decision tasks and showing how they can be cast as special cases of our setting. We also would like to note that our approach can be employed in other prediction problems (see Sec. 10).

3.1 Multilabel Categorization

In the multilabel categorization task each instance is associated with a set of relevant labels from the set $[k]$. The multilabel categorization task can be cast as a special case of a ranking task in which the goal is to rank the relevant labels above the irrelevant ones. Many learning algorithms for this task employ class-dependent features (for example, see Schapire and Singer, 2000). For simplicity, assume that each class is associated with n features and denote by $\phi(\mathbf{x},r)$ the feature vector for class r. We would like to note that features obtained for different classes typically relay different information and are often substantially different.

A categorizer, or label ranker, is based on a weight vector ω. A vector ω induces a score for each class $\omega\cdot\phi(\mathbf{x},r)$ which, in turn, defines an ordering of the classes. A learner is required to build a vector ω that successfully ranks the labels according to their relevance, namely for each pair of classes (r,s) such that r is relevant while s is not, the class r should be ranked higher than the class s. Thus we require that $\omega\cdot\phi(\mathbf{x},r)>\omega\cdot\phi(\mathbf{x},s)$ for every such pair (r,s). We say that a label ranking is imperfect if there exists *any* pair (r,s) which violates this requirement. The loss associated with each such violation is $[1-(\omega\cdot\phi(\mathbf{x},r)-\omega\cdot\phi(\mathbf{x},s))]_+$ and the loss of the categorizer is defined as the maximum over the losses induced by the violated pairs. In order to map the problem to our setting, we define a virtual instance for every pair (r,s) such that r is relevant and s is not. The new instance is the n dimensional vector defined by $\phi(\mathbf{x},r)-\phi(\mathbf{x},s)$. The label associated with all of the instances is set to 1. It is clear that an imperfect categorizer makes a prediction mistake on at least one of the instances, and that the losses defined by both problems are the same.

3.2 Ordinal Regression

In the problem of ordinal regression an instance $\mathbf{x}$ is a vector of n features that is associated with a target rank $y\in[k]$. A learning algorithm is required to find a vector ω and k thresholds $b_1\leq\cdots\leq b_{k-1}\leq b_k=\infty$. The value of $\omega\cdot\mathbf{x}$ provides a score from which the prediction value can be defined as the smallest index i for which $\omega\cdot\mathbf{x}<b_i$, $\hat{y}=\min\{i|\omega\cdot\mathbf{x}<b_i\}$. In order to obtain a correct prediction, an ordinal regressor is required to ensure that $\omega\cdot\mathbf{x}\geq b_i$ for all $i<y$ and that $\omega\cdot\mathbf{x}<b_i$ for $i\geq y$. It is considered a prediction mistake if any of these constraints is violated. In order to map the ordinal regression task to our setting, we introduce $k-1$ instances. Each instance is a vector in $\mathbb{R}^{n+k-1}$. The first n entries of the vector are set to be the elements of $\mathbf{x}$, the remaining $k-1$ entries are set to $-\delta_{i,j}$. That is, the i'th entry in the j'th vector is set to -1 if $i=j$ and to 0 otherwise. The label of the first $y-1$ instances is 1, while the remaining $k-y$ instances are labeled as -1. Once we learned an expanded vector in $\mathbb{R}^{n+k-1}$, the regressor ω is obtained by taking the first n components of the expanded vector and the thresholds $b_1,\ldots,b_{k-1}$ are set to be the last $k-1$ elements. A prediction mistake of any of the instances corresponds to an incorrect rank in the original problem.

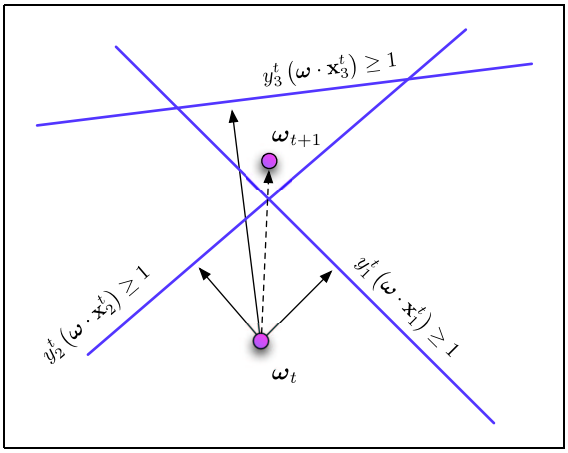


Figure 1: Illustration of the simultaneous projections algorithm: each instance casts a constraint on ω and each such constraint defines a halfspace of feasible solutions. We project on each halfspace in parallel and the new vector is a weighted average of these projections

4. Simultaneous Projection Algorithms

Recall that on trial t the algorithm receives a matrix, $\mathbf{X}^t$, of k_t instances, and predicts $\hat{\mathbf{y}}^t = \mathbf{X}^t \omega^t$. After performing its prediction, the algorithm receives the corresponding labels $\mathbf{y}^t$. Each instance-label pair casts a constraint on ω_t, $y_j^t \left(\omega^t \cdot \mathbf{x}_j^t\right) \geq 1$. If all the constraints are satisfied by ω^t then ω^{t+1} is set to be ω^t and the algorithm proceeds to the next trial. Otherwise, we would like to set ω^{t+1} as close as possible to ω^t while satisfying all constraints.

Such an aggressive approach may be sensitive to outliers and over-fitting. Thus, we allow some of the constraints to remain violated by introducing a trade-off between the change to ω^t and the loss attained on $(\mathbf{X}^t, \mathbf{y}^t)$. Formally, we would like to set ω^{t+1} to be the solution of the following minimization problem,

$$\min_{\omega \in \mathbb{R}^n} \tfrac{1}{2} \|\omega - \omega^t\|^2 + C\,\ell(\omega; (\mathbf{X}^t, \mathbf{y}^t)), \tag{1}$$

where C is a trade-off parameter. As we discuss below, this formalism effectively translates to a cap on the maximal change to ω^t. We rewrite the above optimization by introducing a *single* slack variable as follows:

$$\begin{aligned} &\min_{\omega \in \mathbb{R}^n, \xi \geq 0} \frac{1}{2} \left\|\omega - \omega^t\right\|^2 + C\xi \\ &\text{s.t.} \qquad \forall j \in [k_t]: \; y_j^t \left(\omega \cdot \mathbf{x}_j^t\right) \geq 1 - \xi \qquad \xi \geq 0 \end{aligned} \quad . \tag{2}$$

We denote the objective function of Eq. (2) by $\mathcal{P}^t$ and refer to it as the *instantaneous* primal problem to be solved on trial t. The dual optimization problem of $\mathcal{P}^t$ is the maximization problem

$$\max_{\alpha_1^t, .., \alpha_{k_t}^t} \; \sum_{j=1}^{k_t} \alpha_j^t - \frac{1}{2} \left\| \omega^t + \sum_{j=1}^{k_t} \alpha_j^t y_j^t \mathbf{x}_j^t \right\|^2 \quad \text{s.t.} \quad \sum_{j=1}^{k_t} \alpha_j^t \leq C \;\; , \;\; \forall j : \alpha_j^t \geq 0 \quad . \tag{3}$$

The complete derivation is given in Appendix A.

Input:
Aggressiveness parameter $C > 0$
Initialize:
$\omega_1 = (0, \ldots, 0)$
For $t = 1, 2, \ldots, T$:
Receive instance matrix $X^t \in \mathbb{R}^{k_t \times n}$
Predict $\hat{\mathbf{y}}^t = \mathbf{X}^t \omega^t$
Receive correct labels $\mathbf{y}^t$
Suffer loss $\ell(\omega^t; (\mathbf{X}^t, \mathbf{y}^t))$
If $\ell > 0$:
Choose importance weights μ^t s.t.
$\mu^t_j \geq 0$ and $\sum_{j=1}^{k_t} \mu^t_j = 1$
Choose individual dual solutions α^t_j
Update $\omega^{t+1} = \omega^t + \sum_{j=1}^{k_t} \mu^t_j \alpha^t_j y^t_j \mathbf{x}^t_j$

Figure 2: Template of simultaneous projections algorithm.

Each dual variable corresponds to a single constraint of the primal problem. The minimizer of the primal problem is calculated from the optimal dual solution as follows, $\omega^{t+1} = \omega^t + \sum_{j=1}^{k_t} \alpha^t_j y^t_j \mathbf{x}^t_j$.

Unfortunately, in the common case, where each $\mathbf{x}^t_j$ is in an arbitrary orientation, there does not exist an analytic solution for the dual problem (Eq. 3). The difficulty stems from the fact that the sum of the weights α^t_j cannot exceed C. We tackle the problem by breaking it down into k_t reduced problems, each of which focuses on a single dual variable. By doing so we replace the global sum constraint, $\sum_{j=1}^{k_t} \alpha^t_j$, with multiple box constraints, $\alpha^t_j \leq C$, which can easily be dealt with. Formally, for the j'th variable, the j'th reduced problem solves Eq. (3) while fixing $\alpha^t_{j'} = 0$ for all $j' \neq j$. Each reduced optimization problem amounts to the following problem

$$\max_{\alpha^t_j} \ \alpha^t_j - \frac{1}{2} \left\| \omega^t + \alpha^t_j y^t_j \mathbf{x}^t_j \right\|^2 \quad \text{s.t.} \quad \alpha^t_j \in [0, C] \ . \tag{4}$$

As we demonstrate in the sequel, this reduction into independent problems serves two roles. First, it leads to simple solutions for the reduced problems. Second, and more important, the individual solutions can and are grouped into a feasible solution of the original problem for which we can prove various loss bounds.

We thus next obtain an exact or approximate solution for each reduced problem as if it were independent of the rest. We then choose a non-negative vector $\mu \in \Delta_{k_t}$ where Δ_{k_t} is the k_t dimension probability simplex, formally $\mu_i \geq 0$ and $\sum_{j=1}^{k_t} \mu_j = 1$. Given the vector μ, we multiply each α^t_j by a corresponding value μ^t_j. Our choice of μ assures us $\{\mu^t_j \alpha^t_j\}$ constitutes a feasible solution to the dual problem defined in Eq. (3) for the following reason. Each $\mu^t_j \alpha^t_j \geq 0$ and the fact that $\alpha^t_j \leq C$ implies that $\sum_{j=1}^{k_t} \mu^t_j \alpha^t_j \leq C$. Finally, the algorithm uses the combined solution and sets $\omega^{t+1} = \omega^t + \sum_{j=1}^{k_t} \mu^t_j \alpha^t_j y^t_j \mathbf{x}^t_j$. An illustration of the algorithm is provided in Fig. 4.

Variant	Choosing μ_j^t	Choosing α_j^t
SimPerc	$\begin{cases} \frac{1}{\lvert\mathcal{M}^t\rvert} & j \in \mathcal{M}^t \\ 0 & \text{otherwise} \end{cases}$	C
ConProj	$\begin{cases} \frac{1}{\lvert\mathcal{M}^t\rvert} & j \in \mathcal{M}^t \\ 0 & \text{otherwise} \end{cases}$	$\min\left\{C, \frac{\ell\left(\omega^t;(\mathbf{x}_j^t,y_j^t)\right)}{\left\|\mathbf{x}_j^t\right\|^2}\right\}$
ConProj	$\begin{cases} \frac{1}{\lvert\Gamma^t\rvert} & j \in \Gamma^t \\ 0 & \text{otherwise} \end{cases}$	$\min\left\{C, \frac{\ell\left(\omega^t;(\mathbf{x}_j^t,y_j^t)\right)}{\left\|\mathbf{x}_j^t\right\|^2}\right\}$
SimOpt	See Fig. 4	

Figure 3: Schemes for choosing μ and α.

5. Solving the Reduced Problems

We next present four schemes to obtain a solution for the reduced problem (Eq. 4) and then combine the solution into a single update. The first three schemes provide feasible solutions for the reduced problems and are easy to implement. However, these solutions are not necessarily optimal. In Sec. 5.4 we describe a rather involved yet efficient procedure for finding the optimal solution of each reduced problem along side with their weight vector μ which constitute the means for combining the individual solutions.

5.1 Simultaneous Perceptron

The simplest of the update forms generalizes the famous Perceptron algorithm from Rosenblatt (1958) by setting α_j^t to C if the j'th instance is incorrectly labeled, and to 0 otherwise. We then set the weight of μ_j^t to $\frac{1}{|\mathcal{M}^t|}$ for $j \in \mathcal{M}^t$ and to 0 otherwise. We abbreviate this scheme as the *SimPerc* algorithm.

5.2 Soft Simultaneous Projections

The soft simultaneous projections scheme uses the fact that each reduced problem has an analytic solution, which yields $\alpha_j^t = \min\left\{C, \ell\left(\omega^t;(\mathbf{x}_j^t,y_j^t)\right) / \left\|\mathbf{x}_j^t\right\|^2\right\}$. We independently assign each α_j^t this optimal solution. We next set μ_j^t to be $\frac{1}{|\Gamma^t|}$ for $j \in \Gamma^t$ and to 0 otherwise. We would like to comment that this solution may update α_j^t also for instances which were correctly classified as long as the margin they attain is not sufficiently large. We abbreviate this scheme as the *SimProj* algorithm.

5.3 Conservative Simultaneous Projections

Combining ideas from the above methods, the conservative simultaneous projections scheme optimally sets α_j^t according to the analytic solution. It differs from the SimProj algorithm in the way it selects μ^t. In the conservative scheme only the instances which were incorrectly predicted ($j \in \mathcal{M}^t$) are assigned a positive weight. Put another way, μ_j^t is set to $\frac{1}{|\mathcal{M}^t|}$ for $j \in \mathcal{M}^t$ and to 0 otherwise. We abbreviate this scheme as the *ConProj* algorithm.

5.4 Jointly Optimizing μ and α

Recall that our goal is to propose a feasible solution to Eq. (3) and we do so by independently considering the optimization problem of Eq. (4) for each j. Following, we multiply each α_j^t by a coefficient μ_j^t so that all $\mu_j^t \alpha_j^t$ form a feasible solution. The following analysis shows that the two steps can be unified. For brevity, we omit the superscript t. The task of jointly optimizing both μ and α casts a seemingly non-convex optimization and finding the optimal solution is a priori a hard problem. In this section we derive a somewhat counterintuitive result. By exploring the structure of the problem on hand we show that this joint optimization problem can efficiently be solved in $k_t \log k_t$ time. .

We begin by taking the derivative of the optimal values for α_j while assuming that the values μ_j are fixed and define a convex combination. The reduced problem of Eq. (4) becomes

$$\begin{aligned} \max_{\alpha_j} \ & \mu_j \alpha_j - \frac{1}{2} \left\| \omega + \mu_j \alpha_j y_j \mathbf{x}_j \right\|^2 \\ \text{s.t.} \quad & \alpha_j \in [0, C] \end{aligned} ,$$

which can be rewritten as

$$\max_{\alpha_j} \mu_j \alpha_j \left(1 - y_j \left(\omega \cdot \mathbf{x}_j\right)\right) - \frac{1}{2} \mu_j^2 \alpha_j^2 \left\|\mathbf{x}_j\right\|^2 - \frac{1}{2} \left\|\omega\right\|^2 \quad \text{s.t.} \quad \alpha_j \in [0, C] \ .$$

For brevity, we denote the squared norm of $\mathbf{x}_j$ by ν_j. Thus, omitting constants that do not depend on α_j and μ_j, the above optimization problem can be written as

$$\max_{\alpha_j} \ \mu_j \alpha_j \left(1 - y_j \left(\omega \cdot \mathbf{x}_j\right)\right) - \frac{1}{2} \mu_j^2 \alpha_j^2 \nu_j \quad \text{s.t.} \quad 0 \le \alpha_j \le C \ . \tag{5}$$

Let us denote the hinge-loss on instance j, $\max\{0, 1 - y_j \left(\omega \cdot \mathbf{x}_j\right)\}$ by ℓ_j. By taking the derivative of the Lagrangian with respect to α_j and equating the result with zero, we get that

$$\alpha_j = \min \left\{ C, \frac{l_j}{\mu_j \nu_j} \right\} \ .$$

We can now look at two disjoint cases. The first case is when $\alpha_j = \frac{l_j}{\mu_j \nu_j} < C$. In this case α_j takes the value of $\frac{l_j}{\mu_j \nu_j}$ and the value of the optimization problem above becomes

$$\frac{\ell_j^2}{\nu_j} - \frac{1}{2} \frac{\ell_j^2}{\nu_j} = \frac{1}{2} \frac{l_j^2}{\nu_j} \ .$$

We note in passing that this expression does not depend on μ_j.

The second case is when $\alpha_j = C$. Plugging α_j into Eq. (5) we get the following expression, as a function of μ, which we denote by $f_j(\mu_j)$,

$$f_j(\mu_j) = \mu_j C \ell_j - \frac{1}{2} \mu_j^2 C^2 \nu_j \ . \tag{6}$$

We next take the derivative of f_j above with respect to μ_j and obtain

$$\frac{\partial f}{\partial \mu_j} = C \ell_j - \mu_j C^2 \nu_j \ ,$$

from which we conclude that the optimal value for μ_j is

$$\frac{\ell_j}{C\nu_j} \ .$$

Plugging the optimal value for μ_j back in Eq. (6) we get that the maximum of $f_j(\mu_j)$ is

$$f_j\left(\frac{\ell_j}{C\nu_j}\right) = \frac{\ell_j^2}{\nu_j} - \frac{1}{2}\frac{\ell_j^2}{\nu_j} = \frac{1}{2}\frac{\ell_j^2}{\nu_j} \ .$$

To recap, we may express the optimal value of Eq. (5) as a function of μ_j as follows.

$$f_j(\mu_j) = \begin{cases} \mu_j C\ell_j - \frac{1}{2}\mu_j^2 C^2\nu_j & \mu_j \leq \frac{\ell_j}{C\nu_j} \\ \frac{1}{2}\frac{\ell_j^2}{\nu_j} & \text{otherwise} \end{cases} \ .$$

Thus, f_j is monotonically increasing in the range $0 \leq \mu_j \leq \frac{\ell_j}{C\nu_j}$ and is constant for values greater than $\frac{\ell_j}{C\nu_j}$.

Recall that our primary goal is to find a convex combination of μ_j. Thus, we would like to find the optimal assignment to μ given that α is set optimally. We end up with the following optimization problem.

$$\begin{aligned} &\max \sum_j f_j(\mu_j) \\ &\text{s.t.} \qquad \forall j : \mu_j \geq 0 \qquad \sum_j \mu_j = 1 \end{aligned} \ . \tag{7}$$

As previously discussed, for all j, f_j increases in the range $0 \leq \mu_j \leq \frac{\ell_j}{C\nu_j}$ and is constant afterwards. We may use this fact to further classify the structure of the optimal solution of Eq. (7). Assume first that $\sum_j \frac{\ell_j}{C\nu_j} \leq 1$. In this case we can increment each, $\mu_j = \frac{\ell_j}{C\nu_j} + B$, where B is a non-negative constant which assures that $\sum_j \mu_j = 1$. This assignment of μ is clearly optimal, as f_j is increasing and reaches its maximum obtainable value for $\mu_j \geq \frac{\ell_j}{C\nu_j}$.

Now, suppose $\sum_j \frac{\ell_j}{C\nu_j} > 1$. In such a case there exists an optimal solution with $\mu_j \leq \frac{\ell_j}{C\nu_j}$ for all j. Suppose in contrary that for every optimal solution μ there exists some $\hat{j}$ where $\mu_{\hat{j}} = \frac{\ell_{\hat{j}}}{C\nu_{\hat{j}}} + \varepsilon$ for some non-negative ε. Since $\sum_j \frac{\ell_j}{C\nu_j} > 1$ there exists some j' with $\mu_{j'} < \frac{\ell_{j'}}{C\nu_{j'}}$. Since $f_{j'}$ monotonely increases when $\mu_{j'} < \frac{\ell_{j'}}{C\nu_{j'}}$, while $f_{\hat{j}}$ is constant for $\mu_{\hat{j}} > \frac{\ell_{\hat{j}}}{C\nu_{\hat{j}}}$ we can create a new assignment $\mu^\star$ increasing the value of the sum $\sum_j f_j(\mu_j)$ by setting $\mu^\star_{\hat{j}} = \frac{\ell_{\hat{j}}}{C\nu_{\hat{j}}}$ and add ε to $\mu^\star_{j'}$. We thus conclude that for each solution where for some $\mu_{\hat{j}} > \frac{\ell_{\hat{j}}}{C\nu_{\hat{j}}}$ there exists an assignment $\mu^\star$ where for all j, $\mu_j \leq \frac{\ell_j}{C\nu_j}$ and the objective of Eq. (7) is at least as high.

Thus, when $\sum_j \frac{\ell_j}{C\nu_j} > 1$ we may add the constraint that $\mu_j \leq \frac{\ell_j}{C\nu_j}$ and obtain the following optimization problem.

$$\begin{aligned} &\max_\mu \sum_j f_j(\mu_j) \\ &\text{s.t.} \qquad \forall j : 0 \leq \mu_j \leq \frac{\ell_j}{C\nu_j} \qquad \sum_j \mu_j = 1 \end{aligned} \ .$$

We next introduce non-negative Lagrange multipliers τ, $\{\beta_j\}$, and $\{\eta_j\}$ to obtain the following Lagrangian,

$$L = \sum_j C\ell_j\mu_j - \frac{1}{2}\mu_j^2 C^2\nu_j - \sum_j \beta_j\mu_j - \tau\left(\sum_j \mu_j - 1\right) + \sum_j \eta_j\left(\mu_j - \frac{\ell_j}{C\nu_j}\right) \ .$$

Taking the derivative with respect to μ_j and comparing to 0 we get the following condition.

$$C\ell_j - \mu_j C^2\nu_j - \beta_j - \tau + \eta_j = 0 \ ,$$

or

$$\mu_j = \frac{C\ell_j - \beta_j - \tau + \eta_j}{C^2\nu_j} \ .$$

The complementary slackness assures us that when $\mu_j > 0$ then $\beta_j = 0$ and thus

$$\mu_j = \frac{C\ell_j - \tau + \eta_j}{C^2\nu_j} \ .$$

Similarly, $\eta_j > 0$ only when $\mu_j = \frac{\ell_j}{C\nu_j}$ and is used only when τ is negative, However, τ may be negative only when $\sum_j \frac{\ell_j}{C\nu_j} < 1$, which we covered before. To summarize, we can write the optimal solution as

$$\mu_j = \max\{0, \frac{C\ell_j - \tau}{C^2\nu_j}\} \ . \tag{8}$$

We now focus our attention on the task of finding the value of τ. First, note that every value of τ partitions the set $1,\ldots,k_t$ into two sets, indices j whose $\mu_j > 0$ and indices for which $\mu_j = 0$. Formally, let $H = \left\{j | C\ell_j > \tau\right\}$ and $L = [k_t] \setminus H$ denote the two sets partitioned according to τ. Clearly $j \in H \iff \mu_j > 0$. Clearly, knowing the value τ allows us to compute the partition to H and L. The converse, however, is also true. Had we known H and L it would have been straightforward to compute τ by using the fact that μ is in the probability simplex, $\sum_j \mu_j = 1$, to get that

$$\sum_{j\in H} \frac{C\ell_j - \tau}{C^2\nu_j} = 1 \ . \tag{9}$$

Eq. (9) allows us to easily compute τ and obtain

$$\tau = \frac{\sum_{j\in H} \frac{C\ell_j}{C^2\nu_j} - 1}{\sum_{j\in H} \frac{1}{C^2\nu_j}} \ . \tag{10}$$

In order to verify τ serve as a feasible solution, we're required to verify that $\sum_{j\in H} \mu_j = 1$ and that for all $j \in L:\ C\ell_j - \tau \leq 0$. The following lemma states that there is only a single feasible value for τ.

Lemma 1 *Let ℓ_j denote the hinge-loss of instance j, $\max\{0, 1 - y_j(\omega\cdot\mathbf{x}_j)\}$. Let ν_j denote the squared norm of $\mathbf{x}_j$. Denote by f_j the function of μ_j given by $f_j(\mu_j) = \mu_j C\ell_j - \frac{1}{2}\mu_j^2 C^2\nu_j$. Finally let μ denote an optimal solution to Eq. (7) computed according to Eq. (8) and $H(\tau)$ denote the set of indices j for which $C\ell_j > \tau$. Then, there is a single value τ that satisfies that $\sum_{j\in H\tau} \mu_j = 1$ while maintaining that $\forall\, j \notin H : \mu_j = 0$.*

Input:
$\ell_j, \nu_j \quad j \in [k_t]$
Algorithm:
Sort the indices $\{1,\ldots,k_t\}$ by decreasing order of ℓ_j
For $i = 2,\ldots,k_t$:
Define $H = \{1,\ldots,i-1\}$
Compute $\tau = \frac{\sum_{j\in H} \frac{C\ell_j}{C^2\nu_j} - 1}{\sum_{j\in H} \frac{1}{C^2\nu_j}}$
Validate $C\ell_i \leq \tau$. If not, continue to next iteration
Set $\mu_j = \frac{C\ell_j - \tau}{C^2\nu_j}$
Set $\alpha_j = \min\left\{C, \frac{l_j}{\mu_j \nu_j}\right\}$

Figure 4: Calculating μ and α efficiently.

Proof Suppose by contradiction that there are two feasible values for τ, and denote these values as τ_1 and τ_2. Denote $H(\tau_1)$ and $H(\tau_2)$ by H_1 and H_2 respectively. Assume without loss of generality that $\tau_1 < \tau_2$.

First we note that $H_2 \subset H_1$. However,

$$1 = \sum_{j\in H_1} \frac{C\ell_j - \tau_1}{C^2\nu_j} > \sum_{j\in H_1} \frac{C\ell_j - \tau_2}{C^2\nu_j} \quad ,$$

where the inequality is due to our assumption that $\tau_1 < \tau_2$. Since $\sum_{j\in H_2} \frac{C\ell_j - \tau_2}{C^2\nu_j}$ must equal 1, we conclude that H_2 must strictly contain more items than H_1. We have thus obtained a contradiction. ■

Using Lemma 1 we can devise an efficient algorithm for finding the optimal value for τ. We first sort the indices $1,\ldots,k_t$ by decreasing order of ℓ_j. Then, for every $i = 2,\ldots,k_t$, we define $H_i = \{1,\ldots,i-1\}$ and compute the value suitable value of τ according to Eq. (10). Finally we verify if $C\ell_i \leq \tau$. The algorithm for finding τ is formally given in Fig. 4.

To recap, we have suggested a mechanism for jointly optimizing both μ and α. We showed that it suffices to find a value τ which consistently divides the set $[k_t]$ into two sets as follows. The first set corresponds to indices j for which μ_j is zero and the second includes the non-zero components of μ. Furthermore, we showed that once τ is known, obtaining the vector μ is a simple task. Last, we described how Lemma 1 translates into an efficient algorithm for finding τ. We thus derived another simultaneous projections scheme, denoted by *SimOpt*, which jointly optimized α and μ. This variant of the simultaneous projections framework is guaranteed to yield the largest increase in the dual compared to all other simultaneous projections schemes. We describe empirical results which validate experimentally this property in Sec. 9.

6. Analysis

The algorithms described in the previous section perform updates in order to increase the instantaneous dual problem defined in Eq. (3). We now use the mistake bound model to derive an upper

bound on the number of trials on which the predictions of SimPerc and ConProj algorithms are imperfect. Following Shalev-Shwartz and Singer (2006a), the first step in the analysis is to tie the instantaneous dual problems to a global loss function. To do so, we introduce a primal optimization problem defined over the *entire* sequence of examples as follows,

$$\min_{\omega\in\mathbb{R}^n} \frac{1}{2}\|\omega\|^2 + C\sum_{t=1}^{T} \ell\left(\omega; \left(X^t, Y^t\right)\right) \quad .$$

We rewrite the optimization problem as the following equivalent constrained optimization problem,

$$\min_{\omega\in\mathbb{R}^n,\xi\in\mathbb{R}^T} \frac{1}{2}\|\omega\|^2 + C\sum_{t=1}^{T}\xi_t \quad \text{s.t.} \quad \forall t\in[T], \forall j\in[k_t] : y_j^t\left(\omega\cdot\mathbf{x}_j^t\right) \geq 1-\xi_t \quad \forall t : \xi_t \geq 0 \quad . \tag{11}$$

We denote the value of the objective function at (ω,ξ) for this optimization problem by $\mathcal{P}(\omega,\xi)$. A competitor who may see the entire sequence of examples in advance may in particular set (ω,ξ) to be the minimizer of the problem which we denote by $(\omega^\star,\xi^\star)$. Standard usage of Lagrange multipliers yields that the dual of Eq. (11) is,

$$\max_{\lambda} \sum_{t=1}^{T}\sum_{j=1}^{k_t}\lambda_{t,j} - \frac{1}{2}\left\|\sum_{t=0}^{T}\sum_{j=1}^{k_t}\lambda_{t,j}y_j^t\mathbf{x}_j^t\right\|^2 \quad \text{s.t.} \quad \forall t : \sum_{j=1}^{k_t}\lambda_{t,j} \leq C \quad \forall t,j : \lambda_{t,j}\geq 0 \quad . \tag{12}$$

We denote the value of the objective function of Eq. (12) by $\mathcal{D}(\lambda_1,\cdots,\lambda_T)$, where each λ_t is a vector in $\mathbb{R}^{k_t}$. Through our derivation we use the fact that any set of dual variables $\lambda_1,\cdots,\lambda_T$ defines a feasible solution $\omega = \sum_{t=1}^{T}\sum_{j=1}^{k_t}\lambda_{t,j}y_j^t\mathbf{x}_j^t$ with a corresponding assignment of the slack variables.

Clearly, the optimization problem given by Eq. (12) depends on all the examples from the first trial through time step T and thus can only be solved in hindsight. We note however, that if we ensure that $\lambda_{s,j}=0$ for all $s>t$ then the dual function no longer depends on instances occurring on rounds proceeding round t. As we show next, we use this primal-dual view to derive the skeleton algorithm presented in Fig. 2 by finding a new feasible solution for the dual problem on every trial. Formally, the instantaneous dual problem, given by Eq. (3), is equivalent (after omitting an additive constant) to the following constrained optimization problem,

$$\max_{\lambda} \mathcal{D}(\lambda_1,\cdots,\lambda_{t-1},\lambda,\mathbf{0},\cdots,\mathbf{0}) \quad \text{s.t.} \quad \lambda\geq\mathbf{0} \quad , \quad \sum_{j=1}^{k_t}\lambda_j \leq C \quad . \tag{13}$$

That is, the instantaneous dual problem is obtained from $\mathcal{D}(\lambda_1,\cdots,\lambda_T)$ by fixing $\lambda_1,\ldots,\lambda_{t-1}$ to the values set in previous rounds, forcing λ_{t+1} through λ_T to the zero vectors, and choosing a feasible vector for λ_t. Given the set of dual variables $\lambda_1,\ldots,\lambda_{t-1}$ it is straightforward to show that the prediction vector used on trial t is $\omega^t = \sum_{s=1}^{t-1}\sum_j \lambda_{s,j}y_j^s\mathbf{x}_j^s$. Equipped with these relations and omitting constants which do not depend on λ_t, Eq. (13) can be rewritten as,

$$\max_{\lambda_1,\ldots,\lambda_{k_t}} \sum_{j=1}^{k_t}\lambda_j - \frac{1}{2}\left\|\omega^t + \sum_{j=1}^{k_t}\lambda_j y_j^t\mathbf{x}_j^t\right\|^2 \quad \text{s.t.} \quad \forall j : \lambda_j\geq 0, \quad \sum_{j=1}^{k_t}\lambda_j \leq C \quad . \tag{14}$$

The problems defined by Eq. (14) and Eq. (3) are equivalent. Thus, weighing the variables $\alpha_1^t,\ldots,\alpha_{k_t}^t$ by $\mu_1^t,\ldots,\mu_{k_t}^t$ also yields a feasible solution for the problem defined in Eq. (13), namely $\lambda_{t,j} = \mu_j^t\alpha_j^t$. We now tie all of these observations together by using the weak-duality theorem. Our first bound is given for the SimPerc algorithm.

Theorem 2 *Let $\left(\mathbf{X}^1,\mathbf{y}^1\right),\ldots,\left(\mathbf{X}^T,\mathbf{y}^T\right)$ be a sequence of examples where $\mathbf{X}^t$ is a matrix of k_t examples and $\mathbf{y}^t$ are the associated labels. Assume that for all t and j the norm of an instance $\mathbf{x}_j^t$ is at most R. Then, for any $\omega^\star \in \mathbb{R}^n$ the number of trials on which the prediction of SimPerc is imperfect is at most,*

$$\frac{\frac{1}{2}\|\omega^\star\|^2 + C\sum_{t=1}^T \ell\left(\omega^\star;(\mathbf{X}^t,\mathbf{y}^t)\right)}{C-\frac{1}{2}C^2R^2} \quad .$$

Proof To prove the theorem we make use of the weak-duality theorem. Recall that any dual feasible solution induces a value for the dual's objective function which is upper bounded by the optimum value of the primal problem, $\mathcal{P}\left(\omega^\star,\xi^\star\right)$. In particular, the solution obtained at the end of trial T is dual feasible, and thus $\mathcal{D}(\lambda_1,\ldots,\lambda_T) \leq \mathcal{P}(\omega^\star,\xi^\star)$. We now rewrite the left-hand side of the above equation as the following sum,

$$\mathcal{D}(\mathbf{0},\ldots,\mathbf{0}) + \sum_{t=1}^T \left[\mathcal{D}(\lambda_1,\ldots,\lambda_t,\mathbf{0},\ldots,\mathbf{0}) - \mathcal{D}(\lambda_1,\ldots,\lambda_{t-1},\mathbf{0},\ldots,\mathbf{0})\right] \quad .$$

Note that $\mathcal{D}(\mathbf{0},\ldots,\mathbf{0})$ equals 0. Therefore, denoting by Δ_t the difference in two consecutive dual objective values, $\mathcal{D}(\lambda_1,\ldots,\lambda_t,\mathbf{0},\ldots,\mathbf{0}) - \mathcal{D}(\lambda_1,\ldots,\lambda_{t-1},\mathbf{0},\ldots,\mathbf{0})$, we get that $\sum_{t=1}^T \Delta_t \leq \mathcal{P}(\omega^\star,\xi^\star)$. We now turn to bounding Δ_t from below. First, note that if the prediction on trial t is perfect ($\mathcal{M}^t = \emptyset$) then SimPerc sets λ_t to the zero vector and thus $\Delta_t = 0$. We can thus focus on trials for which the algorithm's prediction is imperfect. We remind the reader that by unraveling the update of ω^t we get that $\omega^t = \sum_{s<t}\sum_{j=1}^{k_s} \lambda_{s,j} y_j^s \mathbf{x}_j^s$. We now rewrite Δ_t as follows,

$$\Delta_t = \sum_{j=1}^{k_t} \lambda_{t,j} - \frac{1}{2}\left\|\omega^t + \sum_{j=1}^{k_t} \lambda_{t,j} y_j^t \mathbf{x}_j^t\right\|^2 + \frac{1}{2}\left\|\omega^t\right\|^2 \quad . \tag{15}$$

By construction, $\lambda_{t,j} = \mu_j^t \alpha_j^t$ and $\sum_{j=1}^{k_t} \mu_j^t = 1$, which lets us further expand Eq. (15) and write,

$$\Delta_t = \sum_{j=1}^{k_t} \mu_j^t \alpha_j^t - \frac{1}{2}\left\|\omega^t + \sum_{j=1}^{k_t} \mu_j^t \alpha_j^t y_j^t \mathbf{x}_j^t\right\|^2 + \frac{1}{2}\sum_{j=1}^{k_t} \mu_j^t \left\|\omega^t\right\|^2 \quad .$$

The squared norm, $\|\cdot\|^2$ is a convex function in its vector argument and thus Δ_t is concave, which yields the following lower bound on Δ_t,

$$\Delta_t \geq \sum_{j=1}^{k_t} \mu_j^t \left[\alpha_j^t - \frac{1}{2}\left\|\omega^t + \alpha_j^t y_j^t \mathbf{x}_j^t\right\|^2 + \frac{1}{2}\left\|\omega^t\right\|^2\right] \quad . \tag{16}$$

The SimPerc algorithm sets μ_j^t to be $1/|\mathcal{M}^t|$ for all $j \in \mathcal{M}^t$ and to be 0 otherwise. Furthermore, for all $j \in \mathcal{M}^t$, α_j^t is set to C. Thus, the right-hand side of Eq. (16) can be further simplified and written as,

$$\Delta_t \geq \sum_{j \in \mathcal{M}^t} \mu_j^t \left[C - \frac{1}{2}\left\|\omega^t + C y_j^t \mathbf{x}_j^t\right\|^2 + \frac{1}{2}\left\|\omega^t\right\|^2\right] \quad .$$

In order to further explore Eq. (6) we require the following simple lemma

Lemma 3 *Let ω^t denote the predictor used by the SimPerc scheme on trial t. Let $j \in \mathcal{M}^t$ denote an index of a mispredicted instance on trial t. Then*

$$\frac{1}{2}\left\|\omega^t + Cy_j^t \mathbf{x}_j^t\right\|^2 - \frac{1}{2}\left\|\omega^t\right\|^2 \leq \frac{1}{2}C^2\left\|y_j^t \mathbf{x}_j^t\right\|^2 \quad .$$

Proof We start by expanding the norm of the vector after the update,

$$\frac{1}{2}\left\|\omega^t + Cy_j^t \mathbf{x}_j^t\right\|^2 = \frac{1}{2}\left\|\omega^t\right\|^2 + Cy_j^t \omega^t \cdot \mathbf{x}_j^t + \frac{1}{2}C^2\left\|y_j^t \mathbf{x}_j^t\right\|^2 \quad .$$

Thus, the change in the norm is,

$$\begin{aligned}\frac{1}{2}\left\|\omega^t + Cy_j^t \mathbf{x}_j^t\right\|^2 - \frac{1}{2}\left\|\omega^t\right\|^2 &= \frac{1}{2}\left\|\omega^t\right\|^2 + Cy_j^t \omega^t \cdot \mathbf{x}_j^t + \frac{1}{2}C^2\left\|y_j^t \mathbf{x}_j^t\right\|^2 - \frac{1}{2}\left\|\omega^t\right\|^2 \\ &= Cy_j^t \omega^t \cdot \mathbf{x}_j^t + \frac{1}{2}C^2\left\|y_j^t \mathbf{x}_j^t\right\|^2 \quad .\end{aligned}$$

The set $\mathcal{M}^t$ consists of indices of instances which were incorrectly classified. Thus, $y_j^t(\omega^t \cdot \mathbf{x}_j^t) \leq 0$ for every $j \in \mathcal{M}^t$. The equation above can thus be further bounded by $\frac{1}{2}C^2\left\|y_j^t \mathbf{x}_j^t\right\|^2$. ■

Lemma 3 assures us that for all instances whose $\mu_j > 0$ the term $\frac{1}{2}\left\|\omega^t + Cy_j^t \mathbf{x}_j^t\right\|^2 - \frac{1}{2}\left\|\omega^t\right\|^2$ can be upper bounded. Therefore, Δ_t can further be bounded from below as follows,

$$\Delta_t \geq \sum_{j \in \mathcal{M}^t} \mu_j^t \left[C - \frac{1}{2}C^2\left\|y_j^t \mathbf{x}_j^t\right\|^2\right] \geq \sum_{j \in \mathcal{M}^t} \mu_j^t \left[C - \frac{1}{2}C^2R^2\right] = C - \frac{1}{2}C^2R^2 \quad ,$$

where for the second inequality we used the fact that the norm of all the instances is bounded by R. To recap, we have shown that on trials for which the prediction is imperfect $\Delta_t \geq C - \frac{1}{2}C^2R^2$, while in perfect trials where no mistake is made $\Delta_t = 0$. Putting all the inequalities together we obtain the following bound,

$$\left(C - \frac{1}{2}C^2R^2\right)\varepsilon \leq \sum_{t=1}^{T} \Delta_t = \mathcal{D}(\lambda_1, \ldots, \lambda_T) \leq \mathcal{P}(\omega^\star, \xi^\star) \quad ,$$

where ε is the number of imperfect trials. Finally, rewriting $\mathcal{P}(\omega^\star, \xi^\star)$ as

$$\frac{1}{2}\|\omega^\star\|^2 + C\sum_{t=1}^{T} \ell(\omega^\star; (\mathbf{X}^t, \mathbf{y}^t) \quad ,$$

yields the bound stated in the theorem. ■

The ConProj algorithm updates the same set of dual variables as the SimPerc algorithm. However, it selects α_j^t to be the optimal solution of Eq. (4). Thus, the value of Δ_t attained by the ConProj algorithm is never lower than the value attained by the SimPerc algorithm, assuming both versions start with the same predictor ω_t. The case of the SimOpt algorithm is very similar, as it promises to optimally increase the value of Δ_t and thus is never lower than the value attained by the SimPerc algorithm. The following corollary is a direct consequence of these observations.

Corollary 4 *Under the same conditions of Thm. 2 and for any $\omega^\star \in \mathbb{R}^n$, the number of trials on which the prediction of either ConProj or SimOpt is imperfect is at most,*

$$\frac{\frac{1}{2}\|\omega^\star\|^2 + C\sum_{t=1}^{T}\ell\left(\omega^\star;(\mathbf{X}^t,\mathbf{y}^t)\right)}{C-\frac{1}{2}C^2R^2} \quad .$$

Note that the predictions of the SimPerc algorithm do not depend on the specific value of C, thus for $R=1$ and an optimal choice of C the bound attained in Thm. 2 now becomes.

$$\varepsilon \le \ell\left(\omega^\star;(\mathbf{X}^t,\mathbf{y}^t)\right) + \frac{1}{2}\|\omega^\star\|^2 + \frac{1}{2}\sqrt{\|\omega^\star\|^4 + \|\omega^\star\|^2\sum_{t=1}^{T}\ell\left(\omega^\star;(\mathbf{X}^t,\mathbf{y}^t)\right)} \quad .$$

See Appendix B for a complete analysis.

We conclude this section with a few closing words about the SimProj variant. The SimPerc and ConProj algorithms ensure a minimal increase in the dual by focusing solely on classification errors and ignoring margin errors. While this approach ensures a sufficient increase of the dual, in practice it appears to be a double edged sword as the SimProj algorithm performs empirically better. This superior empirical performance can be motivated by a viewing the similarity of the update forms performed by the SimProj and SimOpt variants, which means that the actual increase in the dual attained by the SimProj algorithm is larger than can be guaranteed using worst case analysis.

7. Decomposable Losses

Recall that our algorithms tackle complex decision problems by decomposing each instance into multiple binary decision tasks, thus, on trial t the algorithm receives k_t instances. The classification scheme is evaluated by looking at the maximal violation of the margin constraints $\ell(\hat{\mathbf{y}}^t,\mathbf{y}^t) = \max_j \left[1-y_j^t\hat{y}_j^t\right]_+$. While such approach often captures the inherent relation between the multiple binary tasks, it may often be desired to introduce more complex evaluation measures. In this section we introduce a generalization of the algorithm for various decomposable losses. As a corollary we obtain a *Simultaneous Projection* algorithm that is competitive with the average performance error on each set of k_t instances.

First, we introduce the notion of the decomposable losses. On trial t the algorithms receives a partition of the k_t instances into r_t sets. Let $S_1^t,\ldots,S_{r_t}^t$ denote a partition of $[k_t]$ into r_t sets, namely, $\cup_l S_l^t = [k_t]$ and $\forall l \neq k:\ S_l^t \cap S_k^t = \emptyset$. A set S_l ties the instances in the sense that failing to predict *any* instance in S_l amounts to the same error as failing to predict all of them. We thus suffer a unit loss at trial t for each set S_l that was imperfectly predicted. The definition of the loss is extended to

$$\hat{\ell}\left(\hat{\mathbf{y}}^t,\mathbf{y}^t\right) = \frac{1}{r_t}\sum_{l=1}^{r_t}\max_{j\in S_l^t}\left[1-y_j^t\hat{y}_j^t\right]_+ \quad . \tag{17}$$

By construction, the setting suggested in Sec. 2 is a special case of the setting we consider in this section. We show in the sequel though that our original analysis carries over this this more general setting.

Thus, each iteration the algorithm receives k_t instances and a partition of the labels into sets $S_1^t,\ldots,S_{r_t}^t$. The instantaneous primal (Eq. 2) is thus extended to include a single slack variable for

Input:
Aggressiveness parameter $C > 0$
Initialize:
$\omega_1 = (0, \ldots, 0)$
For $t = 1, 2, \ldots, T$:
Receive instance matrix $X^t \in \mathbb{R}^{k_t \times n}$
Predict $\hat{\mathbf{y}}^t = \mathbf{X}^t \omega^t$
Receive correct labels $\mathbf{y}^t$
Receive partition of labels $S_1^t, \ldots, S_{r_t}^t$
Suffer loss $\hat{\ell}(\omega^t; (\mathbf{X}^t, \mathbf{y}^t))$
If $\hat{\ell} > 0$:
Choose importance weights μ^t s.t. for each set S_l^t, $\sum_{j \in S_l^t} \mu_j^t = 1$
Choose individual dual solutions α_j^t
Update $\omega^{t+1} = \omega^t + \sum_{l=1}^{r_t} \sum_{j \in S_l^t} \mu_j^t \alpha_j^t y_j^t \mathbf{x}_j^t$

Figure 5: The extended simultaneous projections algorithm for decomposable losses.

each set as follows:

$$\begin{aligned} &\min_{\omega \in \mathbb{R}^n, \xi \geq 0} \frac{1}{2} \left\| \omega - \omega^t \right\|^2 + \frac{C}{r_t} \sum_{l=1}^{r_t} \xi_l \\ &\text{s.t.} \qquad \forall l \in [r_t],\ \forall j \in S_l^t :\ \ y_j^t \left(\omega \cdot \mathbf{x}_j^t \right) \geq 1 - \xi_l \qquad \forall l \in [r_t] :\ \ \xi_l \geq 0 \end{aligned} \quad . \tag{18}$$

The dual of Eq. (18) is thus

$$\begin{aligned} &\max_{\alpha_1^t, \ldots, \alpha_{k_t}^t} \ \sum_{j=1}^{k_t} \alpha_j^t - \frac{1}{2} \left\| \omega^t + \sum_{j=1}^{k_t} \alpha_j^t y_j^t \mathbf{x}_j^t \right\|^2 \\ &\text{s.t.} \qquad \forall l : \sum_{j \in S_l^t} \alpha_j^t \leq \frac{C}{r_t} \qquad \forall j : \alpha_j^t \geq 0 \end{aligned} \quad .$$

Note that since $\forall k \neq l : S_l^t \cap S_k^t = \emptyset$ then the induced constraint $\sum_{j \in S_l^t} \alpha_j^t \leq \frac{C}{r_t}$ corresponds to a unique set of dual variables α_j^t. We can thus apply the same technique and select a non-negative vector μ where the entries corresponding to each set S_l^t form a probability distribution, namely $\forall l : \sum_{j \in S_l^t} \mu_j^t = 1$. To recap, we can employ any of the variants on each set *separately* and attain a dual feasible solution. We denote these variants as the decomposition variants. In Fig. 5 we provide the pseudo-code of the algorithm.

We next show that our mistake bound analysis can be extended to the decomposable loss. The analysis follows closely to the analysis presented in Sec. 6, where the global primal and global dual are modified so as to use the decomposition loss. We therefore focus only on highlighting the necessary changes. Eq. (11) thus becomes

$$\begin{aligned} &\min_{\omega \in \mathbb{R}^n, \xi \in \mathbb{R}^T} \frac{1}{2} \left\| \omega \right\|^2 + C \sum_{t=1}^{T} \sum_{l=1}^{r_t} \frac{\xi_{t,l}}{r_t} \\ &\text{s.t.} \qquad \forall t \in [T], \forall l \in [r_t], \forall j \in S_l^t : y_j^t \left(\omega \cdot \mathbf{x}_j^t \right) \geq 1 - \xi_{t,l} \qquad \forall t \forall l : \xi_{t,l} \geq 0 \end{aligned} \quad . \tag{19}$$

and its dual is

$$\max_{\lambda} \sum_{t=1}^{T}\sum_{j=1}^{k_t}\lambda_{t,j} - \frac{1}{2}\left\|\sum_{t=0}^{T}\sum_{j=1}^{k_t}\lambda_{t,j}y_j^t\mathbf{x}_j^t\right\|^2 \quad \text{s.t.} \quad \forall t\forall l\in[r_t]:\sum_{j\in S_l^t}\lambda_{t,j}\le\frac{C}{r_t} \quad \forall t,j:\lambda_{t,j}\ge 0 \ .$$

Clearly, the instantaneous dual can still be seen as optimizing the global dual, while fixing the dual variables $\lambda_{t',j}$ for all $t'\neq t$.

To recap, we replace the loss of the instantaneous optimization problem defined in Eq. (1) with the average over a decomposition of losses $\hat{\ell}$ as defined by Eq. (17). Next, in order to obtain a mistake bound, we look at the global optimization task defined by Eq. (19). As previously showed, the simultaneous projection scheme can be viewed as an incremental update to the dual of Eq. (19). It is interesting to note that for every decomposition of the k_t instances into sets, the value of $\hat{\ell}(\omega;(\mathbf{X}^t,\mathbf{y}^t))$ is upper bounded by $\ell(\omega;(\mathbf{X}^t,\mathbf{y}^t))$, as $\hat{\ell}$ is the average over the margin violations while ℓ corresponds to the *worst* margin violation. Thus, the loss underpinning the global optimization from Eq. (11) upper bounds the loss yielding Eq. (19). The following corollary immediately holds.

Corollary 5 *Under the same conditions of Thm. 2, the loss suffered along the run of either decomposition variant is at most,*

$$\frac{\frac{1}{2}\|\omega^\star\|^2 + C\sum_{t=1}^{T}\hat{\ell}(\omega^\star;(\mathbf{X}^t,\mathbf{y}^t))}{C-\frac{1}{2}C^2R^2} \ .$$

In conclusion, the simultaneous projection scheme allows us to easily obtain online algorithms and update schemes for complex problems, such algorithms are obtained by decomposing a complex problem into multiple binary problems. It is often the case where the maximal violation over all binary problems correctly captures the inherent violation of the original complex problem. In this section we explored cases where a more refined definition of error is required. Specifically, if we define each binary instance in a separate set, we obtain an algorithmic framework where our competitor is evaluated according to the average loss.

8. Simultaneous Multiplicative Updates

In this section we describe and analyze a multiplicative version of the simultaneous projection scheme. Recall that our motivation was to introduce a solution to the instantaneous optimization problem given in Eq. (2). The instantaneous objective captures the following trade-off. On one hand we would like to set ω to be as *close* as possible to ω^t. On the other hand, we would like to minimize the loss incurred by the instances received on trial t. In previous sections we used the squared Euclidean norm to define the measure of distance between ω^t and ω. In this section we take a different approach and use the relative entropy as the notion of closeness between two vectors. By doing so we derive a multiplicative version of our online algorithmic framework. In this section we confine ourselves to linear predictors that lie in the probability simplex, that is, we consider non-negative vectors ω such that $\sum_{i=1}^{n}\omega_i = 1$. Previously, we used a fixed value of 1 for the margin that is needed in order to suffer no loss, where it was understood that we may simultaneously scale the prediction vector and the margin. Since we now prevent such scaling due to the choice of the simplex domain, we need to slightly modify the definition of the loss and introduce the following definition, $\ell_\gamma(\hat{\mathbf{y}}^t,\mathbf{y}^t) = \max_j\left[\gamma - y_j^t\hat{y}_j^t\right]_+$.

Recall that on trial t the algorithm receives k_t instances arranged in a matrix $\mathbf{X}^t$. After extending a prediction vector, $\omega^t \mathbf{X}^t$, the algorithm receives the vector of correct labels $\mathbf{y}^t$ and suffers a loss for any incorrect prediction. If no mistake is made the algorithm proceeds to the next round. Otherwise we would like to set ω^t to be the solution of the following optimization problem

$$\min_{\omega \in \Delta_n} D_{\mathrm{KL}}(\omega \| \omega^t) + C\,\ell_\gamma(\omega; (\mathbf{X}^t, \mathbf{y}^t)) \quad , \tag{20}$$

where C is a trade-off parameter. The term D_{KL} is the relative entropy operator, also known as the Kullback-Leibler divergence, and is defined as

$$D_{\mathrm{KL}}(\omega \| \omega^t) = \sum_{i=1}^{n} \omega_i \log \frac{\omega_i}{\omega_i^t} \quad .$$

The dual problem of Eq. (20) is,

$$\begin{aligned} & \gamma \sum_{j=1}^{k_t} \alpha_j^t - \log\left(\sum_{i=1}^{n} \omega_i^t \exp\left(\sum_{j=1}^{k_t} \tau_i^j \right) \right) \\ & \text{s.t.} \qquad \sum_{j=1}^{k_t} \alpha_j^t \le C \qquad \forall j : \alpha_j^t \ge 0 \qquad \forall j : \tau^j = \alpha_j^t y_j^t \mathbf{x}_j^t \end{aligned} \quad . \tag{21}$$

The prediction vector ω is set as follows,

$$\omega_i = \omega_i^t \frac{\exp\left(\sum_{j=1}^{k_t} \tau_i^j \right)}{\sum_{l=1}^{n} \omega_l^t \exp\left(\sum_{j=1}^{k_t} \tau_i^j \right)} \quad . \tag{22}$$

The complete derivation of the dual problem and the update of ω is given in Appendix A.

We follow the same technique suggested in Sec. 4 and decompose Eq. (21) into k_t separate problems, each concerning a single dual variable. The resulting j'th reduced dual problem is thus

$$\begin{aligned} & \gamma \alpha_j^t - \log\left(\sum_{i=1}^{n} \omega_i^t \exp\left(\tau_i^j \right) \right) \\ & \text{s.t.} \qquad 0 \le \alpha_j^t \le C \qquad \tau^j = \alpha_j^t y_j^t \mathbf{x}_j^t \end{aligned} \quad . \tag{23}$$

We next obtain an exact or approximate solution for each reduced problem as if it were independent of the rest. We follow by choosing a vector $\mu \in \Delta_{k_t}$, and multiply each α_j^t by a corresponding value μ_j^t. Our choice of μ assures us $\{\mu_j^t \alpha_j^t\}$ constitutes a feasible solution to the dual problem defined in Eq. (21) for the following reason. Each $\mu_j^t \alpha_j^t \ge 0$ and the fact that $\alpha_j^t \le C$ implies that $\sum_{j=1}^{k_t} \mu_j^t \alpha_j^t \le C$. Finally, the algorithm uses the combined solution and sets ω^{t+1} according to Eq. (22). The template of the multiplicative simultaneous projections algorithm is described in Fig. 6.

We may now apply the methods introduced in Sec. 5 and introduce the multiplicative schemes. The SimPerc scheme can be trivially applied to the multiplicative setting. We next show a closed-form solution to α_j^t for each reduced problem if the components of each instance are from $\{-1, 0, 1\}^n$. To so we need to introduce the following notation.

$$W_j^+ = \frac{\sum_{i: y_j^t x_{ji}^t = 1} \omega_i^t}{\sum_{l=1}^{n} \omega_l^t} \;,\; W_j^- = \frac{\sum_{i: y_j^t x_{ji}^t = -1} \omega_i^t}{\sum_{l=1}^{n} \omega_l^t} \;,\; \text{and } W_j^0 = 1 - W_j^+ - W_j^- \quad .$$

Input:
Aggressiveness parameter $C > 0$
Initialize:
$\omega_1 = (\frac{1}{n}, \ldots, \frac{1}{n})$
For $t = 1, 2, \ldots, T$:
Receive instance matrix $X^t \in \mathbb{R}^{k_t \times n}$
Predict $\hat{\mathbf{y}}^t = \mathbf{X}^t \omega^t$
Receive correct labels $\mathbf{y}^t$
Suffer loss $\ell(\omega^t; (\mathbf{X}^t, \mathbf{y}^t))$
If $\ell > 0$:
Choose importance weights μ^t s.t. $\sum_{j=1}^{k_t} \mu_j^t = 1$
Choose individual dual solutions α_j^t
Compute $\tau^j = \alpha_j^t y_j^t \mathbf{x}_j^t$
Update $\omega_i^{t+1} = \frac{\omega_i^t \exp\left(\sum_{j=1}^{k_t} \mu_j^t \tau_i^j\right)}{\sum_l \omega_l^t \exp\left(\sum_{j=1}^{k_t} \mu_j^t \tau_l^j\right)}$

Figure 6: The multiplicative simultaneous projections algorithm.

The optimal value of α_j^t is thus log of the root of a quadric equation with W_j^+, W_j^-, W_j^0 as coefficients. We also need to take into account the boundary constraints on α_j^t, namely, $0 \leq \alpha_j^t \leq C$. Thus, α_j^t is the minimum between the following root and C,

$$\log\left(\frac{\gamma W_j^0 + \sqrt{\gamma^2 (W_j^0)^2 + 4(1-\gamma^2) W_j^+ W_j^-}}{2(1-\gamma) W_j^+}\right) ,$$

The derivation can be found at Appendix C. Using the closed-form solution for α_j^t we can adapt both the ConProj and SimProj scheme to the multiplicative setting.

We next turn our attention to the analysis of the multiplicative algorithm and focus on the SimPerc scheme. The analysis here follows closely the analysis presented in Sec. 6. Hence, the remainder of this section focuses on highlighting the key changes that are required. Formally, we prove the following theorem.

Theorem 6 *Let $\left(\mathbf{X}^1, \mathbf{y}^1\right), \ldots, \left(\mathbf{X}^T, \mathbf{y}^T\right)$ be a sequence of examples where $\mathbf{X}^t$ is a matrix of k_t examples and $\mathbf{y}^t$ are the associated labels. Assume that for all t and j the ℓ_∞ norm of an instance $\mathbf{x}_j^t$ is at most R. Then, for any $\omega^\star \in \Delta_n$ the number of trials on which the prediction of SimPerc is imperfect is at most,*

$$\frac{\sum_{i=1}^n \omega_i^\star \log \frac{\omega_i^\star}{1/n} + C \sum_{t=1}^T \ell_\gamma(\omega^\star; (\mathbf{X}^t, \mathbf{y}^t))}{C - \frac{1}{2} C^2 R^2} .$$

Proof Following the technique introduced in Sec. 6, our goal is to upper bound the number of imperfect trials compared to the performance of any fixed competitor, even one defined in hindsight.

Our competitor is thus evaluated using the global optimization problem given by,

$$\begin{aligned} &\min_{\omega\in\Delta_n,\xi\geq 0} \sum_{i=1}^{n} \omega_i \log\frac{\omega_i}{\omega_i^t} + C\sum_{t=1}^{T}\xi_t \\ &\text{s.t.} \qquad \forall t\in[T], \forall j\in[k_t] : y_j^t\left(\omega\cdot\mathbf{x}_j^t\right) \geq \gamma - \xi_t \qquad \forall t : \xi_t \geq 0 \end{aligned} \quad . \tag{24}$$

The dual of Eq. (24) is

$$\begin{aligned} &\gamma\sum_{t=1}^{T}\sum_{j=1}^{k_t}\lambda_j^t - \log\left(\sum_{i=1}^{n}\exp\left(\sum_{t=1}^{T}\sum_{j=1}^{k_t}\tau_i^{tj}\right)\right) \\ &\text{s.t.} \qquad \forall t\in[T] : \sum_{j=1}^{k_t}\lambda_j^t \leq C \qquad \forall t, \forall : \lambda_j^t \geq 0 \qquad \forall t, \forall j : \tau^{tj} = \lambda_j^t y_j^t \mathbf{x}_j^t \end{aligned} \quad . \tag{25}$$

We denote the objective of Eq. (25) by $\mathcal{D}(\lambda_1,\ldots,\lambda_T)$. The instantaneous dual of Eq. (21) can be seen as incrementally building an assignment for the dual: At trial t we fix λ^s for $s<t$ to their previous values, and fix λ^s for $s>t$ to 0. Thus ω^t is defined as follows

$$\omega_i^t = \frac{\exp\left(\sum_{s=1}^{t}\sum_{j=1}^{k_s}\tau_i^{sj}\right)}{\sum_{l=1}^{n}\exp\left(\sum_{s=1}^{T}\sum_{j=1}^{k_s}\tau_l^{sj}\right)} \quad .$$

The key difference between the multiplicative schemes and the previously analyzed scheme lies in Lemma 3. We thus progress to derive a similar lemma for the multiplicative setting.

Lemma 7 *Let $\theta = \sum_{l=1}^{t-1}\sum_{j=1}^{k_l}\lambda_j^t y_j^t \mathbf{x}_j^t$ denote the dual variables assigned in trials prior to t by the SimPerc scheme. Let $j\in\mathcal{M}^t$ denote an index of a mispredicted instance on trial t. Then, the difference,*

$$\log\left(\sum_{i=1}^{n}\exp\left(\theta_i + Cx_{ji}^t y_j^t\right)\right) - \log\left(\sum_{i=1}^{n}\exp\left(\theta_i\right)\right) \quad ,$$

is upper bounded by $\frac{1}{2}C^2\left\|\mathbf{x}\right\|_\infty^2$.

Proof Denote the vector $C\mathbf{x}_j^t y_j^t$ by τ. Let $F(\theta)$ denote the value of $\log\left(\sum_{i=1}^{n}e^{\theta_i}\right)$. Hence, we would like to upper bound the difference $F(\theta+\tau)-F(\theta)$. We prove the lemma based on the following inequality

$$F(\theta+\tau) - F(\theta) \;\leq\; \sum_{i=1}^{n}\frac{e^{\theta_i}}{\sum_l e^{\theta_l}}\tau_i + \frac{1}{2}\max_i \tau_i^2 \quad .$$

The above inequality was used and proved in numerous previous analyses of multiplicative update methods. See for instance Examples 2 and 5 in Kivinen and Warmuth (2001). Consider the term $\sum_{i=1}^{n}\frac{e^{\theta_i}}{\sum_l e^{\theta_l}}\tau_i$. Recall that the prediction in trial t is made by using the predictor defined by Eq. (22). Thus, the above term is the following inner product between the vector τ and the predictor used on round t,

$$\sum_{i=1}^{n}\frac{e^{\theta_i}}{\sum_l e^{\theta_l}}\tau_i = \sum_{i=1}^{n}\omega_i^t\tau_i = \langle\omega^t,\tau\rangle = Cy_j^t\langle\omega^t,\mathbf{x}_j^t\rangle \leq 0 \quad ,$$

where the last inequality is due to the fact that we assume $j \in \mathcal{M}^t$ (the prediction was incorrect) and the inner-product is non-positive. Therefore, we obtain the required upper bound

$$F(\theta+\tau) - F(\theta) \leq \frac{1}{2}\max_i \tau_i^2 \frac{1}{2}\|\tau\|_\infty^2 = \frac{1}{2}C^2 \left\|\mathbf{x}_j^t\right\|_\infty^2 \quad .$$

■

To recap, we showed that the instantaneous dual can be seen as incrementally constructing an assignment for a global dual function (Eq. 25). Furthermore, we showed that Lemma 3 can be adapted to the multiplicative settings. The rest of the proof follows the same lines of the proof given in Sec. 6. Namely, trials in which a prediction mistake was made, the SimPerc scheme is guaranteed a substantial increase in the incremental dual buildup. Finally, using weak-duality we obtain that the evaluation measure for the competitor is the lower bounded by,

$$\frac{\sum_{i=1}^n \omega_i^\star \log \frac{\omega_i^\star}{1/n} + C\sum_{t=1}^T \ell_\gamma(\omega^\star;(\mathbf{X}^t,\mathbf{y}^t))}{C - \frac{1}{2}C^2R^2} \quad .$$

■

The multiplicative ConProj scheme assigns α_j^t the value which maximizes the reduced instantaneous dual. The ConProj scheme thus maximizes the difference between the value of the global dual in two consecutive rounds. We thus obtain an equivalent corollary of Corollary 4 for the multiplicative setting.

Corollary 8 *Under the same conditions of Thm. 6 and for any $\omega^\star \in \mathbb{R}^n$, the number of trials on which the prediction of the ConProj scheme is imperfect is at most,*

$$\frac{\sum_{i=1}^n \omega_i^\star \log \frac{\omega_i^\star}{1/n} + C\sum_{t=1}^T \ell_\gamma(\omega^\star;(\mathbf{X}^t,\mathbf{y}^t))}{C - \frac{1}{2}C^2R^2} \quad .$$

We thus showed that the multiplicative SimPerc and ConProj schemes entertain a similar mistake bound as the original formulation. Note, however, that in the multiplicative settings we assume that the ℓ_∞ norm of all instances are bounded by R, while in the additive settings, we have assumed that the ℓ_2 norm of the instances is bounded by R.

9. Experiments

In this section we describe experiments we performed with synthetic and real data sets. The goal of the experiments is to underscore the properties of the simultaneous projection algorithms and to demonstrate some of their merits. Specifically, we examine how the various simultaneous projections variants perform with respect to the size of each block, how does the performance deteriorate with label noise, and the dependency of the algorithms on the number of relevant features. Our experiments with synthetic data are followed with email categorization experiments. On the synthetic data we compare our simultaneous projections algorithms with a commonly used technique whose updates are based on the most violating constraint on each online round (see for instance Crammer

et al., 2006). In the multiclass email categorization experiment, we also use the Sopopo algorithm described in Shalev-Shwartz and Singer (2006b) and a numerical solver which finds the optimal solution of the optimization problem on hand. To recap, we experimented with the following three families of online algorithms.

Simultaneous Projections Algorithms We evaluated all the variants given in Fig. 3, in both their additive and multiplicative forms. We denote the different variants using the notation `name`.A or `name.M` where `name` denotes the specific simultaneous projection scheme as given in Fig. 3 and the .A or .M suffix designate whether the update took an additive or multiplicative form. For example the additive SimProj algorithm is denoted by SimProj.A

MaxPA The algorithm extends the binary Passive-Aggressive family of algorithms (Crammer et al., 2006) to structured prediction problems. The algorithm uses a construction which is similar to our instantaneous primal objective (Eq. 2), and analogously attempts to obtain a feasible solution to its dual. The difference lies in the fact that the MaxPA algorithm focuses on a single instance whose margin constraint is mostly violated and updates *only* its corresponding dual variable, while fixing all other dual variables to zero. The single dual variable is then optimally computed. This update form constitutes a feasible solution to the instantaneous dual and casts a simple update for the online algorithm.

Optimal Solver The optimal solver algorithm employs a numerical solver on each iteration, and solves optimally the instantaneous primal given by Eq. (2). Specifically, we used the Pegasos algorithm from Shalev-Shwartz et al. (2007) to perform the optimization task. We chose this algorithm since it provides a simple solver which proved superior to second order methods in various classification tasks (Shalev-Shwartz et al., 2007).

Sopopo The Sopopo[1] algorithm is a novel algorithm for label ranking and is thus used only in our label ranking experiments with email data. The Sopopo algorithm computes the *optimal* solution to an instantaneous optimization problem similar cast by Eq. (2) while using a slightly different setting. We further elaborate on the different setting in Sec. 9.2.

9.1 Experiments with Synthetic Data

We tested the performance of the additive and the multiplicative variants of our algorithm in a series of experiments using synthetic data. Our goal in this section is to underscore the merits of our simultaneous projections approach in comparison with the commonly used max update techniques (MaxPA). Since it is often computationally prohibitive to run a full numerical solver on each iteration, we omitted the optimal solver from this set comparisons.

Before we describe the results of our experiments with synthetic data, let us first discuss the procedures used to generate the data. In order to compare both the additive and the multiplicative versions of our framework, we confined ourselves to the more restrictive setting of the multiplicative schemes as described in Sec. 8. Specifically, the data was generated by randomly constructing instances $\mathbf{x}^t \in \{-1,0,1\}^n$ and classification was performed by selecting a probability vector $\omega \in \Delta_n$. We used a sparse classifier where the number of relevant features in ω varied from 10% to 50% active features. The non-zero components of ω were chosen uniformly at random from $[0,1]$. We

1. The name Sopopo stands for SOft Projection Onto Polyhedra.

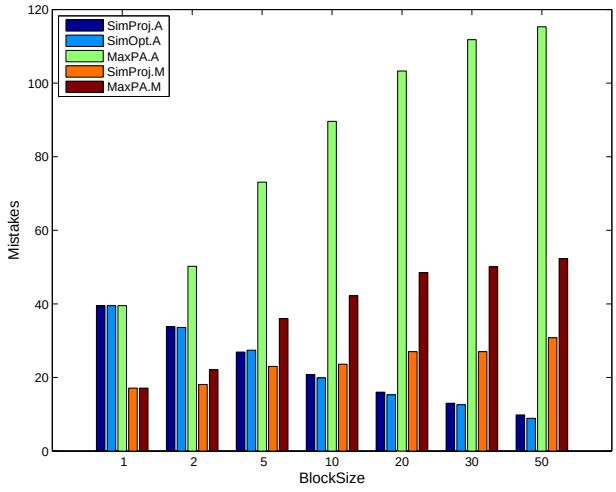


Figure 7: The number of mistakes suffered by the various the additive and multiplicative simultaneous projections methods. The performance of the algorithms is compared as a function of the block size.

then normalized the vector such that its L_1 norm would be one. We generated linearly separable data whose margin was calculated as follows. Each entry in $\mathbf{x}$ was set to 0 with probability p and otherwise it was chosen from $\{-1,1\}$ with equal probability. We then computed the value of γ for which 75% of all instances sampled from the process above would fall inside a margin of γ. We then repeatedly generated and rejected instances, until we managed to construct sufficient number of examples. We refer to a set examples grouped together into a single classification task as a block.

We ran each online experiment for 1000 trials and recorded the number of mistakes performed by the online algorithms. Each experiment was repeated 10 times. The results we present are the averages of these runs. For each experiment, we performed a search for a good value of C. We checked 9 values for C, ranging from 2^{-5} to 2^3. For the multiplicative variants, we also performed a search for a good value of γ by examining values in the range 0.5 to 2 times the margin used during the data generation process. We compared all simultaneous projection variants presented earlier, as well as the multiplicative and additive versions of the MaxPA update.

The first experiment with synthetic data assesses the performance of the various update schemes as a function of the block sizes. We used instances in $\{-1,0,1\}^{500}$ where ω contained 50 relevant features. The results are described in Fig. 7. We clearly see that while both schemes entertain the same mistake bound, in practice the SimProj algorithms always perform better than the maximal update. The difference in practical performance can be attributed to the fact that the simultaneous projections mechanism uses more information regarding the structure of the problem at hand.

Note that for both the *MaxPA.A* scheme and the multiplicative schemes the performance deteriorates as the block size increases. A converse trend is exhibited in the case of *SimProj.A* and *SimOpt.A*. One possible explanation for the improvement with block size increase may be observed by considering the geometrical structure of the instances. Recall that we generate uniformly selected linearly separable data. Thus, the update form the additive variants apply can be seen as performing the update using the average instance. For large blocks the average instance is equivalent to the vec-

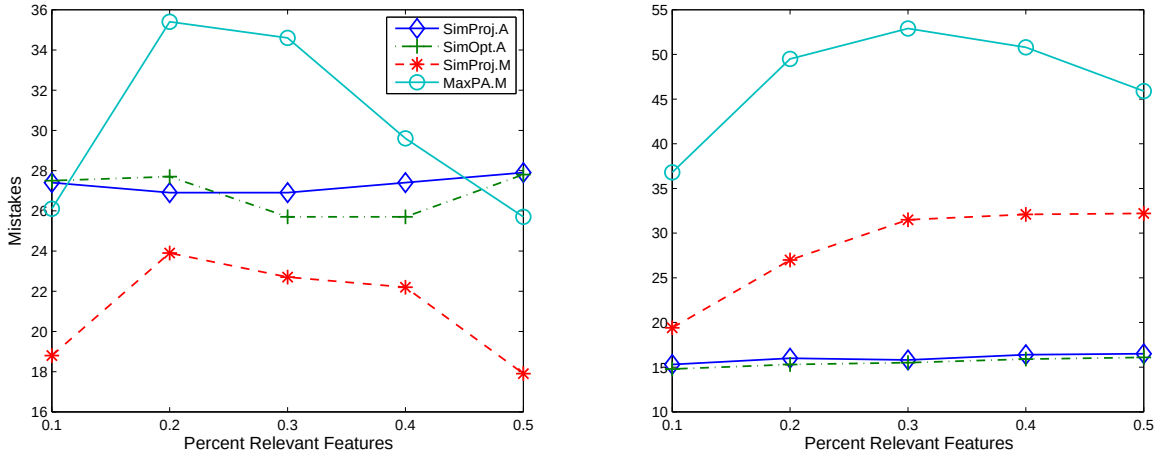


Figure 8: The performance of the additive and multiplicative simultaneous projections algorithms as a function of the sparsity of the hypothesis generating the data. Results are shown for block size of 5 instances (left) and of 20 instances (right).

tor used to describe the separating hyperplane. Such averaging does not occur in the multiplicative case, or the *MaxPA.A* scheme.

Our second experiment examines the effect of the sparsity of ω on the performance of the algorithms. As before, we used instances in $\{-1,0,1\}^{500}$. We varied the number of non-zero elements of ω from 50 to 250. The results of this experiments are plotted in Fig. 8. We ran this experiment with a block size of 5 instances per trial (Fig. 8 left) and a block size of 20 (Fig. 8 right). We omit the plot of *MaxPA.A* as its performance is much worse than any of the other algorithms. It is apparent that the additive versions are rather insensitive to the sparsity of the prediction matrix. The converse is true for the multiplicative variants. For both block sizes, we see that the performance of the multiplicative versions deteriorate as we increase the percentage of relevant features from 10% to 30%. This decrease in performance is then replaced with a gradual increase in performance once the number of relevant features is over 30%. This increase in performance may be attributed to the fact that there are more relevant features which are set approximately uniformly. Thus, the optimal solution is rather close to the initial vector and the multiplicative algorithm converges faster.

Our last experiment with synthetic data examines the effect of label noise on the performance of the simultaneous projections algorithms. We employed the same settings as in the previous experiment with instances of 500 dimensions and 50 non-zero entries in ω. After the data was generated, we chose to contaminate with label noise each trial with a fixed probability. If the block was selected for contamination, we flipped the label of each the instances in the block with probability 0.4. We repeated the experiment with varying probabilities of picking a block for contamination. We tested values from the set $\{0$ (no label noise), 0.05,0.1,0.15, $0.2\}$. To avoid too aggressive online updates, we increased the range of the search for C to be in $[2^{-9},2^{3}]$. The results of this experiment are plotted in Fig. 9. We ran this experiment with a block size of 5 instances per trials (Fig. 9 left) and block size of 20 (Fig. 9 right) instances per trial.

We can clearly see that the number of mistakes of all the SimProj variants scale linearly with the noise rate. It is also apparent that the number of mistakes of the MaxPA.A algorithm scales super

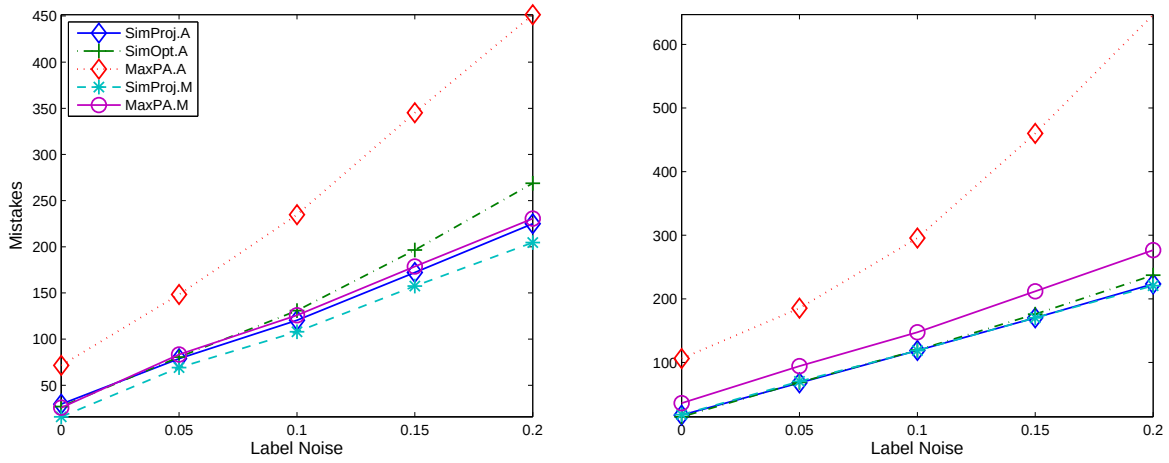


Figure 9: The number of mistakes of the additive and multiplicative simultaneous projections algorithms as a function of the label noise. Results are depicted for block sizes of 5 instances (left) and 20 instances (right) per trial.

username	k	m	Max-SP	SimPerc	ConProj	SimProj	SimOpt	OptSolv
beck-s	101	1972	48.2	48.4	48.8	**43.4**	46.9	45.5
farmer-d	25	3398	25.8	28.8	28.4	25.0	**24.2**	27.5
kaminski-v	41	4478	48.0	47.2	47.6	46.0	44.4	**44.0**
kitchen-l	47	4016	42.5	45.1	43.8	41.4	42.4	**40.3**
lokay-m	11	2490	20.8	24.1	23.9	**18.6**	20.6	20.3
sanders-r	30	1189	18.6	20.9	22.2	**17.7**	19.3	18.2
williams-w3	18	2770	2.8	3.5	3.4	2.7	2.8	**2.6**

Table 1: The percentage of online mistakes of the four additive variants compared to MaxPA and the optimal solver of each instantaneous problem. Experiments were performed on seven users of the Enron data set.

linearly with the noise rate. The simultaneous projections variants (both additive and multiplicative) exhibit the best performance. We see that for the smaller block sizes (Fig. 9 left) the best performing version is the *SimProj.M* variant. Note, however, that the variants of SimOpt perform worse than the variants of SimProj . This fact can possibly attributed to the more aggressive update taken by the SimOpt variant when a mistake occurs. As the number of instances per trial increases, the performance of all of simultaneous projections variants is comparable and they all perform better than any of the MaxPA variants.

9.2 Email Classification Experiments

We next tested performance of the different additive and multiplicative simultaneous projection methods described in Sec. 5 on multiclass email categorization tasks and compared them to previously studied algorithms for multiclass categorization. We compared our algorithms to the MaxPA

algorithms and to the optimal solver. The experiments were performed with the Enron email data set. The data set is available from http://www.cs.cmu.edu/∼enron/enron_mail_030204.tar.gz. The learning goal is to correctly classify email messages into user defined folders. Thus, the instances in this data set are email messages, while the set of classes are the user defined folders denoted by $\{1,\ldots,k\}$. We ran the experiments on the sequence of email messages from 7 different users.

Since each user employs different criteria for email classification, we treated each person as a separate online learning problem. We represented each email message as a vector with a component for every word in the corpus. In order to apply our algorithms, we next describe the class-dependent map we use for the additive algorithms. On each trial, and for each class r, we constructed class-dependent vectors as follows. We set $\phi_j(\mathbf{x}^t,r)$ to 2 if the j'th word appeared in the message and it also appeared in a fifth of the messages previously assigned to folder r. Similarly, we set $\phi_j(\mathbf{x}^t,r)$ to -1 if the j'th word appeared in the message but appeared in less than 2 percent of previous messages. In all other cases, we set $\phi_j(\mathbf{x}^t,r)$ to 0. This class-dependent construction is very similar to the construction used in Fink et al. (2006), which yielded high classification accuracy. Next, we employed the mapping described in Sec. 3, and defined a set of $k-1$ instances for each message as follows. Let the relevant class of an instance be denoted by y. Then, for every irrelevant class $s \neq y$, we define an instance $\mathbf{x}_s^t = \phi(\mathbf{x}^t,y) - \phi(\mathbf{x}^t,s)$ and set its label to 1. All these instances were combined into a single matrix $\mathbf{X}^t$ and were provided to the algorithm in trial t.

For the multiplicative algorithms we took a slightly different approach. Recall that the multiplicative variants assume that the components of each instance are in $\{-1,0,1\}$. Hence, in order to adhere to this requirement, $\phi_j(\mathbf{x}^t,r)$ was set to 1 if the j'th word appeared in the message and it also appeared in a fifth of the messages previously assigned to folder r and to 0 in all other cases. Note that this feature generation is performed without knowing the correct label of the instance $\mathbf{x}^t$ and is thus limited to the information available to the online algorithm.

We repeated all tests for 11 values of the trade-off parameter C, testing values from 2^{-5} to 2^5. For each algorithm we present the results for the best choice of C. We first compare the results of the various additive approaches. The results of this experiments are described in Fig. 1. It is apparent that the SimProj and SimOpt variants consistently perform better than the MaxPA variant, and their performance is on par with the performance of the optimal solver. It is interesting to note that in 3 of the 7 users, the SimProj algorithm actually performs better than the optimal solver. The superior performance of the SimProj algorithm may most likely attributed to to the fact that the optimal solver is more aggressive in its update, and is thus more sensitive to noise. We can also see that the SimOpt version, while guaranteeing a larger increase in the global dual, does not necessarily assure better empirical performance. The performances of SimPerc and ConProj are comparable with no obvious winner. Last, we would like to note that the computational cost of the simultaneous projections algorithms is comparable to that of the MaxPA algorithm.

In Fig. 10 we plot the relative number of mistakes of each algorithm with respect to the number of mistakes made by the optimal solver as a function of the number trials for 6 of the 7 users. (The user williams-w3 was omitted as he classifies most his emails into a single folder.) In order to keep the graphs intelligible, we use the optimal solver algorithm as the baseline and plot the difference in performance of the other additive variants. The graphs clearly indicate the superiority of the SimProj and SimOpt variants over the MaxPA algorithm, and show that SimProj often exhibits the best accuracy.

We next compared the performance of the multiplicative and additive variants. The results of this experiments are summarized in Table 2. Observe that the multiplicative versions usually

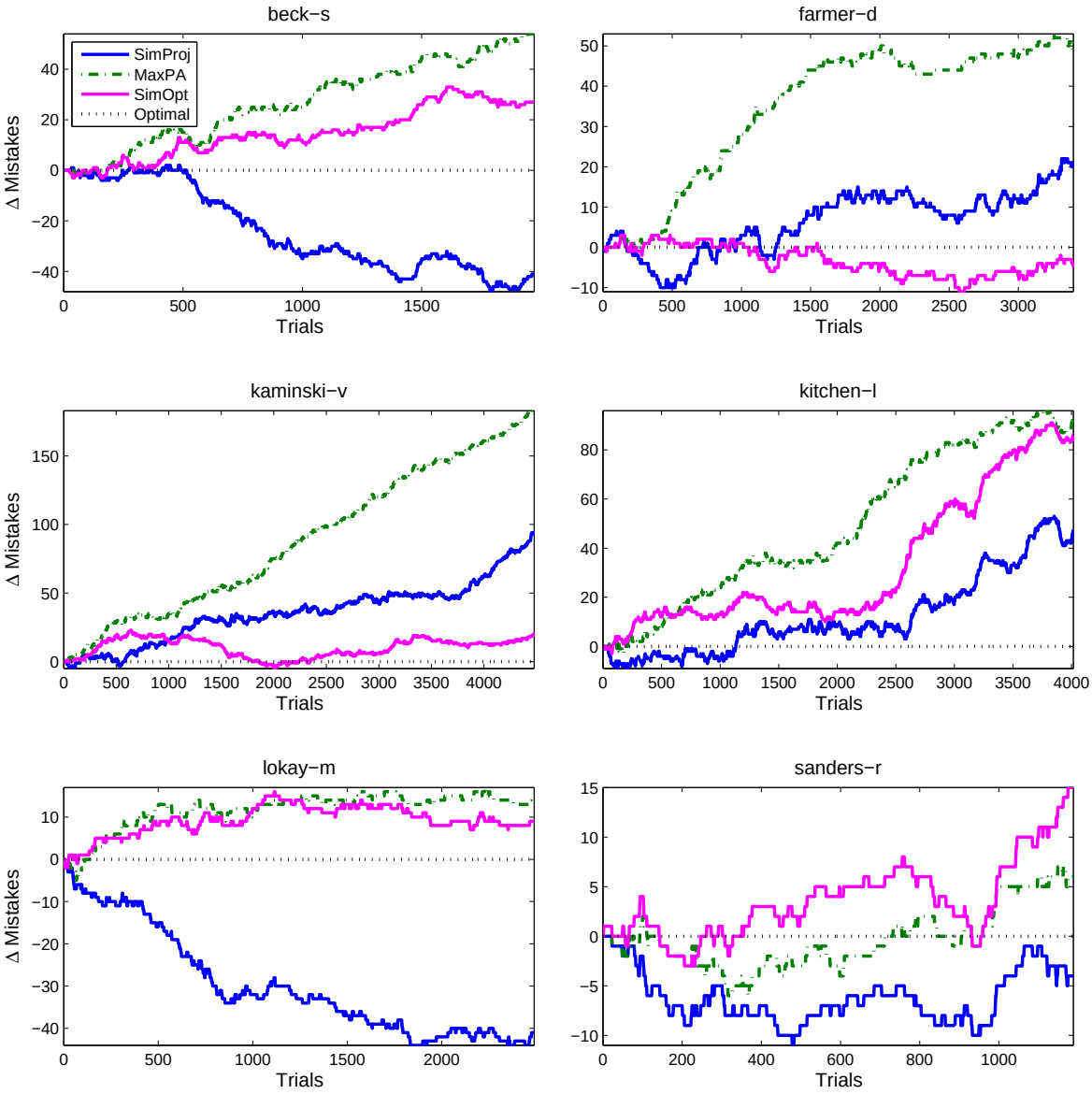


Figure 10: The cumulative number of mistakes of the simultaneous projection algorithms compared with the performance of the Max-PA algorithm and the optimal solver as a function of the number of trials. We plot the difference in the number of mistakes between each algorithm and the optimal solver.

			Additive				Multiplicative			
username	k	m	Max-SP	SimPerc	ConProj	SimProj	Max-SP	SimPerc	ConProj	SimProj
beck-s	101	1972	48.2	48.4	48.8	**43.4**	45.0	43.7	43.6	45.8
farmer-d	25	3398	25.8	28.8	28.4	**25.0**	33.1	35.1	34.8	33.0
kaminski-v	41	4478	48.0	47.2	47.6	**46.0**	50.0	49.8	49.8	49.8
kitchen-l	47	4016	42.5	45.1	43.8	**41.4**	46.8	47.3	47.2	46.5
lokay-m	11	2490	20.8	24.1	23.9	**18.6**	21.8	22.9	22.9	21.4
sanders-r	30	1189	18.6	20.9	22.2	**17.7**	17.8	17.9	17.9	19.3
williams-w3	18	2770	2.8	3.5	3.4	2.7	**2.6**	2.7	**2.6**	2.7

Table 2: The percentage of online mistakes of three additive variants and the MaxPA algorithm compared to their multiplicative counterparts. Experiments were performed on seven users of the Enron data set.

perform on par or slightly worse than the additive versions. This possibly surprising result may be partially attributed to the slightly different feature selection process we used for the multiplicative algorithms. The result also underscores the conjecture that we surfaced above when discussing the synthetic experiments. Namely, the additive simultaneous projections algorithms seem to better capture the structure of the data at hand. The multiplicative versions, however, seem to be less sensitive to the trade-off parameter C taking a preference to the larger values in our test setting.

In our last experiment, we compared the results of our algorithms to the Sopopo algorithm from Shalev-Shwartz and Singer (2006b). The results of this experiment are described in Table 3. Before we discuss the results of this comparison, it is important to note the difference in the model the algorithms use. The algorithms we compare can be roughly divided into two classes of learning algorithms: *single*-prototype algorithms and *multi*-prototype algorithms. As the name imply, the single prototype algorithms maintain a single hypothesis on each online trial. The prediction is obtained by taking the inner-product of the hypothesis with some class dependent map of the instance at hand. The class attaining the highest score is considered the predicted label. All the simultaneous projections algorithms as well as the single prototype version of MaxPA fall into this category. Multi-prototype algorithms take a different approach. On each trial, the online algorithm maintains an hypothesis *for each* possible output class. In order to make a prediction, the algorithm computes the inner product between *each* hypothesis and the instance at hand. The class attaining the largest product is the predicted label. We can see that the various SimProj variants are comparable to the Sopopo algorithm, while the former often performs better (4 of the 7 users we have). It is worth nothing that the Sopopo algorithm exploits the specific settings present in multi-prototype multiclass problems, and efficiently finds the optimum of a projection problem on each trial while our algorithms only find an approximate solution. However, Sopopo is a multi prototype algorithm and thus employs a larger hypothesis space which is more difficult to learn in an online setting. In addition, by employing a single vector representation of the email message, Sopopo cannot benefit from the on-the-fly feature selection which results in class-dependent features.

10. Discussion

We presented a new approach for online categorization with complex output structure. Our algorithms decouples the complex optimization task into multiple sub-tasks, each of which is simple

			Single Prototype					Multi Prototype	
username	k	m	MaxPA	SimPerc	ConProj	SimProj	SimOpt	MaxPA	Sopopo
beck-s	101	1972	48.2	48.4	48.8	**43.4**	46.9	56.0	53.2
farmer-d	25	3398	25.8	28.8	28.4	25.0	24.2	24.0	**23.0**
kaminski-v	41	4478	48.0	47.2	47.6	46.0	44.4	45.9	**43.4**
kitchen-l	47	4016	42.5	45.1	43.8	41.4	42.4	42.2	**40.9**
lokay-m	11	2490	20.8	24.1	23.9	**18.6**	20.6	20.0	19.0
sanders-r	30	1189	18.6	20.9	22.2	**17.7**	19.3	27.9	26.9
williams-w3	18	2770	2.8	3.5	3.4	**2.7**	2.8	4.1	4.1

Table 3: The percentage of online mistakes of four additive simultaneous projection algorithms. The simultaneous projection algorithms are compared with MaxPA (Single-prototype (SP) and Multi-prototype (MP)) and the Sopopo algorithm. Experiments were performed on seven users of the Enron data set.

enough to be solved analytically. While the dual representation of the online problem imposes a global constraint on *all* the dual variables, namely $\sum_j \alpha_j^t \leq C$, our framework of simultaneous projections which are followed by averaging the solutions automatically adheres with this constraint and hence constitute a feasible solution. It is worthwhile noting that our approach can also cope with *multiple* constraints of the more general form $\sum_j \nu_j \alpha_j \leq C$, where $\nu_j \geq 0$ for all j. The box constraint implied for each individual projection problem distils to $0 \leq \alpha_j \leq C/\nu_j$ and thus the simultaneous projection algorithm can be used verbatim.

The main scope of this paper is prediction tasks for complex structured decision problems, such as multiclass classification. We approach this problem by first describing the structured problem as a complex optimization problem dealing with multiple binary problems simultaneously. We then use our simultaneous projections scheme to propose a feasible solution to the optimization problem which competes with any decomposition loss (see Sec. 7).

While such an approach is very natural for various structured problems, it is interesting to consider settings in which multiple unrelated binary problems, should be served simultaneously. This approach was the basis for our synthetic experiments, which showed us that the simultaneous projections methods perform much better than the MaxPA approach even though their theoretical bound is similar. One possible explanation of this phenomenon may be attributed to the structure of the space spanned by the examples. In order to illustrate this point, consider for example, the case where all instances on trial t are of similar norm and are orthogonal to each other. The update performed by the simultaneous projections approach is thus optimal. If, on the other hand, all instances received on trial t are exactly the same, then the simultaneous projections approach cannot hope to attain anything better than the MaxPA algorithm. These two extreme cases suggests that further analysis may show that the geometrical structure of the data may shed more light on the progress attained by the simultaneous projections approach.

It is also interesting to explore settings in which the simultaneous projections approach is not immediately applicable. The simultaneous projections approach easily captures the structural requirements expressed by the box constraint $\sum_{j=1}^{k_t} \alpha_j^t \leq C$. While there are many practical problems where such constraints suffice to capture the structure of the problem, more complex constraints are quite prevalent. A notable examples for such a complex setting is the framework of the max-margin

Markov (MMM) networks (Taskar et al., 2003). While the original learning setting of MMM networks was cast for the batch setting, an equivalent online formulation can be easily obtained. In the MMM framework, the dual problems has dual variables whose number is *exponential* in the original size of the problem. These variables are tied via a simple box constraint. The dual is then transformed into an equivalent form with a much smaller number of variables which are strongly tied with multiple constraints involving all these new variables. While the simultaneous projections approach is well suited for the original formulation, the exponential size of the problem renders such approach unsuitable. On the other hand, the simultaneous projections approach cannot easily construct a feasible dual solution where multiple equality constraints are required. It is thus interesting to explore alternative approaches in which the direct dual whose size is infeasible is reduced to many reduced smaller problems, and only a polynomial subset of which are considered and solved.

Appendix A. Derivation of the Dual Problems

In this section we derive the dual problems of the primal problems presented in the main sections. We start with the derivation of the dual of the optimization problem given in Eq. (2). Using Lagrange multipliers, we rewrite Eq. (2) as follows

$$\min_{\omega\in\mathbb{R}^n,\xi}\ \max_{\alpha^t\geq 0,\beta\geq 0}\ \frac{1}{2}\left\|\omega-\omega^t\right\|^2+C\xi+\sum_{j=1}^{k_t}\alpha_j^t\left(1-\xi-y_j^t\left(\omega\cdot\mathbf{x}_j^t\right)\right)-\beta\xi_t \ .$$

We rearrange the terms in the above equation and rewrite it as follows,

$$\min_{\omega\in\mathbb{R}^n,\xi}\ \max_{\alpha^t\geq 0,\beta\geq 0}\ \sum_{j=1}^{k_t}\alpha_j^t+\frac{1}{2}\left\|\omega-\omega^t\right\|^2-\sum_{j=1}^{k_t}\alpha_j^t y_j^t\left(\omega\cdot\mathbf{x}_j^t\right)+\xi\left(C-\sum_{j=1}^{k_t}\alpha_j^t-\beta\right) \ . \tag{26}$$

The dual of Eq. (26) is attained by changing the order of the min and max and is given by

$$\max_{\alpha^t\geq 0,\beta\geq 0}\ \min_{\omega\in\mathbb{R}^n}\ \sum_{j=1}^{k_t}\alpha_j^t+\frac{1}{2}\left\|\omega-\omega^t\right\|^2-\sum_{j=1}^{k_t}\alpha_j^t y_j^t\left(\omega\cdot\mathbf{x}_j^t\right)+\min_{\xi}\xi\left(C-\sum_{j=1}^{k_t}\alpha_j^t-\beta\right) \ . \tag{27}$$

The equation above can be written equivalently as

$$\max_{\alpha^t\geq 0}\ \min_{\omega\in\mathbb{R}^n}\ \underbrace{\sum_{j=1}^{k_t}\alpha_j^t+\frac{1}{2}\left\|\omega-\omega^t\right\|^2-\sum_{j=1}^{k_t}\alpha_j^t y_j^t\left(\omega\cdot\mathbf{x}_j^t\right)}_{\mathcal{L}(\alpha,\omega)}\qquad \text{s.t.}\qquad \sum_{j=1}^{k_t}\alpha_j^t\leq C \ . \tag{28}$$

In order to see that Eq. (28) and Eq. (27) are equivalent, note that the expression

$$\min_{\xi}\xi\left(C-\sum_{j=1}^{k_t}-\beta\right) \ ,$$

takes the value of $-\infty$ when $\sum_{j=1}^{k_t}\alpha_j^t+\beta\neq C$. Such an assignment for α^t and β cannot constitute the optimal solution for the maximization problem. The constraint $\beta\geq 0$ thus translates to the constraint $\sum_{j=1}^{k_t}\alpha_j^t\leq C$. Fixing α^t, the derivative of $\mathcal{L}$ with respect to ω is given by

$$\frac{\partial\mathcal{L}}{\partial\omega}=\omega-\omega^t-\sum_{j=1}^{k_t}\alpha_j^t y_j^t\mathbf{x}_j^t \ .$$

Comparing the derivative to 0, yields the following equation, $\omega = \omega^t + \sum_{j=1}^{k_t} \alpha_j^t y_j^t \mathbf{x}_j^t$. Plugging this equality of ω Eq. (28) yields the following simplified constrained optimization problem,

$$\begin{aligned} &\max_{\alpha^t \geq 0} \sum_{j=1}^{k_t} \alpha_j^t + \frac{1}{2} \left\| \sum_{j=1}^{k_t} \alpha_j^t y_j^t \mathbf{x}_j^t \right\|^2 - \sum_{j=1}^{k_t} \alpha_j^t y_j^t \left(\left(\omega^t + \sum_{l=1}^{k_t} \alpha_l^t y_l^t \mathbf{x}_l^t \right) \cdot \mathbf{x}_j^t \right) \\ &\text{s.t.} \qquad \sum_{j=1}^{k_t} \alpha_j^t \leq C \end{aligned} .$$

Rearranging the terms and adding constants which do not depend of α^t, we obtain the following dual problem,

$$\begin{aligned} &\max_{\alpha^t} \sum_{j=1}^{k_t} \alpha_j^t - \frac{1}{2} \left\| \omega^t + \sum_{j=1}^{k_t} \alpha_j^t y_j^t \mathbf{x}_j^t \right\|^2 \\ &\text{s.t.} \qquad \sum_{j=1}^{k_t} \alpha_j^t \leq C \qquad \forall j \in [k_t] : \ \alpha_j^t \geq 0 \end{aligned} .$$

We now turn our attention to the derivation of the dual of the optimization problem given by Eq. (20). Eq. (20) can be rewritten as follows

$$\begin{aligned} &\min_{\omega \in \Delta_n, \xi \geq 0} \sum_{i=1}^{n} \omega_i \log \frac{\omega_i}{\omega_i^t} + C\xi \\ &\text{s.t.} \qquad \forall j \in [k_t] : \ y_j^t \left(\omega \cdot \mathbf{x}_j^t \right) \geq \gamma - \xi \qquad \xi_l \geq 0 \end{aligned} . \tag{29}$$

We again use Lagrange theory and rewrite the optimization task above as,

$$\min_{\omega \in \Delta_n, \xi \geq 0} \max_{\alpha_j^t \geq 0} \sum_{i=1}^{n} \omega_i \log \frac{\omega_i}{\omega_i^t} + C\xi + \sum_{j=1}^{k_t} \alpha_j^t \left(\gamma - \xi - y_j^t \left(\omega \cdot \mathbf{x}_j^t \right) \right) .$$

Rearranging the terms in the above expression we obtain

$$\min_{\omega \in \Delta_n, \xi \geq 0} \max_{\alpha_j^t \geq 0} \sum_{i=1}^{n} \omega_i \log \frac{\omega_i}{\omega_i^t} + \xi \left(C - \sum_{j=1}^{k_t} \alpha_j^t \right) + \sum_{j=1}^{k_t} \alpha_j^t \left(\gamma - y_j^t \left(\omega \cdot \mathbf{x}_j^t \right) \right) . \tag{30}$$

The dual of Eq. (30) is thus obtained by reversing the order of the min and max and is thus given by

$$\max_{\alpha_j^t \geq 0} \min_{\omega \in \Delta_n, \xi \geq 0} \sum_{i=1}^{n} \omega_i \log \frac{\omega_i}{\omega_i^t} + \xi \left(C - \sum_{j=1}^{k_t} \alpha_j^t \right) + \sum_{j=1}^{k_t} \alpha_j^t \left(\gamma - y_j^t \left(\omega \cdot \mathbf{x}_j^t \right) \right) . \tag{31}$$

The equation above can be rewritten equivalently as follows

$$\begin{aligned} &\max_{\alpha_j^t, \beta^t} \min_{\omega \in \mathbb{R}^n} \sum_{i=1}^{n} \omega_i \log \frac{\omega_i}{\omega_i^t} + \sum_{j=1}^{k_t} \alpha_j^t \left(\gamma - y_j^t \left(\omega \cdot \mathbf{x}_j^t \right) \right) + \beta^t \left(\sum_{i=1}^{n} \omega_i - 1 \right) \\ &\text{s.t.} \qquad \sum_{j=1}^{k_t} \alpha_j^t \leq C \qquad \forall j : \alpha_j^t \geq 0 \end{aligned} . \tag{32}$$

In order to see Eq. (32) and Eq. (31) are equivalent, note that the expression $\min_\xi \xi \left(C - \sum_{j=1}^{k_t} \alpha_j^t\right)$ takes the value of $-\infty$ when $\sum_{j=1}^{k_t} \alpha_j^t > C$. Since our goal is to maximize the dual, such solution cannot constitute the optimal assignment for α^t. Similarly, when $\sum_{i=1}^{n} \omega_i \neq 1$, the maximization takes the value ∞, thus such a solution cannot constitute the optimal assignment of minimization problem. Finally, we may ignore the constraint $\omega_i \geq 0$ as the optimal solution to the above problem always yields a solution that satisfies this constraint.

In order to further analyze the Eq. (32), let us first denote the vector $\sum_{j=1}^{k_t} \alpha_j^t y_j^t \mathbf{x}_j^t$ by τ. Taking the derivative of Eq. (32) with respect to ω and comparing the result to 0 yields the following,

$$\omega_i = \omega_i^t e^{\tau_i - 1 + \beta} \quad .$$

Recall that β is the Lagrange multiplier associated with the constraint $\sum_{i=1}^{n} \omega_i = 1$, thus $e^{-1+\beta}$ serves as a normalization constant, and the optimal assignment for ω_i takes the following form.

$$\omega_i = \frac{\omega_i^t e^{\tau_i}}{\sum_{l=1}^{n} \omega_l^t e^{\tau_l}} \quad .$$

Plugging the value for ω into Eq. (32) yields the following dual problem for Eq. (29)

$$\max_{\alpha_j^t} \gamma \sum_{j=1}^{k_t} \alpha_j^t - \log\left(\sum_{i=1}^{n} \omega_i^t e^{\tau_i}\right) \qquad \text{s.t.} \qquad \sum_{j=1}^{k_t} \alpha_j^t \leq C \qquad \forall j : \alpha_j^t \geq 0 \qquad \tau = \sum_{j=1}^{k_t} \alpha_j^t y_j^t \mathbf{x}_j^t \quad .$$

Appendix B. Derivation of the SimPerc Mistake Bound

Thm. 2 assures us that the number of mistakes performed by the SimPerc algorithm is bound by

$$\frac{\frac{1}{2}\|\omega^\star\|^2 + C\sum_{t=1}^{T} \ell(\omega^\star; (\mathbf{X}^t, \mathbf{y}^t))}{C - \frac{1}{2}C^2R^2} \quad . \tag{33}$$

Observe that the prediction made by the SimPerc algorithm does not depend on the value of C. We may thus choose C as to tighten the above bound. Assume $R = 1$ and denote $\sum_{t=1}^{T} \ell(\omega^\star; (\mathbf{X}^t, \mathbf{y}^t))$ by L and $\|\omega^\star\|^2$ by B. It is easy to verify that if $L = 0$, the optimal assignment for C is attained by setting $C = 1$. The bound in this case distills to B. Otherwise, assume $L > 0$. Then, Eq. (33) can be written as

$$\varphi(C) = \frac{\frac{1}{2}B + CL}{C - \frac{1}{2}C^2} \quad . \tag{34}$$

The above expression is convex with respect to the parameter C. Hence, in order to find the optimal value of C, it suffices to take the derivative of Eq. (34) with respect to C and compare the result to 0, which yields,

$$\frac{L(C - \frac{1}{2}C^2) - (\frac{1}{2}B + CL)(1 - C)}{\left(C - \frac{1}{2}C^2\right)^2} = 0 \quad ,$$

which implies that,

$$LC^2 + BC - B = 0 \quad . \tag{35}$$

The largest root of Eq. (35) is given by

$$\begin{aligned} C = \frac{-B+\sqrt{B^2+4BL}}{2L} &= \frac{-B+\sqrt{B^2+4BL}}{2L} \cdot \frac{-B-\sqrt{B^2+4BL}}{-B-\sqrt{B^2+4BL}} \\ &= \frac{B^2-B^2-4BL}{2L\left(-B-\sqrt{B^2+4BL}\right)} \\ &= \frac{2}{\left(1+\sqrt{1+4\frac{L}{B}}\right)} \end{aligned} \quad ,$$

It is easy to verify that for $L > 0$ this value of C lies in $(0,2)$ and thus constitutes the optimal solution of Eq. (34). Plugging Eq. (35) into Eq. (33) yields the following

$$\begin{aligned} \frac{\frac{1}{2}B+CL}{C-\frac{1}{2}C^2} &= \frac{\frac{B}{C}+L}{1-\frac{1}{2}C} \\ &= \frac{\frac{1}{2}B\left(1+\sqrt{1+4\frac{L}{B}}\right)+2L}{2\left(1-\frac{1}{2}\frac{2}{\left(1+\sqrt{1+4\frac{L}{B}}\right)}\right)} \\ &= \frac{\left(1+\sqrt{1+4\frac{L}{B}}\right)\left(\frac{1}{2}B\left(1+\sqrt{1+4\frac{L}{B}}\right)+2L\right)}{2\left(\sqrt{1+4\frac{L}{B}}\right)} \\ &= \frac{\frac{1}{2}B+2L}{\frac{1}{2}\left(\sqrt{1+4\frac{L}{B}}\right)} + \frac{1}{4}B + \frac{1}{4}B\left(1+\sqrt{1+4\frac{L}{B}}\right)+L \\ &= \frac{1}{2}B+L+\frac{1}{2}\sqrt{B^2+4LB} \end{aligned} \quad .$$

Using the definition of L and B to expand the above expression completes our proof.

Appendix C. An Analytic Solution for the Multiplicative Framework

Recall that throughout Sec. 8 section we assumed that the entries of each instance $\mathbf{x}_j^t$ lie in $\{-1,0,1\}$. On trial t we would like to find the optimal solution to the reduced problem given by Eq. (23). To do so we recall the notation used in Sec. 8,

$$W_j^+ = \frac{\sum_{i:y_j^t x_{ji}^t=1} \omega_i^t}{\sum_{l=1}^n \omega_l^t} \;,\; W_j^- = \frac{\sum_{i:y_j^t x_{ji}^t=-1} \omega_i^t}{\sum_{l=1}^n \omega_l^t} \;,\; \text{and } W_j^0 = 1 - W_j^+ - W_j^- \quad .$$

The optimization problem defined by Eq. (23) can thus be rewritten as

$$\begin{aligned} &\gamma\alpha_j^t - \log\left(W_j^+ e^{\alpha_j^t} + W_j^- e^{-\alpha_j^t} + W_j^0\right) \\ &\text{s.t.} \qquad 0 \le \alpha_j^t \le C \end{aligned} \quad .$$

Taking the derivative of the above equation with respect to α_j^t and comparing the result to zero yields the following,

$$\gamma - \frac{W^+ e^{\alpha_j^t} - W^- e^{-\alpha_j^t}}{W^+ e^{\alpha_j^t} + W^- e^{-\alpha_j^t} + W^0} = 0 \ . \tag{36}$$

Rearranging terms, Eq. (36) reduces to

$$\gamma\left(W^+ e^{\alpha_j^t} + W^- e^{-\alpha_j^t} + W^0\right) = W^+ e^{\alpha_j^t} - W^- e^{-\alpha_j^t} \ \Rightarrow \ (1-\gamma)W^+ e^{\alpha_j^t} - (1+\gamma)W^- e^{-\alpha_j^t} - \gamma W^0 = 0 \ .$$

For brevity, we denote $e^{\alpha_j^t}$ by β. The equation above is equivalent to the following equation in β,

$$(1-\gamma)W^+\beta - (1+\gamma)W^-\beta^{-1} - \gamma W^0 = 0 \ .$$

Multiplying both sides of the above equation by β, we obtain the following quadratic equation

$$(1-\gamma)W^+\beta^2 - \gamma W^0\beta - (1+\gamma)W^- \ ,$$

whose largest root (the second root is negative) is

$$\beta = \frac{\gamma W^0 + \sqrt{\gamma^2 (W^0)^2 + 4(1-\gamma^2)W^+W^-}}{2(1-\gamma)W^+} \ .$$

Since α_j^t must reside in $[0, C]$ we set α_j^t to be the minimum between $\log(\beta)$ and C, yielding,

$$\alpha_j^t = \min\left\{C, \log\left(\frac{\gamma W^0 + \sqrt{\gamma^2 (W^0)^2 + 4(1-\gamma^2)W^+W^-}}{2(1-\gamma)W^+}\right)\right\} \ .$$

References

E. L. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Machine Learning: Proceedings of the Seventeenth International Conference*, 2000.

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 Mathematical Physics*, 7:200–217, 1967.

Y. Censor and S.A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, New York, NY, USA, 1997.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47, 2002.

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, Mar 2006.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, January 1995.

M. Fink, S. Shalev-Shwartz, Y. Singer, and S. Ullman. Online multiclass learning by interclass hypothesis sharing. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(1): 451–471, 1998.

C. Hildreth. A quadratic programming procedure. *Naval Research Logistics Quarterly*, 4:79–85, 1957. Erratum, ibidem, p.361.

J. Kivinen and M. Warmuth. Relative loss bounds for multidimensional regression problems. *Journal of Machine Learning*, 45(3):301–329, July 2001.

J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

A.R. De Pierro and A.N. Iusem. A relaxed version of bregman's method for convex programming. *Journal of Optimization Theory and Applications*, 51:421–440, 1986.

J. C. 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*. MIT Press, 1998.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):1–40, 1999.

R.E. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 32(2/3), 2000.

S. Shalev-Shwartz and Y. Singer. Online learning meets optimization in the dual. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, 2006a.

S. Shalev-Shwartz and Y. Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 7 (July):1567–1599, 2006b.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 17*, 2003.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, April 1999.