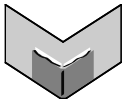
The Journal of Machine Learning Research
Volume 15
Print-Archive Edition

Pages 1–1370



Microtome Publishing
Brookline, Massachusetts
www.mtome.com

The Journal of Machine Learning Research
Volume 15
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 2014.

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 15, 2014

Journal of Machine Learning Research 15 (2014) 1-58 Submitted 7/10; Revised 7/12; Published 1/14

Bridging Viterbi and Posterior Decoding: A Generalized Risk Approach to Hidden Path Inference Based on Hidden Markov Models

Jüri Lember JURI.LEMBER@UT.EE
Institute of Mathematical Statistics
Tartu University
J. Liivi 2-507, Tartu, 50409, Estonia

Alexey A. Koloydenko ALEXEY.KOLOYDENKO@RHUL.AC.UK
Department of Mathematics
Royal Holloway University of London
Egham, TW20 0EX, UK

Editor: Richard Maclin

Abstract

Motivated by the unceasing interest in hidden Markov models (HMMs), this paper re-examines hidden path inference in these models, using primarily a risk-based framework. While the most common *maximum a posteriori* (MAP), or Viterbi, path estimator and the *minimum error*, or *Posterior Decoder* (PD) have long been around, other path estimators, or decoders, have been either only hinted at or applied more recently and in dedicated applications generally unfamiliar to the statistical learning community. Over a decade ago, however, a family of algorithmically defined decoders aiming to hybridize the two standard ones was proposed elsewhere. The present paper gives a careful analysis of this hybridization approach, identifies several problems and issues with it and other previously proposed approaches, and proposes practical resolutions of those. Furthermore, simple modifications of the classical criteria for hidden path recognition are shown to lead to a new class of decoders. Dynamic programming algorithms to compute these decoders in the usual forward-backward manner are presented. A particularly interesting subclass of such estimators can be also viewed as hybrids of the MAP and PD estimators. Similar to previously proposed MAP-PD hybrids, the new class is parameterized by a small number of tunable parameters. Unlike their algorithmic predecessors, the new risk-based decoders are more clearly interpretable, and, most importantly, work "out-of-the box" in practice, which is demonstrated on some real bioinformatics tasks and data. Some further generalizations and applications are discussed in the conclusion.

Keywords: admissible path, decoder, HMM, hybrid, interpolation, MAP sequence, minimum error, optimal accuracy, power transform, risk, segmental classification, symbol-by-symbol, posterior decoding, Viterbi algorithm

1. Introduction

Besides their classical and traditional applications in signal processing and communications (Viterbi, 1967; Bahl et al., 1974; Hayes et al., 1982; Brushe et al., 1998) (see also further references in Cappé et al., 2005) and speech recognition (Huang et al., 1990; Jelinek, 1976,

2001; McDermott and Hazen, 2004; Ney et al., 1994; Padmanabhan and Picheny, 2002; Rabiner and Juang, 1993; Rabiner et al., 1986; Shu et al., 2003; Steinbiss et al., 1995; Ström et al., 1999), hidden Markov models have recently become indispensable in computational biology and bioinformatics (Burge and Karlin, 1997; Durbin et al., 1998; Eddy, 2004; Krogh, 1998; Brejová et al., 2007b; Majoros and Ohler, 2007) as well as in natural language modeling (Manning and Schütze, 1999; Vogel et al., 1996) and information security (Mason et al., 2006).

At the same time, their spatial extensions, known as hidden Markov random field models (HMRFM), have been immensely influential in spatial statistics (Besag and Green, 1993; Green and Richardson, 2002; Künsch et al., 1995; McGrory et al., 2009), and particularly in image analysis, restoration, and segmentation (Besag, 1986; Geman and Geman, 1984; Li et al., 2000; Marroquin et al., 2003; Winkler, 2003). Indeed, hidden Markov models have been called 'one of the most successful statistical modeling ideas that have [emerged] in the last forty years' (Cappé et al., 2005).

HM(RF)Ms owe much of their success to the following: The posterior distribution of the hidden layer inherits the Markov property from the prior distribution (although the posterior distribution is generally inhomogeneous even if the prior distribution is homogeneous). At the same time, the marginal law of the observed layer can still include global, that is non-Markovian, dependence, hence the richness of the observed system (Künsch et al., 1995).

The Markov property of the posterior distribution and the conditional independence of the observed variables given the hidden ones, have naturally led to a number of computationally feasible methods for inference about the hidden realizations as well as model parameters. HMMs are also naturally a special case of *graphical models* (Lauritzen, 1996; Bishop, 2006, Chap. 8).

HMMs, or one dimensional HMRFMs, have been particularly popular not least due to the fact that the linear order of the indexing set (usually associated with time) makes exploration of hidden realizations relatively straightforward from the computational viewpoint. In contrast, higher dimensional HMRFMs generally require approximate, possibly stochastic, techniques in order to compute optimal configurations of the hidden field (Cocozza-Thivent and Bekkhoucha, 1993; Joshi et al., 2006; Winkler, 2003; McGrory et al., 2009). In particular, a *maximum a posteriori* (MAP) estimator of the hidden layer of an HMM is efficiently and exactly computed by a dynamic programming algorithm bearing the name of Viterbi, whereas a general higher dimensional HMRFM would employ, for example, a simulated annealing type method (Geman and Geman, 1984; Winkler, 2003) to produce approximate solutions to the same task.

There are also various useful extensions of the ordinary HMM, such as variable duration semi-Markov models, coupled HMMs (Brand et al., 1997), and factorial HMMs (Bishop, 2006, Chap. 13), etc. All of the material in this paper is applicable to those extensions in a straightforward way. However, to simplify the exposition we focus below on the ordinary HMM.

1.1 Notation and Main Ingredients

We adopt the machine and statistical learning convention, referring to the hidden and observed processes as Y and X, respectively, in effect reversing the convention that is more

commonly used in the HMM context. Thus, let $Y = \{Y_t\}_{t\geq 1}$ be a Markov chain with state space $S = \{1, \ldots, K\}$, $K > 1$, and initial probabilities $\pi_s = P(Y_1 = s)$, $s \in S$. Although we include inhomogeneous chains in most of what follows, for brevity we will still be suppressing the time index wherever this does not cause ambiguity. Hence, we write $\mathbb{P} = (p_{ij})_{i,j\in S}$ for all transition matrices. Let $X = \{X_t\}_{t\geq 1}$ be a process with the following properties. First, given $\{Y_t\}_{t\geq 1}$, the random variables $\{X_t\}_{t\geq 1}$ are conditionally independent. Second, for each $t = 1, 2, \ldots$, the distribution of X_t depends on $\{Y_t\}_{t\geq 1}$ (and t) only through Y_t. The process X is sometimes called the *hidden Markov process* (HMP) and the pair (Y, X) is referred to as a *hidden Markov model* (HMM). The name is motivated by the assumption that the process Y (sometimes called a *regime*) is generally non-observable. The conditional distribution of X_1 given $Y_1 = s$ is called an *emission distribution*, written as P_s, $s \in S$. We shall assume that the emission distributions are defined on a measurable space $(\mathcal{X}, \mathcal{B})$, where $\mathcal{X}$ is usually $\mathbb{R}^d$ and $\mathcal{B}$ is the corresponding Borel σ-algebra. Without loss of generality, we assume that the measures P_s have densities f_s with respect to some reference measure λ, such as the counting or Lebesgue measure.

Given a set $\mathcal{A}$, integers m and n, $m < n$, and a sequence $a_1, a_2, \ldots \in \mathcal{A}^\infty$, we write a_m^n for the subsequence $(a_m, \ldots, a_n)$. When $m = 1$, it will be often suppressed. Thus, $x^T := (x_1, \ldots, x_T)$ and $y^T := (y_1, \ldots, y_T)$ stand for the fixed observed and unobserved realizations, respectively, of the HMM $(X_t, Y_t)_{t\geq 1}$ up to time $T \geq 1$. Any sequence $s^T \in S^T$ is called a *path*. This parallel notation (that is, s^T in addition to y^T) is necessitated largely by our forthcoming discussion of various loss functions, which do require two arguments. We shall denote the joint probability density of (x^T, y^T) by $p(x^T, y^T)$, that is,

$$p(x^T, y^T) := \mathbf{P}(Y^T = y^T) \prod_{t=1}^{T} f_{y_t}(x_t).$$

To make mathematical expressions more compact, we overload the notation when this causes no ambiguity. Thus, $p(s^T)$ stands for the probability mass function $\mathbf{P}(Y^T = s^T)$ of path s^T, and $p(x^T)$ stands for the (unconditional) probability density function $\sum_{s^T\in S^T} p(x^T, s^T)$ of the observed data x^T. Furthermore, we write $p_t(s)$ and $p_t(s \mid x^T)$ for $\mathbf{P}(Y_t = s)$ and $\mathbf{P}\left(Y_t = s \mid X^T = x^T\right)$, respectively. It is standard (see Bishop, 2006, Chap. 13; Ephraim and Merhav, 2002; Cappé et al., 2005) in this context to define the so-called *forward* and *backward* variables

$$\alpha_t(s) := p(x^t \mid Y_t = s)P(Y_t = s), \quad \beta_t(s) := \begin{cases} 1, & \text{if } t = T \\ p(x_{t+1}^T \mid Y_t = s), & \text{if } t < T \end{cases}, \tag{1}$$

where $p(x^t \mid Y_t = s)$ and $p(x_{t+1}^T \mid Y_t = s)$ are the conditional densities of the data segments x^t and x_{t+1}^T, respectively, given $Y_t = s$.

1.2 Path Estimation

Our focus here is estimation of the hidden path y^T. This task can also be viewed as *segmentation* of the data sequence into regions with distinct class labels (Lember et al., 2011). Treating y^T as missing data (Rabiner, 1989), or parameters, a classical and by far the most popular solution to this task is to maximize $p(x^T, s^T)$ in $s^T \in S^T$. Often, especially

in the digital communication literature (Lin and Costello Jr., 1983; Brushe et al., 1998), $p(x^T, s^T)$ is called the *likelihood function* which might become potentially problematic in the presence of any genuine model parameters. Such "maximum likelihood" paths are also called *Viterbi paths* or *Viterbi alignments* after the Viterbi algorithm (Viterbi, 1967; Rabiner, 1989) commonly used for their computation. If $p(s^T)$ is thought of as the prior distribution of Y^T, then the Viterbi path also maximizes $p(s^T \mid x^T) := \mathbf{P}(Y^T = s^T \mid X^T = x^T)$, the probability mass function of the posterior distribution of Y^T, hence the term '*maximum a posteriori (MAP) path*'.

In spite of its computational attractiveness, inference based on the Viterbi paths may be unsatisfactory for a number of reasons, including its sub-optimality with regard to the number of correctly estimated states y_t. Also, using the language of information theory, there is no reason to expect a Viterbi path to be typical (Lember and Koloydenko, 2010). Indeed, "there might be many similar paths through the model with probabilities that add up to a higher probability than the single most probable path" (Käll et al., 2005). The fact that a MAP estimate need not be representative of the posterior distribution has also been recently discussed in a more general context by Carvalho and Lawrence (2008). Atypicality of Viterbi paths particularly concerns situations when estimation of y^T is combined with inference about model parameters, such as the transition probabilities p_{ij} (Lember and Koloydenko, 2010). Even when estimating, say, the probability of heads from independent tosses of a biased coin, we naturally hope to observe a typical realization and not the constant one of maximum probability.

An alternative and very natural way to estimate y^T is by maximizing the posterior probability $p_t(s \mid x^T)$ of each individual hidden state Y_t, $1 \le t \le T$ (Bahl et al., 1974). We refer to the corresponding estimator as *pointwise maximum a posteriori (PMAP)*. PMAP is well-known to maximize the expected number of correctly estimated states (Section 2), hence the characterization '*optimal accuracy*' (Holmes and Durbin, 1998). In statistics, especially spatial statistics and image analysis, this type of estimation is known as *Marginal Posterior Mode* (Winkler, 2003) or *Maximum Posterior Marginals* (Rue, 1995) (MPM) estimation. This is also known as the *posterior decoding* (PD) in computational biology (Brejová et al., 2007b) and machine translation (Ganchev et al., 2008), and has been reported to be particularly successful in pairwise sequence alignment (Holmes and Durbin, 1998) and when more than one path has its posterior probability as "high" or nearly as "high" as that of the Viterbi path (Eddy, 2004). In the wider context of biological applications of discrete high-dimensional probability models, this has also been called *consensus* estimation, and in the absence of constraints, *centroid* estimation (Carvalho and Lawrence, 2008). In communications applications of HMMs, largely influenced by the BCJR algorithm (Bahl et al., 1974), the terms '*optimal symbol-by-symbol detection*' (Hayes et al., 1982), '*symbol-by-symbol MAP estimation*' (Robertson et al., 1995), and '*MAP state estimation*' (Brushe et al., 1998) have been used for this. Remarkably, even before observing the data, optimal accuracy (that is, based on the prior instead of the posterior distribution) decoding can still be more accurate than the Viterbi decoding (Subsection 5.4).

1.2.1 How Different are PMAP and MAP Inferences and How Much Room is in between the Two?

This is a natural question in both practice and theory, especially for anyone interested in improving performance of applications based on these methods while maintaining their computational attractiveness.

A not so uncommon misconception that the difference between PMAP and Viterbi inferences is negligible may in part be explained by the concluding remark made by Bahl et al. (1974) in the special context of linear codes: "Even though Viterbi decoding is not optimal in the sense of bit error rate, in most applications of interest the performance of both [PMAP and Viterbi] algorithms would be effectively identical." This conclusion may in turn be explained by the dominance of binary chains in the telecommunication applications, and the binary state space indeed leaves too little room for the two inferences to differ. However, as HMMs with larger state spaces gained more prominence, it became clear that appreciable differences between the PMAP and Viterbi inferences do occur (see, for example, Ganchev et al., 2008). In fact, already two decades after Bahl et al. (1974), Brushe et al. (1998) contemplated hybridization of the PMAP and Viterbi decoders, writing "Indeed, there may be applications where a delicate performance dependence exists between [the Viterbi and PMAP] estimates. In such cases, the use of a hybrid scheme ... may result in performance gains." We return to their idea later in this paper.

Although interesting comparisons of the PMAP and Viterbi decoders on special tasks (e.g., Ganchev et al., 2008), have been recently reported, we are not aware of any systematic general studies of the two decoders that would exploit such comparisons in order to design new interesting hybrid schemes. Soon after the first version of this article was posted on `arXiv`, however, Yau and Holmes (2010) reported similar interests in this subject, supported by real and simulated examples. Of course, it has long been well-known (Rabiner, 1989) that despite being optimal in the sense of maximizing the expected number of correctly estimated states, a PMAP path can at the same time have very low, possibly zero, probability. Thus, on the logarithmic scale, the difference in path probabilities between the PMAP and Viterbi decoders can easily be *infinite*. In Section 5, we give a real data example with only six hidden states to show that besides the infinite difference in the log-probabilities, the two decoders can differ significantly (by more than 13%) in accuracy. This could have been expected if the data were indeed generated by an HMM and if that same HMM were used for decoding. However, when the model is misspecified, which is very common in practice, empirical performance measures, such as the symbol-by-symbol error rate, are generally biased as estimators of corresponding model based expected performance measures. In particular, in such situations there is no guarantee that the PMAP decoding is empirically more accurate than MAP. Although these points are fairly straightforward, we felt, especially during the reviewing process, that some readers might still appreciate a concrete illustration, which we give in Section 5. Other readers can simply glance over Section 5 without interrupting the overall flow of the manuscript.

It is actually not difficult to constrain the PMAP decoder to *admissible* paths (Subsection 2.2.1), where admissibility is defined relative to the posterior distribution. Specifically, given x^T, a path y^T is called *admissible if its posterior probability* $p(y^T \mid x^T)$ *is defined and positive*, that is, if $p(x^T, y^T) > 0$. We then point out that constraining the PMAP decoder

to the paths of positive prior probability, as already done by others (see more below), is not sufficient (albeit necessary) for admissibility of the PMAP paths. Note that in a slightly more general form allowing for state aggregation, Käll et al. (2005) do exactly this, that is, force PMAP paths to have positive prior probability, referring to the result as "a possible path through the model". Thus, Käll et al. (2005) appear to ignore that having a positive prior probability is *not sufficient in general for a PMAP path to be "a possible path through the model"*, unless, of course, "the model" is to be understood as the hidden Markov chain only and not the whole HMM. We will refer to the PMAP decoder constrained to the admissible paths as the *admissibly constrained PMAP*, or, simply *constrained PMAP*. This also details and clarifies our earlier discussion of admissibility (Lember et al., 2011, Section 2), which, like Rabiner (1989); Käll et al. (2005), also ignored the distinction between *a priori* and *a posteriori* modes of admissibility.

A variation on the same idea of making PMAP paths admissible has been applied for prediction of membrane proteins, giving rise to the *posterior Viterbi decoding (PVD)* (Fariselli et al., 2005). PVD, however, maximizes the product $\prod_{t=1}^{T} p_t(s_t \mid x^T)$ (Fariselli et al., 2005) (and also Equation 9 below) and not the sum $\sum_{t=1}^{T} p_t(s_t \mid x^T)$, whereas the two criteria are *no longer equivalent in the presence of path constraints* (Subsection 2.2.1). While acknowledging this latter distinction between their decoder and PVD and not distinguishing between the prior and posterior modes of admissibility, Käll et al. (2005) appear to be unaware of the other distinction between their decoder and PVD: PVD paths are guaranteed to be of not only positive prior probability but also of positive posterior probability, that is, admissible (in our sense of the term). Holmes and Durbin (1998) proposed a PMAP decoder to compute optimal pairwise sequence alignments. Holmes and Durbin (1998) used the term "legitimate alignment", which suggests admissibility, but the description of their algorithm (Holmes and Durbin, 1998, Section 3.8) appears to be insufficiently detailed to verify if the output is guaranteed to be admissible, or only of positive prior probability, or, if inadmissible solutions are altogether an issue in that context.

Our own experiments (Section 5) show that both PVD and constrained PMAP decoder can return paths of very low (posterior) probabilities. Moreover, in many applications, for example, gene identification and protein secondary structure prediction, the pointwise (e.g., nucleotide level) error rate is not necessarily the main measure of accuracy (see also Subsection 1.2.2 below), hence the constrained PMAP need not be an ultimate answer in that respect either. Together with the above problem of atypicality of MAP paths, this has been addressed by moving from single path inference towards *envelopes* (Holmes and Durbin, 1998). Thus, for example, in computational biology a common approach would be to aggregate individual states into a smaller number of semantic labels (e.g., codon, intron, intergenic). In effect, this would realize the notion of path similarity by mapping many "similar" state paths to a single label path, or *annotation* (Krogh, 1997; Käll et al., 2005; Fariselli et al., 2005; Brejová et al., 2007b). However, since this mapping would usually be many-to-one (what Brejová et al., 2007a refer to as the "multiple path problem"), the annotation of the Viterbi path would generally be inferior to the optimal (in the MAP sense) annotation. On the other hand, to compute the MAP annotation in many practically important HMMs can be NP-hard (Brejová et al., 2007a) (which is not surprizing given that the coarsened hidden chain on the set of labels is generally no longer Markov). Unlike the Viterbi/MAP decoder, the PMAP decoder, owing it to its symbol-by-symbol

nature, handles annotations as easily as it does state paths, including the enforcement of admissibility. Interpreting admissibility relative to the prior distribution, this was shown by Käll et al. (2005), and this paper extends their result to admissible (that is, of positive posterior probability) paths and indicates further extensions (Section 8).

A number of alternative heuristic approaches are also known in computational biology, but none appears to be fully satisfactory (Brejová et al., 2007b). Overall, although the original Viterbi decoder has still been the most popular paradigm in many applications, and in computational biology in particular, alternative approaches have often demonstrated significantly better performance, for example, in predicting various biological features. For example, Krogh (1997) suggested the *1-best* algorithm for optimal labeling. More recently, Fariselli et al. (2005) have demonstrated PVD to be superior to the 1-best algorithm, and, not surprisingly, to the Viterbi and PMAP decoders, on tasks of predicting membrane proteins.

Thus, a starting point of this contribution was that restricting the PMAP decoder to admissible paths is but one of *numerous ways to combine the strong points of the MAP and PMAP path estimators*. Indeed, the popular seminal tutorial (Rabiner, 1989) briefly mentions maximization of the expected number of correctly decoded (overlapping) blocks of length two or three, rather than single states as a sensible remedy against vanishing probabilities (albeit leaving it unclear if prior or posterior probability was meant). With $k \geq 1$ and $\widehat{y^T}(k)$ being the block length and corresponding path estimate, respectively, this approach yields Viterbi inference as k increases to T (with $\widehat{y^T}(1)$ corresponding to PMAP). Therefore, this could be interpreted as discrete interpolation between the PMAP and Viterbi inferences. Intuitively, following Rabiner's logic, one might also expect $p(x^T, \widehat{y^T}(k))$ to increase with k. However, *this is not true* and it is possible for the decoder with $k = 2$ to produce an inadmissible (with the prior probability being also zero) path $\widehat{y^T}(2)$ while the PMAP path is admissible: $p(x^T, \widehat{y^T}(2)) = 0 = p(\widehat{y^T}(2)) < p(x^T, \widehat{y^T}(1))$. *We are not aware of this observation being previously made in the literature.* Moreover, our experiments in Section 5 show that this situation is far from being uncommon.

On a related note, concerned with the same deficiencies of the MAP and PMAP inferences, Yau and Holmes (2010) have most recently also used the decision-theoretic framework to allow for full asymmetry in the otherwise symmetric pairwise loss (Equation 30 below with $k = 2$) that underpins the $\widehat{y^T}(2)$ inference. This is no doubt a very natural extension to provide to the end user, and (partially) asymmetric pairwise losses had indeed been incorporated in a prominent web-server in the context of RNA secondary structure prediction (Sato et al., 2009).

Despite the possibility of $\widehat{y^T}(2)$ or its asymmetric siblings to be inadmissible, we find the idea of interpolation between the PMAP and Viterbi inferences very interesting. Besides Yau and Holmes (2010) acknowledging the need for intermediate modes of inference, to the best of our knowledge, the only published work that explicitly proposed such an interpolation is that of Brushe et al. (1998). However, the approach of Brushe et al. (1998) is algorithmic, which makes it difficult to interpret its paths in general and analyze their properties (e.g., asymptotic behavior in particular). More importantly, Brushe et al. (1998) claim that the family of their interpolating decoders will work in practice, which, as we explain in detail in Section 6, need not be true apart from trivial situations. Despite these

and other deficiencies of their approach, it raises some interesting questions and inspires interesting modifications, which we also discuss in Section 6. It had not been our original intention to dwell on the algorithmic approach in this manuscript as this approach is peripheral to the present theme of the risk-based approach. However, encouraged by some of the reviewers and taking into account their queries on and interest in that particular discussion, we have now made that discussion into a full section (Section 6), which might, however, appear somewhat hypertrophied to some readers.

1.2.2 Further Motivation

One other motivation for considering new decoders is that unlike the error rate or path probability, analytic optimization of other performance measures (e.g., Matthew's correlation Aydin et al., 2006, Q_2, Q_{ok}, SOV Fariselli et al., 2005, etc.) used in practice is difficult if at all possible. Having a large family of computationally efficient decoders, such as the new generalized hybrid decoders, and using some training data, one can select empirically a member from the family that optimizes the performance measure of interest. More generally, it seems advantageous for applications to be aware of the new choices of decoders and their properties.

Also, depending on the application, the emphasis sometimes shifts from purely automatic decoding with hard decisions to data exploration. Indeed, some performance measures may be hard to formalize and subsequently hard to compute. For example, an estimated path can be deemed correct if it is only structurally identical to the true path, say, conforming to the description "a long run of 1's followed by a short run of 2's followed by a long run of alternating 2's and 3's". It is then particularly valuable to gain insights into the topology of the state space in the sense of identifying compartments of high concentration of the posterior distribution. The significance of identifying clusters (of similar sequences) of high (total) posterior probability in high-dimensional discrete spaces has been recently discussed by Carvalho and Lawrence (2008), and a thorough discussion of the advantages of topological and geometric approaches to analysis of complex data in general has more recently been given by Carlsson (2009). Thus, it may be beneficial to output a family of related decodings instead of one or several ("N best") decodings that are optimal relative to a single criterion such as MAP. For instance, by slowly varying the optimization criterion (e.g., decreasing the penalty for false discovery of rare states or transitions), saliency of detections of interesting features can be assessed and a better understanding of a neighborhood of solutions can be gained (e.g., discerning between an "archipelago" and a "continent"), all without having to compute, or even define explicitly, a path similarity measure (such as those based on, for example, BLAST scores Altschul et al., 1990). At the same time, by varying the optimization criteria more aggressively, alternative structures might be encountered coming from neighborhoods of remote (say, in the Hamming distance sense) local maxima of the posterior distribution. Viewed within this context, this relatively inexpensive type of "neighborhood" inference might become alternative or complementary to the direct sampling (from the posterior distribution); see also Section 5 and Section 8.

1.3 Further Notation and Organization of the Rest of the Paper

In this paper, we consider the path inference problem in the more general framework of statistical learning. Namely, we consider sequence *classifier* mappings

$$g : \mathcal{X}^T \to S^T, \quad T = 1, 2, \ldots,$$

and optimality criteria for their selection. When all g's are obtained using the same decoding principle, or optimality criterion, regardless of T, we refer to them collectively as a *classification method*, or simply, *decoder*. This will be the case in this paper, and therefore we simplify the notation by writing $g(x^T)$ instead of $g(x^T; T)$ or the like. In Section 2, criteria for optimality of g are naturally formulated in terms of risk minimization whereby $R(s^T \mid x^T)$, the *risk of* of outputting path s^T, derives from a suitable *loss function*. A Bayes decoder, that is one that minimizes $R(g(x^T) \mid x^T)$ over all possible g, will be denoted by v with a suitable reference to the risk R. In Section 3, we consider families of risk functions which naturally generalize those corresponding to the Viterbi and PMAP solutions (Subsection 2.1). There we will need the full two argument notation $v(x^T; \cdot)$ using the second argument to single out an individual member of such a family. Furthermore, as shown in Section 4, these risk functions define a family of path decoders $v(x^T; k)$ parameterized by an integer k with $k = 1$ and $k \to \infty$ corresponding to the PMAP and Viterbi cases, respectively (Theorem 6). A continuous mapping via $k = 1/(1-\alpha)$, $0 \leq \alpha \leq 1$ compactifies this parameterization and further enriches the solution space by including fractional k. It is then discussed how the new family of decoders can be embedded into yet a wider class with a principled criterion of optimality. We also compare the new family of decoders with the Rabiner k-block approach. Any decoder would only be of theoretical interest if it could not be efficiently computed. In Section 3, we show that all of the newly defined decoders can be implemented efficiently as a dynamic programming algorithm in the usual forward-backward manner with essentially the same (computational as well as memory) complexity as the PMAP or Viterbi decoders (Theorem 4). Recent advances in the asymptotic theory of some of the main decoders and risks presented in this paper are reviewed in Section 7 together with sketches of how these may be relevant in practice. Various further extensions are discussed in the concluding Section 8.

1.4 Contributions of the Paper

We review HMM-based decoding within the sound framework of statistical decision theory, and do so notably more broadly than has been done before, for example, in the prominent work of Carvalho and Lawrence (2008). We also investigate thoroughly previous work on combining the desirable properties of the two most common decoders, that is the Viterbi and optimal accuracy decoders. In doing so, we discover several relevant claims and suggestions to be unjustified, misleading, or plainly incorrect. We explain in detail those deficiencies, giving relevant counterexamples, and show how they can be resolved. Some such resolutions are naturally left within the native frameworks of the originals, whereas others are more naturally given within the general risk-based framework. All of the resulting decoders are shown to be easily implementable within the usual forward-backward computational frameworks of the optimal accuracy and Viterbi decoders. We argue that the richness, flexibility,

and analytic interpretation of the resulting families of decoders offer new possibilities for applications and invite further theoretical analysis. Specifically, this paper

1) clarifies the definition of admissibility of hidden paths and shows that, when constrained to the paths of positive prior probability, the optimal accuracy decoding can still return inadmissible paths;

2) shows that the suggestion of Rabiner (1989) to maximize the expected rate of correctly recognized blocks can lead to inadmissible paths for blocks of size two, and therefore can be misleading;

3) proposes suitable risk functions to "repair" the above suggestion, and subsequently designs new families of computationally efficient decoders, providing an experimental illustration;

4) unifies virtually all of the key decoders within the same risk-based framework;

5) analyzes the relationships between the risks achieved by the different decoders, yielding a general result on convex decomposition of the key risk functionals for Markov chains;

6) analyzes the related earlier work of Brushe et al. (1998), and in particular:

(a) explains how the idea of hybridization of the Viterbi and optimal accuracy decoders proposed in the above work can fail when the Viterbi path is not unique;

(b) establishes that the claims made in the same work regarding the implementation of their algorithm to hybridize the Viterbi and optimal accuracy decoders are incorrect;

(c) shows how the corresponding forward and backward variables given in the same work can be scaled to produce an operational decoding algorithm;

(d) shows that the resulting decoders are different from the original hybrid decoders of Brushe et al. (1998);

(e) proposes an immediately operational algorithm to hybridize the Viterbi and optimal accuracy decoders (at least when the Viterbi path is unique), which is based on the more common power-transform, and which also allows for extrapolations "beyond" the optimal accuracy decoder;

7) indicates a number of further extensions of the new families of decoders.

At the same time, a thorough performance evaluation, including asymmetric variants of the main loss functions, and using several applications with their own performance measures, is outside the scope of this paper (Section 8).

2. Risk-Based Path Inference

Given a sequence of observations x^T with $p(x^T) > 0$, we view the (posterior) *risk* as a function

$$R(\cdot \mid x^T): \quad S^T \mapsto [0, \infty].$$

Naturally, we seek a state sequence with minimum risk: $v(x^T) := \arg\min_{s^T \in S^T} R\big(s^T \mid x^T\big)$. In the *statistical decision and pattern recognition theories*, the classifier v is known as the *Bayes classifier* (relative to risk R). Within the same framework, the risk is often specified via a *loss-function*

$$L : S^T \times S^T \to [0, \infty],$$

interpreting $L(s^T, y^T)$ as the loss incurred by the decision to predict s^T when the actual state sequence was y^T. Therefore, for any state sequence $s^T \in S^T$, the risk is given by

$$R(s^T \mid x^T) := E[L(s^T, Y^T) \mid X^T = x^T] = \sum_{y^T \in S^T} L(s^T, y^T) p(y^T \mid x^T).$$

2.1 Standard Path Inferences Re-Examined

The most popular loss function is the so-called *symmetrical* or *zero-one* loss L_∞ defined as follows:

$$L_\infty(s^T, y^T) = \begin{cases} 1, & \text{if } s^T \neq y^T; \\ 0, & \text{if } s^T = y^T. \end{cases}$$

We shall denote the corresponding risk by R_∞. With this loss, clearly

$$R_\infty(s^T \mid x^T) = \mathbf{P}(Y^T \neq s^T \mid X^T = x^T) = 1 - p(s^T \mid x^T), \tag{2}$$

thus $R_\infty(\cdot \mid x^T)$ is minimized by a Viterbi path, that is, a sequence of maximum posterior probability. Let $v(\cdot; \infty)$ stand for the corresponding classifier, that is

$$v(x^T; \infty) := \arg \max_{s^T \in S^T} p(s^T \mid x^T),$$

with a suitable tie-breaking rule.

Note that Viterbi paths also minimize the following risk

$$\bar{R}_\infty(s^T \mid x^T) := -\frac{1}{T} \log p(s^T \mid x^T). \tag{3}$$

It can actually be advantageous to use the logarithmic risk (3) since, as we shall see later, this leads to various natural generalizations (Sections 3 and 4).

When sequences are compared pointwise, it is common to use additive loss functions of the form

$$L_1(s^T, y^T) = \frac{1}{T} \sum_{t=1}^{T} l(s_t, y_t), \tag{4}$$

where $l(s, y) \geq 0$ is the loss associated with classifying y as s. Typically, for every state s, $l(s, s) = 0$. It is not hard to see that, with L_1 as in (4), the corresponding risk can be represented as follows

$$R_1(s^T \mid x^T) = \frac{1}{T} \sum_{t=1}^{T} \rho_t(s_t \mid x^T),$$

where $\rho_t(s \mid x^T) = \sum_{y \in S} l(s, y) p_t(y \mid x^T)$. Most commonly, l is again symmetrical, or zero-one, that is $l(s, y) = \mathbb{I}_{\{s \neq y\}}$, where $\mathbb{I}_A$ stands for the indicator function of set A. In

this case, L_1 is naturally related to the *Hamming distance* (Carvalho and Lawrence, 2008). Then also $\rho_t(s_t \mid x^T) = 1 - p_t(s_t \mid x^T)$ so that the corresponding risk is

$$R_1(s^T \mid x^T) = 1 - \frac{1}{T}\sum_{t=1}^{T} p_t(s_t \mid x^T). \tag{5}$$

Let $v(\cdot;1)$ stand for the Bayes classifier relative to this R_1-risk. It is easy to see from the above definition of R_1, that $v(\cdot;1)$ delivers PMAP paths, which minimize the expected number of misclassification errors. In addition to maximizing $\sum_{t=1}^{T} p_t(s_t \mid x^T)$, $v(\cdot;1)$ also maximizes $\prod_{t=1}^{T} p_t(s_t \mid x^T)$, and therefore minimizes the following risk

$$\bar{R}_1(s^T \mid x^T) := -\frac{1}{T}\sum_{t=1}^{T} \log p_t(s_t \mid x^T). \tag{6}$$

2.2 Generalizations

Next, we begin to consider various generalizations of the the standard path inferences.

2.2.1 Admissible PMAP and Posterior Viterbi Decoders

Recall (Subsection 1.2.1) that PMAP paths can be *inadmissible*. According to our definition of admissibility (Subsection 1.2.1), a path is inadmissible if it is of zero posterior probability. Although Rabiner (1989) gives no explicit definition of admissibility, or *validity*, he refers to forbidden transitions, that is, of zero prior probability (which, of course, also implies zero posterior probability) as an example of how a path can be "not valid"; the possibility of a path to have a positive prior probability but zero posterior probability is not discussed there. As far as we are aware, Käll et al. (2005) were the first to formally write down an amended PMAP optimization problem to guarantee path validity, or admissibility. However, they too do not state explicitly if "a possible path through the model" means for them positivity only of the prior probability or also of the posterior probability. If "the model" is to be understood as the HMM in its entirety, then this would require positivity of the posterior probability. However, the optimization presented by Käll et al. (2005) does not guarantee positivity of the posterior probability, that is, it only guarantees positivity of the prior probability. Perhaps, it does not happen very often in practice that the PMAP decoder constrained to return *a priori* possible paths returns an inadmissible path (it does not happen in our own experiments in Section 5 as all of our emission probabilities are non-zero on the entire emission alphabet). However, as the example in Appendix A shows, this is indeed possible.

Thus, to enforce admissibility properly, R_1-risk needs to be minimized over the admissible paths (R_1 minimization over the paths of positive prior probability is revisited in Subsection 2.2.2 below):

$$\min_{s^T : p(s^T \mid x^T) > 0} R_1(s^T \mid x^T) \quad \Leftrightarrow \quad \max_{s^T : p(s^T \mid x^T) > 0} \sum_{t=1}^{T} p_t(s_t \mid x^T). \tag{7}$$

Assuming that $p_t(s \mid x^T)$, $1 \le t \le T$, $s \in S$, have been precomputed (e.g., by the classical forward-backward recursion Rabiner, 1989), a solution to (7) can be easily found by a

Viterbi-like recursion (8)

$$\begin{aligned} \delta_1(j) &:= p_1(j \mid x^T), \ \ \forall\, j \in S, \\ \delta_{t+1}(j) &:= \max_i \left(\delta_t(i) + \log r_t(i,j)\right) + p_{t+1}(j \mid x^T) \text{ for } t = 1, 2, \ldots, T-1, \text{ and } \forall j \in S, \end{aligned} \tag{8}$$

where $r_t(i,j) := \mathbb{I}_{\{p_{ij} f_j(x_{t+1}) > 0\}}$ (recall that $p_{ij} = \mathbf{P}(Y_{t+1} = j \mid Y_t = i)$ and f_j is the density of the conditional probability distribution of X_{t+1} conditioned on $Y_{t+1} = j$). *To the best of our knowledge this has not been stated in the literature before.* We will refer to this decoder as the *Constrained PMAP* decoder.

Next note that in the presence of path constraints, minimization of the R_1-risk (5) is *no longer equivalent* to minimization of the $\bar{R}_1$-risk (6). In particular, the problem (7) is not equivalent to the following problem

$$\min_{s^T : p(s^T \mid x^T) > 0} \bar{R}_1(s^T \mid x^T) \quad \Leftrightarrow \quad \max_{s^T : p(s^T \mid x^T) > 0} \sum_{t=1}^{T} \log p_t(s_t \mid x^T). \tag{9}$$

It is also important to note that the problem (9) above *is equivalent* to what has been termed the *posterior-Viterbi decoding*, or *PVD* (Fariselli et al., 2005):

$$\min_{s^T : p(s^T) > 0} \bar{R}_1(s^T \mid x^T) \quad \Leftrightarrow \quad \max_{s^T : p(s^T) > 0} \sum_{t=1}^{T} \log p_t(s_t \mid x^T),$$

that is, unlike in the case of $R_1(s^T \mid x^T)$ minimization, minimization of $\bar{R}_1(s^T \mid x^T)$ over the paths of positive prior probability is indeed sufficient to produce admissible paths.

A solution to (9) can be computed by a related recursion given in (10) below

$$\begin{aligned} \delta_1(j) &:= \log p_1(j \mid x^T), \ \forall j \in S, \\ \delta_{t+1}(j) &:= \max_i \left(\delta_t(i) + \log r_{ij}\right) + \log p_{t+1}(j \mid x^T), \text{ for } t = 1, 2, \ldots, T-1, \ \forall j \in S, \end{aligned} \tag{10}$$

where $r_{ij} := \mathbb{I}_{\{p_{ij} > 0\}}$ (which for inhomogeneous chains will depend on t).

2.2.2 Beyond PVD and *A priori* Admissible PMAP

Although admissible minimizers of R_1 and $\bar{R}_1$ risk are by definition of positive probability, this probability can still be very small. Indeed, in the above recursions, the weight r_{ij} is 1 even when p_{ij} is very small. We next replace r_{ij} by the true transition probability p_{ij} in minimizing the $\bar{R}_1$-risk (that is maximization of $\prod_{t=1}^{T} p_t(s_t \mid x^T)$). Then the solutions remain admissible and also tend to maximize the prior path probability. To bring the newly obtained optimization problem to a more elegant form (11), we pretend that $\delta_1(j)$ in (10) above was defined as $\delta_1(j) := \log p_1(j \mid x^T) + \log \mathbb{I}_{\{\pi_j > 0\}}$ (which indeed does not change the results of the recursion (10)) and replace the last term by $\log \pi_j$.

Thus, with the above replacements, the recursion (10) now solves the following *seemingly unconstrained* optimization problem (see Theorem 4)

$$\max_{s^T} \Big[\sum_{t=1}^{T} \log p_t(s_t \mid x^T) + \log p(s^T) \Big] \quad \Leftrightarrow \quad \min_{s^T} \Big[\bar{R}_1(s^T \mid x^T) + h(s^T) \Big], \tag{11}$$

where the penalty term

$$h(s^T) \quad = -\tfrac{1}{T}\log p(s^T) \quad =: \bar{R}_\infty(s^T) \tag{12}$$

is the logarithmic risk based on the prior distribution,[1] which does not involve the observed data.

The thereby modified recursions immediately generalize as follows:

$$\begin{aligned}\delta_1(j) &:= \log p_1(j \mid x^T) + C\log\pi_j,\ \forall j \in S,\\ \delta_{t+1}(j) &:= \max_i \big(\delta_t(i) + C\log p_{ij}\big) + \log p_{t+1}(j \mid x^T) \text{ for } t = 1,2,\ldots,T-1,\ \forall j \in S,\end{aligned}$$

solving

$$\min_{s^T}\Big[\bar{R}_1(s^T \mid x^T) + Ch(s^T)\Big], \tag{13}$$

where $C > 0$ is a trade-off constant, which can also be viewed as a regularization parameter. Indeed, Proposition 2 below states that $C > 0$ implies admissibility of solutions to (13). In particular, PVD, that is the problem solved by the original recursion (10), can now be recovered by taking C sufficiently small. (Alternatively, the PVD problem can also be formally written in the form (13) with $C = \infty$ and $h(s^T)$ given, for example, by $\mathbb{I}_{\{p(s^T)=0\}}$.)

What if the actual probabilities p_{ij} (π_j) were also used in the optimal accuracy/PMAP decoding? To motivate this, we re-consider the optimal accuracy/PMAP decoding imposing the positivity constraint not on the posterior but on the prior path probability:

$$\min_{s^T:p(s^T)>0} R_1(s^T \mid x^T) \quad \Leftrightarrow \quad \max_{s^T:p(s^T)>0}\sum_{t=1}^T p_t(s_t \mid x^T). \tag{14}$$

Solution to (14) can be easily found by yet another Viterbi-like recursion given in (15) below

$$\begin{aligned}\delta_1(j) &:= p_1(j \mid x^T),\ \ \forall\, j \in S, \qquad (15)\\ \delta_{t+1}(j) &:= \max_i\,(\delta_t(i) + \log r_{ij}) + p_{t+1}(j \mid x^T) \text{ for } t = 1,2,\ldots,T-1, \text{ and } \forall j \in S,\end{aligned}$$

which is the same as (8) apart from the r_{ij} in place of the $r_t(i,j)$.

We again replace the indicators r_{ij} by the actual probabilities p_{ij}. We once more pretend that $\delta_1(j)$ in (15) above was defined, this time, as $\delta_1(j) := p_1(j \mid x^T) + \log\mathbb{I}_{\{\pi_j>0\}}$. Replacing the last term by $\log\pi_j$ yields the following problem:

$$\max_{s^T}\Big[\sum_{t=1}^T p_t(s_t \mid x_t) + \log p(s^T)\Big] \quad \Leftrightarrow \quad \min_{s^T}\Big[R_1(s^T \mid x^T) + \bar{R}_\infty(s^T)\Big]. \tag{16}$$

A more general problem can be written in the form

$$\min_{s^T}\Big[R_1(s^T \mid x^T) + Ch(s^T)\Big], \tag{17}$$

1. More generally, the same type of risk (e.g., $\bar{R}_\infty$) can be based on the posterior ($p(s^T \mid x^T)$), joint ($p(s^T, x^T)$) or prior ($p(s^T)$) distribution. Compromising between notational accuracy on the one hand and notational simplicity and consistency on the other hand, throughout the paper we disambiguate these cases solely by the argument.

where h is some penalty function (independent of the data x^T). Thus, the problem (14) of optimal accuracy/PMAP decoding over the paths of positive prior probability is obtained by taking C sufficiently small and $h(s^T) = \bar{R}_\infty(s^T)$. (Setting $C \times h(s^T) = \infty \times \mathbb{I}_{\{p(s^T)=0\}}$ also reduces the problem (17) back to (7).)

Clearly, if instead of (14) we started off with (7) ($R_1(s^T \mid x^T)$ minimization over the admissible paths), we would arrive at $\bar{R}_\infty(s^T \mid x^T)$ in place of $\bar{R}_\infty(s^T)$ in (16) above. Inclusion of $\bar{R}_\infty(s^T \mid x^T)$ more generally is treated next in Section 3.

3. Combined Risks

Motivated by the previous section, we consider the following general problem

$$\min_{s^T} \Big[C_1 \bar{R}_1(s^T \mid x^T) + C_2 \bar{R}_\infty(s^T \mid x^T) + C_3 \bar{R}_1(s^T) + C_4 \bar{R}_\infty(s^T) \Big], \tag{18}$$

where $C_i \geq 0$, $i = 1, 2, 3, 4$, $\sum_{i=1}^4 C_i > 0$.[2] This is also equivalent to

$$\min_{s^T} \Big[C_1 \bar{R}_1(s^T \mid x^T) + C_2 \bar{R}_\infty(s^T, x^T) + C_3 \bar{R}_1(s^T) + C_4 \bar{R}_\infty(s^T) \Big], \tag{19}$$

where, recalling (6), $\quad \bar{R}_1(s^T \mid x^T) = -\frac{1}{T} \sum_{t=1}^T \log p_t(s_t \mid x^T)$,

$$\begin{aligned}
\bar{R}_\infty(s^T, x^T) &:= -\frac{1}{T} \log p(x^T, s^T), \\
&= -\frac{1}{T} [\log p(s^T) + \sum_{t=1}^T \log f_{s_t}(x_t)], \\
&= -\frac{1}{T} [\log \pi_{s_1} + \sum_{t=1}^{T-1} \log p_{s_t s_{t+1}} + \sum_{t=1}^T \log f_{s_t}(x_t)],
\end{aligned}$$

recalling (3), $\quad \bar{R}_\infty(s^T \mid x^T) = -\frac{1}{T} \log p(s^T \mid x^T)$,

$$\begin{aligned}
&= \bar{R}_\infty(s^T, x^T) + \frac{1}{T} \log p(x^T), \\
\bar{R}_1(s^T) &:= -\frac{1}{T} \sum_{t=1}^T \log p_t(s_t), \qquad (20) \\
\bar{R}_\infty(s^T) &= -\frac{1}{T} \log p(s^T), \quad \text{recalling (12)}, \\
&= -\frac{1}{T} [\log \pi_{s_1} + \sum_{t=1}^{T-1} \log p_{s_t s_{t+1}}]. \qquad (21)
\end{aligned}$$

The newly introduced risk $\bar{R}_1(s^T)$ involves only the prior marginals. Note that the combination $C_1 = C_3 = C_4 = 0$ corresponds to the MAP/Viterbi decoding; the combination

2. For uniqueness of representation, one may want to additionally require $\sum_{i=1}^4 C_i = 1$.

$C_2 = C_3 = C_4 = 0$ yields the PMAP case, whereas the combinations $C_1 = C_2 = C_3 = 0$ and $C_1 = C_2 = C_4 = 0$ give the *maximum a priori* decoding and *marginal prior mode* decoding, respectively. The case $C_2 = C_3 = 0$ subsumes (13) and the case $C_1 = C_3 = 0$ is the problem

$$\min_{s^T} \left[\bar{R}_\infty(s^T \mid x^T) + C\bar{R}_\infty(s^T) \right]. \tag{22}$$

Thus, a solution to (22) is a generalization of the Viterbi decoding that allows one to suppress ($C > 0$) contribution of the data.

Remark 1 *If $C_2 > 0$, then every solution of* (18) *is admissible and the minimized risk is finite.*

No less important and perhaps a little less obvious is that $C_1, C_4 > 0$ also guarantees admissibility of the solutions, as stated in Proposition 2 below.

Proposition 2 *Let $C_1, C_4 > 0$. Then, the minimized risk* (18) *is finite and any minimizer s^T is admissible.*

Proof Without loss of generality, assume $C_2 = C_3 = 0$. Since $p(x^T) > 0$ (assumed in the beginning of Section 2), there exists some admissible path s^T. Clearly, the combined risk of this path is finite, hence so is the minimum risk. Now, suppose s^T is a minimizer of the combined risk and suppose further that s^T is inadmissible, that is $p(s^T \mid x^T) = 0$. Since the minimized risk (18) is finite, we must have $p(s^T) > 0$. Therefore, it must be that $p(x^T \mid s^T) = 0$, and therefore we must have some t, $1 \le t \le T$, such $f_{s_t}(x_t) = 0$. This would imply that any path through (t, s_t) is inadmissible, hence $p_t(s_t \mid x^T)$, the sum of the posterior probabilities of all such paths, is zero. This implies $\bar{R}_1(s^T \mid x^T) = \infty$, contradicting optimality of s^T. ■

Remark 3 *Note that for any x^T, the Posterior-Viterbi decoding (Fariselli et al., 2005) (Problem 9 above) can be obtained by setting $C_3 = C_4 = 0$ and taking C_2 sufficiently small, that is, $0 < C_2 \ll C_1$. Also, PVD can be obtained almost surely by setting $C_2 = C_3 = 0$ and taking C_4 sufficiently small, that is, $0 < C_4 \ll C_1$.*

It is fairly intuitive that PVD can be realized as solutions to (18), but we nonetheless prove this formally in Appendix B.

If the smoothing probabilities $p_t(s \mid x^T)$, $t = 1, \ldots, T$ and $s \in S$, have been already computed, a solution to (18) can be found also by a standard dynamic programming algorithm. Let us first introduce more notation. For every $t \in 1, \ldots, T$ and $j \in S$, let

$$\gamma_t(j) := C_1 \log p_t(j \mid x^T) + C_2 \log f_j(x_t) + C_3 \log p_t(j).$$

Note that the function γ_t depends on the entire data x^T. Next, let us also define the following scores

$$\begin{aligned} \delta_1(j) &:= (C_2 + C_4) \log \pi_j + \gamma_1(j), \ \forall j \in S, \\ \delta_t(j) &:= \max_i \left(\delta_{t-1}(i) + (C_2 + C_4) \log p_{ij} \right) + \gamma_t(j), \\ &\qquad \text{for } t = 2, 3, \ldots, T, \text{ and } \forall j \in S. \end{aligned} \tag{23}$$

Using the above scores $\delta_t(j)$ and a suitable tie-breaking rule, below we define the back-pointers $i_t(j)$, terminal state i_T, and the optimal path $\widehat{y^T}(i_T)$.

$$\begin{aligned} i_t(j) &:= \arg\max_{i\in S}[\delta_t(i) + (C_2 + C_4)\log p_{ij}], \quad \text{when } t = 1,\dots,T-1; \\ i_T &:= \arg\max_{i\in S}\delta_T(i); \end{aligned} \tag{24}$$

$$\widehat{y^t}(j) := \begin{cases} i_1(j), & \text{when } t = 1; \\ \big(\widehat{y^{t-1}}(i_{t-1}(j)), j\big), & \text{when } t = 2,\dots,T. \end{cases} \tag{25}$$

Thus, given x^{t+1} and the best path that ends in state j (at time $t+1$), $i_t(j)$ represents the t-th state in this path.

The following theorem formalizes the dynamic programming argument; its proof is standard and we state it below for completeness only.

Theorem 4 *Any solution to* (18) *can be represented in the form* $\widehat{y^T}(i_T)$ *provided the ties in* (24) *are broken accordingly.*

Proof With a slight abuse of notation, for every $s^t \in S^t$, let

$$U(s^t) = \sum_{u=1}^{t} \left[\gamma_u(s_u) + (C_2 + C_4)\log p_{s_{u-1}s_u}\right],$$

where $s_0 := 0$ and $p_{0s} := \pi_s$. Hence,

$$-T[C_1\bar{R}_1(s^T \mid x^T) + C_2\bar{R}_\infty(s^T, x^T) + C_3\bar{R}_1(s^T) + C_4\bar{R}_\infty(s^T)] = U(s^T)$$

and any maximizer of $U(s^T)$ is clearly a solution to (18) and (19).

Next, let $U(j) := \delta_1(j)$ for all $j \in S$, and let

$$U(s^{t+1}) = U(s^t) + (C_2 + C_4)\log p_{s_t s_{t+1}} + \gamma_{t+1}(s_{t+1}),$$

for $t = 1, 2, \dots, T-1$ and also $s^t \in S^t$. By induction on t, these yield

$$\delta_t(j) = \max_{s^t : s_t = j} U(s^t)$$

for every $t = 1, 2, \dots, T$ and for all $j \in S$. Clearly, every maximizer $\widehat{y^T}$ of $U(s^T)$ over the set S^T must end up in i_T, or, more precisely, in the set $\arg\max_{j\in S}\delta_T(j)$, allowing for non-uniqueness. Continuing to interpret arg max as a set, recursion (23) implies recursions (24) and (25), hence any maximizer $\widehat{y^T}$ can indeed be computed in the form $\widehat{y^T}(i_T)$ via the *forward* (recursion (24))-*backward* (recursion (25)) procedure. ■

Similarly to the generalized risk minimization of (18), the generalized problem of accuracy optimization (17) can also be further generalized as follows:

$$\min_{s^T}\Big[C_1 R_1(s^T \mid x^T) + C_2\bar{R}_\infty(s^T \mid x^T) + C_3 R_1(s^T) + C_4\bar{R}_\infty(s^T)\Big], \tag{26}$$

where risk

$$R_1(s^T) := \frac{1}{T}\sum_{t=1}^{T}\mathbf{P}(Y_t \neq s_t) = 1 - \frac{1}{T}\sum_{t=1}^{T} p_t(s_t) \tag{27}$$

is the error rate relative to the prior distribution. This problem can also be solved by a recursion formally identical to that in (23) except for the removed logarithms in the marginal probabilities:

$$\gamma_t(j) \quad = \quad C_1 p_t(j \mid x^T) + C_2 \log f_j(x_t) + C_3 p_t(j). \tag{28}$$

The following remarks compare this generalized Problem with the generalized Problem (18) (Remarks 1 and 3, Proposition 2).

Remark 5 *1. As in the generalized posterior-Viterbi decoding* (18)*, here $C_2 > 0$ also implies admissibility of the optimal paths.*

2. Now, $C_4 > 0$ implies that the minimized risk is finite for any x^T, but unlike in (18)*, $C_1, C_4 > 0$ is not sufficient to guarantee admissibility almost surely of the solutions to the problem* (26)*.*

3. Taking $C_3 = C_4 = 0$, the constrained PMAP problem (Käll et al., 2005) (Problem 7 above) is obtained for some C_1, C_2 such that $0 < C_2 \ll C_1$.

We refer to a decoder solving the generalized risk minimization Problem (18) as a *generalized posterior-Viterbi hybrid decoder*. Similarly, a decoder solving the generalized optimal accuracy Problem (26) is referred to as a *generalized PMAP hybrid decoder* to distinguish the product-based risk $\bar{R}_1(s^T \mid x^T)$ in the former case from the sum-based risk $R_1(s^T \mid x^T)$ in the latter case. Both the generalized families, however, naturally extend the PMAP/optimal accuracy/posterior decoder (Section 2.1).

Corollary 15 of Apendix C establishes the usual trade-off type of resuls for the solutions to Problems (18) and (26). The results on the trade-off between $\bar{R}_1$ and $\bar{R}_\infty$ risks will in particular be useful in Corollary 8 (see further below) for establishing monotonicity of the solution to Problem (18).

4. The k-Block Posterior-Viterbi Decoding

The next approach provides a surprisingly different insight into what otherwise has already been formulated as the generalized Problem (18). This, first of all, helps better understand how the generalized Problem (18) resolves the drawback of Rabiner's suggestion (introduced in the last paragraph of Subsection 1.2.1 above). Secondly, the same approach gives an elegant relationship (Theorem 6, Corollary 7) between the main types of risk, which surprisingly amounts to, as far as we know, a novel property of ordinary Markov chains (Equation 34, and Proposition 14 of the concluding Section 8).

Recall (Subsection 1.2) that Rabiner's compromise between MAP and PMAP is to maximize the expected number of correctly decoded pairs or triples of (adjacent) states. With k being the length of the overlapping block ($k = 2, 3, \ldots$) this means to minimize the

conditional risk

$$R_k(s^T \mid x^T) := 1 - \frac{1}{T-k+1} \sum_{t=1}^{T-k+1} p(s_t^{t+k-1} \mid x^T), \tag{29}$$

which derives from the following loss function:

$$L_k(s^T, y^T) := \frac{1}{T-k+1} \sum_{t=1}^{T-k+1} \mathbb{I}_{\{s_t^{t+k-1} \neq y_t^{t+k-1}\}}. \tag{30}$$

When $k = 1$ this gives the usual R_1 maximization, that is, the PMAP decoding, which is known to fault by allowing inadmissible paths. Just as in (4) with $k = 1$, we could also consider a general (possibly asymmetric) loss function $l_k(s_t^{t+k-1}, y_t^{t+k-1})$ for larger k in (30) above. Thus, for $k = 2$ this is the Markov loss function studied by Yau and Holmes (2010).

It is natural to think that minimizers of $R_k(s^T \mid x^T)$ "move" towards Viterbi paths "monotonically" as k increases to T. Indeed, when $k = T$, minimization of $R_k(s^T \mid x^T)$ (29) is equivalent to minimization of $\bar{R}_\infty(s^T \mid x^T)$ achieved by the Viterbi decoding. However, as the experiments in Section 5 below show, minimizers of (29) are not guaranteed to be admissible (even if admissibility were defined relative to the prior distribution) for $k > 1$. Also, as we already pointed out in Subsection 1.2.1, this approach does not give monotonicity, that is, allows the optimal path for $k = 2$ to have lower (prior and posterior) probabilities than those of the PMAP path (that is, $k = 1$). Another drawback of using the loss L_k (30) and its more general variants is that, unlike in the generalized PVD and PMAP hybrid decoders, *the computational complexity of Rabiner's approach grows with the block length k*. We now show how these drawbacks go away when the sum in (29) is replaced by a product, eventually arriving at a subfamily of the generalized posterior Viterbi decoders. Certainly, replacing the sum by the product alters the problem, and it does so in a way that makes the block-wise coding idea work well. Namely, the longer the block, the larger the resulting path probability, which is also now guaranteed to be positive already for $k = 2$. Moreover, this gives another interpretation of the risks $\bar{R}_1(s^T \mid x^T) + C\bar{R}_\infty(s^T \mid x^T)$ (see also Remark 3 above), the prior risks $\bar{R}_1(s^T) + C\bar{R}_\infty(s^T)$, and consequently the generalized Problem (18).

Let k be a positive integer. For the time being, let p represent any first order Markov chain on S^T, and let us define

$$\bar{U}_k(s^T) := \prod_{j=1-k}^{T-1} p\big(s_{\max(j+1,1)}^{\min(j+k,T)}\big), \quad \bar{R}_k(s^T) := -\frac{1}{T} \ln \bar{U}_k(s^T).$$

Thus

$$\bar{U}_k(s^T) = U_1^k \cdot U_2^k \cdot U_3^k,$$

where

$$\begin{aligned} U_1^k &:= p(s_1) \cdots p(s_1^{k-2}) p(s_1^{k-1}), \\ U_2^k &:= p(s_1^k) p(s_2^{k+1}) \cdots p(s_{T-k}^{T-1}) p(s_{T-k+1}^T), \\ U_3^k &:= p(s_{T-k+2}^T) p(s_{T-k+3}^T) \cdots p(s_T). \end{aligned}$$

Thus, $\bar{R}_k$ is a natural generalization of $\bar{R}_1$ (introduced first for the posterior distribution in (6)) since when $k=1$, $\bar{R}_k = \bar{R}_1$.

Theorem 6 *Let k be such that $T \geq k > 1$. Then the following recursion holds*

$$\bar{R}_k(s^T) = \bar{R}_\infty(s^T) + \bar{R}_{k-1}(s^T), \quad \forall s^T \in S^T.$$

Proof Note that

$$U_1^k = U_1^{k-1} p(s_1^{k-1}), \quad U_3^k = p(s_{T-k+2}^T) U_3^{k-1}.$$

Next, for all j such that $j+k \leq T$, the Markov property gives

$$p(s_{j+1}^{j+k}) = p(s_{j+k} \mid s_{j+k-1}) p(s_{j+1}^{j+k-1})$$

and

$$\begin{aligned}
&U_2^k p(s_{T-k+2}^T) = p(s_1^k) p(s_2^{k+1}) \cdots p(s_{T-k+1}^T) p(s_{T-k+2}^T) = \\
&p(s_k \mid s_{k-1}) p(s_1^{k-1}) p(s_{k+1} \mid s_k) p(s_2^k) \cdots p(s_T \mid s_{T-1}) p(s_{T-k+1}^{T-1}) p(s_{T-k+2}^T) = \\
&p(s_k \mid s_{k-1}) p(s_{k+1} \mid s_k) \cdots p(s_T \mid s_{T-1}) p(s_1^{k-1}) \cdots p(s_{T-k+1}^{T-1}) p(s_{T-k+2}^T) = \\
&p(s_k \mid s_{k-1}) \cdots p(s_T \mid s_{T-1}) U_2^{k-1}.
\end{aligned}$$

Hence,

$$\begin{aligned}
\bar{U}_k(s^T) &= U_1^{k-1} p(s_1^{k-1}) p(s_k \mid s_{k-1}) \cdots p(s_T \mid s_{T-1}) U_2^{k-1} U_3^{k-1}, \\
&= p(s_1^T) U_1^{k-1} U_2^{k-1} U_3^{k-1} = p(s^T) \bar{U}_{k-1}(s^T).
\end{aligned}$$

The second equality above also follows from the Markov property. Taking logarithms on both sides and dividing by $-T$ completes the proof. ■

Now, we specialize this result to our HMM context, and, thus, $p(s^T)$ and $p(s^T \mid x^T)$ are again the prior and posterior hidden path distributions.

Corollary 7 *Let k be such that $T \geq k > 1$. For all paths $s^T \in S^T$ the prior risks $\bar{R}_k$ and $\bar{R}_\infty$ satisfy* (31)*. For every $x^T \in \mathcal{X}^T$ and for all paths $s^T \in S^T$, the posterior risks $\bar{R}_k$ and $\bar{R}_\infty$ satisfy* (32)*.*

$$\bar{R}_k(s^T) = \bar{R}_\infty(s^T) + \bar{R}_{k-1}(s^T), \tag{31}$$

$$\bar{R}_k(s^T \mid x^T) = \bar{R}_\infty(s^T \mid x^T) + \bar{R}_{k-1}(s^T \mid x^T). \tag{32}$$

Proof Clearly, conditioned on the data x^T, Y^T remains a first order Markov chain (generally inhomogeneous even if it was homogeneous *a priori*). Hence, Theorem 6 applies. ■

Below, *we focus on the posterior distribution and risks, but the discussion readily extends to any first order Markov chain.*

Let $v(x^T; k)$ be a decoder that minimizes $\bar{R}_k(s^T \mid x^T)$, returning a path $\hat{y}(k)$, that is,

$$\hat{y}(k) = \arg \max_{s^T \in S^T} \bar{U}_k(s^T \mid x^T) = \arg \min_{s^T \in S^T} \bar{R}_k(s^T \mid x^T). \tag{33}$$

Corollary (8) below states how $\bar{R}_k(s^T \mid x^T)$ minimization is a special case of the generalized Problem (18). We refer to the generalized posterior-Viterbi hybrid decoders $v(x^T; k)$ as *k-block* PVD and summarize their properties in Corollary (8).

Corollary 8 *For every $x^T \in \mathcal{X}^T$, and for every $s^T \in S^T$, we have*

$$\bar{R}_k(s^T \mid x^T) = (k-1)\bar{R}_\infty(s^T \mid x^T) + \bar{R}_1(s^T \mid x^T), \qquad \forall k \text{ such that } 1 \le k \le T. \tag{34}$$

$$\hat{y}(k) \text{ is admissible}, \qquad \forall k \text{ such that } k > 1. \tag{35}$$

$$\bar{R}_\infty(\hat{y}(k) \mid x^T) \le \bar{R}_\infty(\hat{y}(k-1) \mid x^T), \qquad \forall k \text{ such that } 1 < k \le T. \tag{36}$$

$$\bar{R}_1(\hat{y}(k) \mid x^T) \ge \bar{R}_1(\hat{y}(k-1) \mid x^T), \qquad \forall k \text{ such that } 1 < k \le T. \tag{37}$$

Proof Equation (34) follows immediately from Equation (32) of Corollary 7. Admissibility of $\hat{y}(k)$ for $k > 1$ in (35) becomes obvious recalling Remark 1. Inequalities (36) and (37) are established by Corollary 15. ■

Equation (34) is also of practical significance showing that $\hat{y}(k)$ is a solution to (18) with $C_1 = 1$, $C_2 = k-1$, $C_3 = C_4 = 0$, and as such can be computed in the same fashion for all k, $1 \le k \le T$ (see Theorem 4 above).

Inequality (36) means that the posterior path probability $p(\hat{y}(k) \mid x^T)$ increases with k. At the same time, increasing k also increases $\bar{R}_1$-risk, that is, decreases the product of the (posterior) marginal probabilities of states along the path $\hat{y}(k)$. Inequalities (36) and (37) clearly show that as k increases, $v(\cdot; k)$ monotonically moves from $v(\cdot; 1)$ (PMAP) towards the Viterbi decoder, that is $v(\cdot; \infty)$. However, the maximum block length is $k = T$.

A natural way to complete this bridging of PMAP with MAP is by embedding the $\bar{R}_k$ risks into the family $\bar{R}_\alpha$ via $\alpha = \frac{k-1}{k} \in [0, 1]$. Thus, (34) extends to

$$\bar{R}_\alpha(s^T \mid x^T) := \alpha\bar{R}_\infty(s^T \mid x^T) + (1-\alpha)\bar{R}_1(s^T \mid x^T) \tag{38}$$

with $\alpha = 0$ and $\alpha = 1$ corresponding to the PMAP and Viterbi cases, respectively. This embedding is clearly still within the generalized Problem (18) via $C_1 = 1 - \alpha$, $C_2 = \alpha$, $C_3 = C_4 = 0$. In particular, $v(x^T; k(\alpha))$ can be computed by using the same dynamic programming algorithm of Theorem 4 for all $k \in [1, \infty]$ (that is, all $\alpha \in [0, 1]$), and inequalities (36) and (37) are special cases of Corollary 15 (part 1) to Lemma 16.

Recalling Remark 3, we note that on the lower end of $0 \le \alpha \le 1$, before reaching PMAP ($\alpha = 0$) we encounter PVD for some sufficiently small $\alpha \approx 0$. Note also that in (35) k need not be integer either, that is, Remark 1 establishes admissibility of $\hat{y}(k(\alpha))$, $k(\alpha) = 1/(1-\alpha)$, for all $\alpha \in (0, 1]$ (that is, all $k \in (1, \infty]$).

Given x^T and a sufficiently large k (equivalently, $\alpha \approx 1$), $\hat{y}(k)$, the minimizer of $\bar{R}_\alpha(s^T \mid x^T)$ (38) (and (34)) would become a Viterbi path $\hat{y}(\infty)$ (since S^T is finite). However, such α (and k) would generally depend on x^T, and in particular k may need to be larger than T, that is, $\hat{y}(T)$ may be different from $\hat{y}(\infty)$.

At the same time, for $k > 1$ we have

$$\bar{R}_\infty(\hat{y}(\infty) \mid x^T) \le \bar{R}_\infty(\hat{y}(k) \mid x^T) \le \bar{R}_\infty(\hat{y}(\infty) \mid x^T) + \frac{\bar{R}_1(\hat{y}(\infty) \mid x^T)}{k-1}, \tag{39}$$

on which we comment more in Section 7 below. The first inequality of (39) above follows immediately from the definition of the Viterbi decoder. To obtain the second inequality, apply (34) to both $\hat{y}(k)$ and $\hat{y}(\infty)$ and subtract one equation from the other. Dividing the

resulting terms by $k-1$, noticing that $\bar{R}_k(\hat{y}(\infty) \mid x^T) \geq \bar{R}_k(\hat{y}(k) \mid x^T)$ and $\bar{R}_1(\hat{y}(k) \mid x^T) \geq 0$, and rearranging the other terms yields the result.

Considering the prior chain Y^T and risks in (31), we immediately obtain statements analogous to (34)-(38) extending these new interpretations to the entire generalized Problem (18). In particular, it might be of general interest to note that for any first order Markov chain (that is, not necessarily representing the posterior distribution of an HMM) the following convexly combined risk

$$\bar{R}_\alpha(s^T) := \alpha \bar{R}_\infty(s^T) + (1-\alpha)\bar{R}_1(s^T)$$

can be efficiently minimized in the usual forward-backward manner (Theorem 4).

5. Experiments

We *illustrate* the performance of the Viterbi, PMAP, and some of the other known and new decoders on the task of predicting protein secondary structure in single amino-acid sequences. We show that the differences in performance between the various decoders can be significant. For this illustration purpose, our decoders are based entirely on the ordinary first order HMM. In particular, when decoding an amino-acid sequence, they do not use cues from decoded homologous sequences (other than by allowing homologous sequences to be part of the training set for estimation of the model parameters). Certainly, successful predictors in practice are significantly more elaborate. In particular, they do exploit intensively information from decoded homologs, and also include interactions at ranges considerably longer than that of the first order HMM (Aydin et al., 2006). However, our current goal is not to compete for the absolute record on the task (which, not so long ago, was reported to be about 70% (Aydin et al., 2006)), but to merely emphasize the following two points. First, the difference in performance between the Viterbi and PMAP decoders can be appreciable in practice already with the ordinary first order HMMs having as few as six hidden states. Secondly, using the new family of decoders (that is, solutions to the generalized risk minimization 18 and 26) gives a potentially useful additional flexibility by exercising trade-offs between principled performance measures (Subsection 1.2.2).

Our data are a non-redundant subset of the *Protein Data Bank* (Berman et al., 2000). Specifically, the secondary structural elements have been found from their atomic coordinates using `SSENVID` (Softberry, Inc., 2001) and the resulting data can be freely downloaded from `http://personal.rhul.ac.uk/utah/113/VA/env_seqssnr.txt`. The data contain $N = 25713$ realizations $(x^{T_n}(n), y^{T_n}(n))$, $n = 1, 2, \ldots, N$, with three original hidden states $\{a, b, c\}$, representing $\alpha-$helix, $\beta-$strand, and coil, respectively. The average length $\bar{T}$ of a realization is 167 positions. The observations $x^{T_n}(n)$ come from a 20 symbol emission alphabet of amino-acids

$$\mathcal{X} = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}.$$

We further distinguish four subclasses of the α-helix class a. The definition and enumeration of the final six classes are as follows: Class one consists of the short, up to seven a long, α-helices. Classes two and three consist of the β-strands (any number of b's) and coil sequences (any number of c's), respectively. Classes four, five, and six derive from the a's

that comprise an α-helix of length at least eight, thereafter referred to as long. Specifically, class four is the so-called N-end, which is the first four a's of a long α-helix. Similarly, class six is the so called C-end, which is the last four a's of a long α-helix. Any a's in the middle of a long α-helix are class five. Refining the original classification has been known to improve prediction of protein secondary structure (Salamov and Solovyev, 1995). For simplicity, here we only sub-divide the α-helix class (whereas Salamov and Solovyev, 1995 go further) given the limited goals of these experiments.

The (maximum likelihood estimates of the) transition and emission distribution matrices as well as the vector of the initial probabilities computed from *all of the realizations* are given in Appendix E.

The following experiments emulate a typical practical situation by re-estimating these parameters from $N-1$ sequences and using the re-estimated values to decode a remaining sequence. We repeat the process N times in the leave-one(sequence)-out fashion. We do not impose stationarity in these experiments as we did not have any prior evidence of stationarity. Indeed, the (estimated) initial distribution $\hat{\pi}$ appears to be very different from the stationary one ($\hat{\pi}_{inv}$, see Appendix E) and many sequences in the data set are quite short.

Figure 1 displays case 877, which is 149 positions long and is split into two pieces at position $t = 72$ (shown in both images). The top (0) row is the ground truth. This case is typical in several senses. First, in this case the PMAP decoder (row 2) shows the median gain in accuracy (of about 11%) over the Viterbi decoder (row 1); see subsequent subsections for a discussion of performance measures. Secondly, the PMAP, or optimal accuracy output, is inadmissible in this case, which is evident from, for example, the isolated state five (yellow) island (transitions between states three and five are forbidden). Rows 3 through 5 are outputs from the PVD, Constrained PMAP, and Rabiner $k = 2$ decoders, respectively. It is typical of the PVD and Constrained PMAP decoders to tie. Outputs from other members of the generalized posterior Viterbi (18) and PMAP (26) hybrid decoders are given in rows 6-18, and 19-31, respectively. Table 1 gives a detailed legend for interpreting the outputs. The monotonicity of the generalized PVD hybrid inference (Corollary 15, part 1, and Corollary 8, inequalities 36 and 37) is illustrated by following the posterior risk columns $\bar{R}_\infty$ and $\bar{R}_1$ across rows 2 (PMAP), then 6 through 17, and finally 1 (Viterbi); PVD (row 3) is attained when $\alpha \approx 0$ (rows 6-9) and here is also indistinguishable from Constrained PMAP (row 4). The monotonicity of the generalized PMAP hybrid inference (Corollary 15, part 3) is illustrated by following the $\bar{R}_\infty$ and R_1 columns across rows 2 (PMAP), then 19 through 30, and finally 1 (Viterbi); Constrained PMAP (row 4) is attained when $\alpha \approx 0$ (rows 19-20) and here is also indistinguishable from PVD (row 3).

Note how the decoder in row 16 (Figure 1) differs from its neighbors, specifically, how it completely misses the terminal activity, which is to a variable extent captured by both its "more accurate" (row 15) and "more probable" (row 17) neighbors.

Rows 18 and 31 are the "data blind" *maximum a priori* and *pointwise maximum a priori* decodings, which are members of both the generalized hybrid families. These decoders tie not only in this but in all the other cases as well; see the structure of the (overall) transition matrix $\mathbb{P}$ in Appendix E also to understand the overwhelming dominance of class 3 ("coil") in the absence of the amino-acid information. By adjusting the R_1 and $\bar{R}_1$ risk terms in the generalized decoders, we can easily accommodate unequal classification penalties to begin

exploring the topology of the posterior distribution (see also Section 8). Thus, for example, we suppress the dominating class 3 to better reveal activity of the remaining classes as shown in Figure 2. Specifically, the marginal posterior probabilities $p_t(s \mid x^T)$ are replaced by $p_t(s \mid x^T)/21$ and $4p_t(s \mid x^T)/21$ for $s = 3$ and $s \neq 3$, respectively; the same re-weighting is also applied to the prior marginal distributions; the Viterbi, Rabiner $k = 2$, as well as the MAPriori decoder (rows 2, 5, 18, respectively) are not affected by this adjustment.

Application specific performance measures will usually be of more interest than the simple measures used here for illustration of the ideas (Section 8). Thus, for example, regarded as β-strand (state 2) detectors, the original decoders (Figure 1) miss four of the seven 2-islands. On the other hand, a more dynamic class 2 activity revealed in Figure 2 correlates very well with the seven objects of class 2. The presence of the adjusted PMAPriori decoder (row 31) also helps to better assess the value of the observed data.

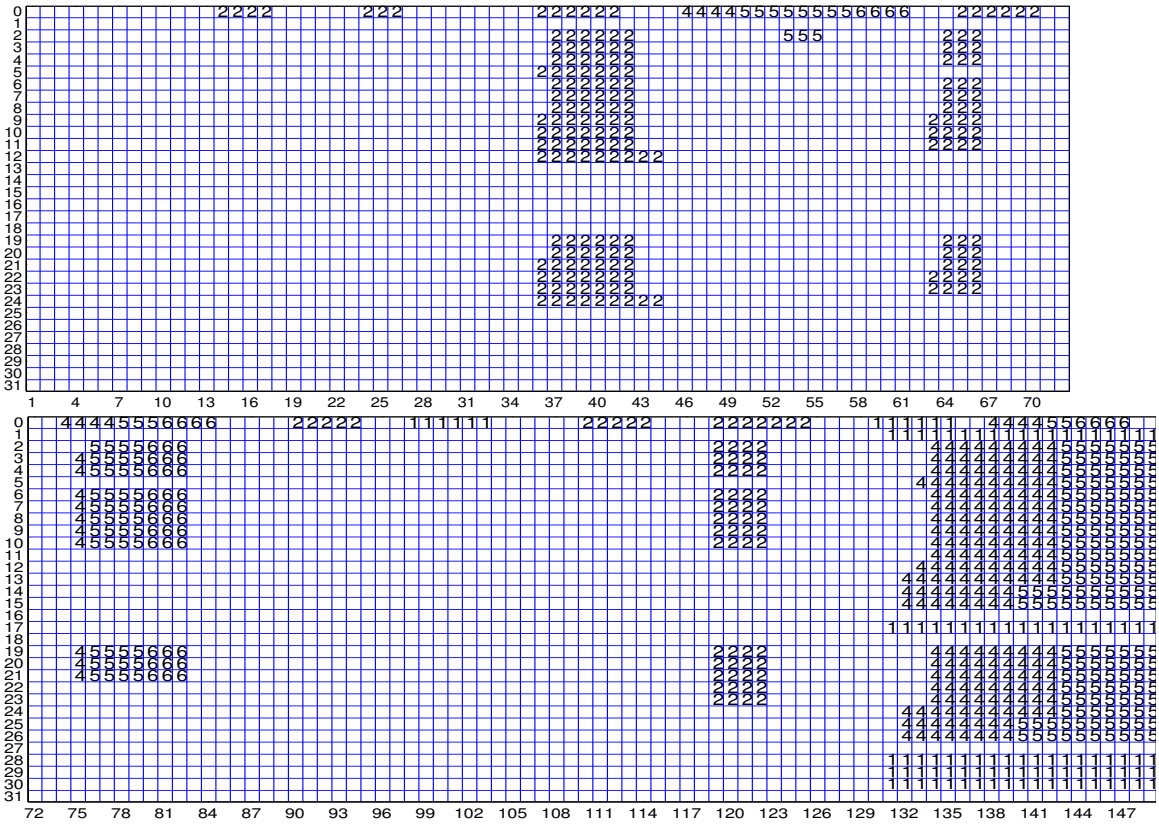


Figure 1: Performance of the well-known and some of the new decoders on Case 877. The dominant class 3 is represented by blank entries. For further legend, see Table 1.

In addition to using the real data, we simulate synthetic data sets each of which having the same number $N = 25713$ of sequences, in the following way. Let $\{\hat{\pi}_{sn}\}_{s\in S}$, $\widehat{\mathbb{P}}_n$, $\{\widehat{P}_{sn}, s \in S\}$ be the estimates of the HMM parameters (initial, transition, and emission distributions, respectively) obtained from $(x^{T_n}(n), y^{T_n}(n))$, the n-th actual realization. Then the n-th

Row	Output $\widehat{y^{149}}$							Empir. error rate(%)	posterior risks		
	Generalized		Alias	C_1	C_2	C_3	C_4				
	PVD	PMAP							$\overline{R}_\infty$	$\overline{R}_1$	$R_1(\%)$
0	Truth							0	0.4907	1.1311	59.2173
1	+	+	Viterbi	0	1	0	0	56.3758	0.1604	0.8296	50.3368
2	+	+	PMAP	1	0	0	0	45.6376	∞	0.6905	46.7752
3a	+		PVD	≈ 1	≈ 0	0	0	46.9799	0.2486	0.6961	46.9188
3b	+			≈ 1	0	0	≈ 0				
4		+	Constr. PMAP	≈ 1	≈ 0	0	0	46.9799	0.2468	0.6961	46.9188
5			Rabiner $k=2$	n/a	n/a	n/a	n/a	53.0201	0.1823	0.7118	47.4429
6	+			0.999	0.001	0	0	46.9799	0.2486	0.6961	46.9188
7	+			0.995	0.005	0	0	46.9799	0.2486	0.6961	46.9188
8	+			0.990	0.010	0	0	46.9799	0.2486	0.6961	46.9188
9	+			0.950	0.050	0	0	46.9799	0.2352	0.6964	46.9322
10	+			0.900	0.100	0	0	46.9799	0.2352	0.6964	46.9322
11	+			2/3	1/3	0	0	53.0201	0.1897	0.7065	47.2499
12	+			0.500	0.500	0	0	54.3624	0.1791	0.7142	47.5372
13	+			1/3	2/3	0	0	56.3758	0.1700	0.7277	48.0356
14	+			0.250	0.750	0	0	57.0470	0.1680	0.7331	48.1738
15	+			0.200	0.800	0	0	57.0470	0.1680	0.7331	48.1738
16	+			0.100	0.900	0	0	57.0470	0.1645	0.7637	48.9620
17	+			0.010	0.990	0	0	56.3758	0.1604	0.8296	50.3368
18	+	+	MA-Prior	0	0	0	1	57.0470	0.1645	0.7637	48.9620
19		+		0.999	0.001	0	0	46.9799	0.2486	0.6961	46.9188
20		+		0.995	0.005	0	0	46.9799	0.2486	0.6961	46.9188
21		+		0.990	0.010	0	0	46.3087	0.2417	0.6962	46.9245
22		+		0.950	0.050	0	0	50.3356	0.2009	0.7021	47.0773
23		+		0.900	0.100	0	0	50.3356	0.2009	0.7021	47.0773
24		+		2/3	1/3	0	0	54.3624	0.1776	0.7165	47.6139
25		+		0.500	0.500	0	0	57.0470	0.1680	0.7331	48.1738
26		+		1/3	2/3	0	0	57.0470	0.1680	0.7331	48.1738
27		+		0.250	0.750	0	0	57.0470	0.1645	0.7637	48.9620
28		+		0.200	0.800	0	0	56.3758	0.1604	0.8296	50.3368
29		+		0.100	0.900	0	0	56.3758	0.1604	0.8296	50.3368
30		+		0.010	0.990	0	0	56.3758	0.1604	0.8296	50.3368
31	+	+	PMA-Prior	0	0	1	0	57.0470	0.1645	0.7637	48.9620

Table 1: Case 877. Performance of the well-known and some of the new decoders. Worst, second worst, best and second best entries in each category are highlighted in red, magenta, blue and cyan respectively. In rows 1, 2, 3a, 6-17, $C_1 = 1 - \alpha = \frac{1}{k}$ and $C_2 = \alpha = 1 - \frac{1}{k}$.

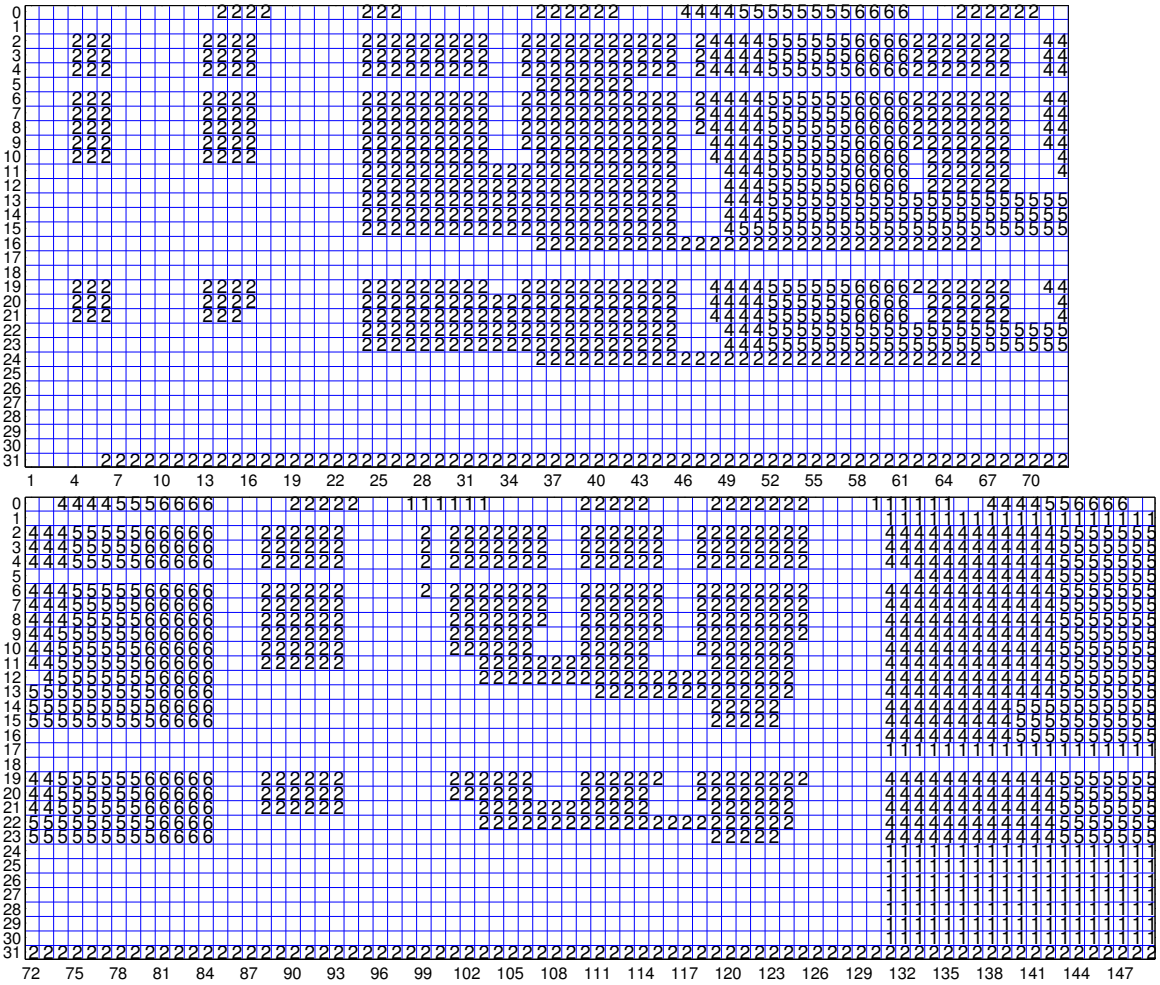


Figure 2: Performance of the selected decoders on Case 877. The dominant class 3 (blank entries) is suppressed by an asymmetric loss incorporated into the R_1 and $\bar{R}_1$ risks of the generalized hybrid decoders. Subsequently, the remaining classes reveal more activity, and in particular all of the seven instances of class 2 can be recognized with essentially only two false alarms.

simulated realization is a sample of length T_n from the (first order homogeneous) HMM with these parameters (note that the initial distributions $\{\hat{\pi}_{sn}\}_{s\in S}$ are necessarily degenerate). The simulations, first of all, help us obtain interval estimates of the performance measures (see more below). Also, they are valuable theoretically. Indeed, the analysis based on the real data tells us what happens in a typical practical scenario in which the (HMM) model is known to be too crude and yet has to be used for its simplicity. The simulations on the contrary tell us what happens when the model is correct. By default, the analysis below refers to the real data, whereas the use of the synthetic data will be acknowledged explicitly.

5.1 Performance Measures and Their Estimation

The performance measures discussed in this subsection will be used in the following two subsections to more completely assess and compare the performance of all the known decoders (including PMAP and Viterbi), and several new members of the generalized families.

Given a decoder v, our principal performance measures are the $R_1(v)$ risk $E[R_1(v(X^T) \mid X^T)]$ (see Equation 5) and the $\bar{R}_\infty$ risk $E[\bar{R}_\infty(v(X^T) \mid X^T)]$ (3); it is not practical to operate with R_∞ (2) since it is virtually 1 for reasonably long realizations. For the $\bar{R}_\infty$ results, see Subsection 5.3.

The R_1 risk is simply the point-wise error rate $\frac{1}{T}\sum_{t=1}^{T} P(\hat{Y}_t \neq Y_t)$, where $\widehat{Y^T}$ is the output of $v(X^T)$. This assumes T to be non-random; more generally, T is random and the R_1 risk is then given by $E_T\left[\frac{1}{T}\sum_{t=1}^{T} P\left(\hat{Y}_t \neq Y_t \mid T\right)\right]$. We refer to $1-R_1$ as *accuracy* when comparing our decoders (e.g., Section 5.2 below). Note that given a decoder v, $R_1(v)$, is simply a parameter of the underlying population of all (T, x^T, y^T) that could potentially be observed. If the current hidden Markov model were not too crude for this population, we would compute such risks if not analytically, then at least by using Monte-Carlo simulations, for any g of interest. In reality, however, we need to estimate them from the given data. The situation is further complicated by the fact that the classification method v is specified only up to the model parameters, which are unknown and also need to be estimated from the data.

All in all, we use the usual cross-validation (CV) estimation. Specifically, to decode $x^{T_n}(n)$, we make g use the estimates of the parameters obtained from the remaining $N-1$ sequences. Thus, if v outputs $\widehat{y^{T_n}}$, then we take the empirical point-wise error rate

$$\hat{e}_n = \frac{1}{T_n}\sum_{t=1}^{T_n} \mathbb{I}_{\{\hat{y}_t \neq y_t(n)\}} \tag{40}$$

to be an estimate of $R_1(v)$. Clearly, if v used the same fixed parameters as used in the definition of $R_1(v)$, then $E[\hat{e}_n] = R_1(v)$, that is, $\hat{e}_n$ would be unbiased for $R_1(v)$, and so would be the average

$$\hat{e}_{CV} = \frac{1}{N}\sum_{n=1}^{N} \hat{e}_n. \tag{41}$$

Obviously, in reality $\hat{e}_{CV}$ is likely to be biased. For this reason we also look at the model-based CV estimate of R_1 given by

$$\hat{R}_1 = \frac{1}{N}\sum_{n=1}^{N} R_1(\widehat{y^{T_n}} \mid x^{T_n}(n)). \tag{42}$$

Computation of $R_1(\cdot \mid x^T)$ indeed relies on the model being correct, hence $\hat{R}_1$ is also likely to be biased. We also report approximate 95% confidence intervals which are based on the usual normal approximation disregarding, among others, any effects of the variability in the realization length T.

If the variation in T were merely an observational artifact, then instead of the above cross-validation averages (42), we would focus on the total error rate for the entire data set given by (43) below.

$$\hat{e} \;=\; \frac{\sum_{n=1}^{N}\sum_{t=1}^{T_n}\mathbb{I}_{\{\hat{y}_t(n)\neq y_t(n)\}}}{\sum_{n=1}^{N}T_n} = \sum_{n=1}^{N} w(n)\hat{e}_n, \quad \text{where } w(n) = \frac{T_n}{\sum_{n=1}^{N}T_n}. \tag{43}$$

However, to obtain sensible confidence intervals in this setting, we need to estimate the variance of $\hat{e}$. Bootstrapping is a possibility, but we instead simulate several (specifically, 15) synthetic data sets as described above in the introduction to this Section, that is, re-sampling individual realizations $(x^{T_n}(n), y^{T_n}(n))$ from the HMM with parameters $\{\hat{\pi}_{sn}\}_{s\in S}$, $\widehat{\mathbb{P}}_n$, $\{\widehat{P}_{sn}, s \in S\}$, $n = 1, 2, \ldots, N$. We then use the t-distribution (on 14 degrees of freedom) to obtain the 95% margins of error.

5.2 Comparison of the Accuracy of the Viterbi and PMAP Decoders

A histogram of the difference $\hat{e}(\text{Viterbi}, n) - \hat{e}(\text{PMAP}, n)$ between the empirical errors (40) of the Viterbi and PMAP decoders is plotted in Figure 3 (black narrow bins). We also observe that in 85.35% of the CV rounds the PMAP decoder is more accurate, and in 10.67%—less accurate, than the Viterbi decoder (in 3.98% of the cases the two methods show the same accuracy). To examine sensitivity of these results to the variation in the realization length, we superimpose in the same Figure 3 a histogram of the subsample consisting of the 1000 longest realizations (blue wide bins). Although the subsample spans a less extreme range $(-16.75\%, 52.62\%)$ than that of the entire sample, the locations of the two histograms are very similar, suggesting the average *gain of accuracy of about 12% when replacing the Viterbi decoder by the PMAP one.*

We also compare the performance of the Viterbi and PMAP decoders by examining their $R_1(\cdot \mid x^{T(n)}(n))$ risks (5), see Figure 4. Note that the difference $\hat{R}_1(\text{Viterbi}) - \hat{R}_1(\text{PMAP})$ is 9% on average, and is largely unchanged (apart from a minor increase) when recomputed on the subsample of the 1000 longest realizations (450-2060 positions).

Finally, $\hat{e}$ (43) is 59.68% ($\pm$0.068%) and 46.10% ($\pm$0.047%) for the Viterbi and PMAP decoders, respectively, and the PMAP comes out $13.58\% \pm 0.0463\%$ more accurate than the Viterbi decoder. The above confidence intervals are, however, likely to be deflated since the model-based simulations show little variation of $\hat{e}$(Viterbi), $\hat{e}$(PMAP), or the differences $\hat{e}(\text{Viterbi}) - \hat{e}(\text{PMAP})$. In fact, based on the 15 model-based simulations, the PMAP is only $7.46\% \pm 0.0463\%$ more accurate than the Viterbi decoder, with the individual error rates of $47.49\% \pm 0.047\%$ and $54.95\% \pm 0.068\%$ for the former and the latter, respectively. Finally, replacing the empirical error rates by the $R_1(\cdot \mid x^T)$ risks (which are now computed exactly since the simulations are model-based), we obtain the difference of $8.55\% \pm 0.0213\%$.

In summary, the PMAP decoder can be notably more accurate than the Viterbi decoder in scenarios with as few as six hidden states.

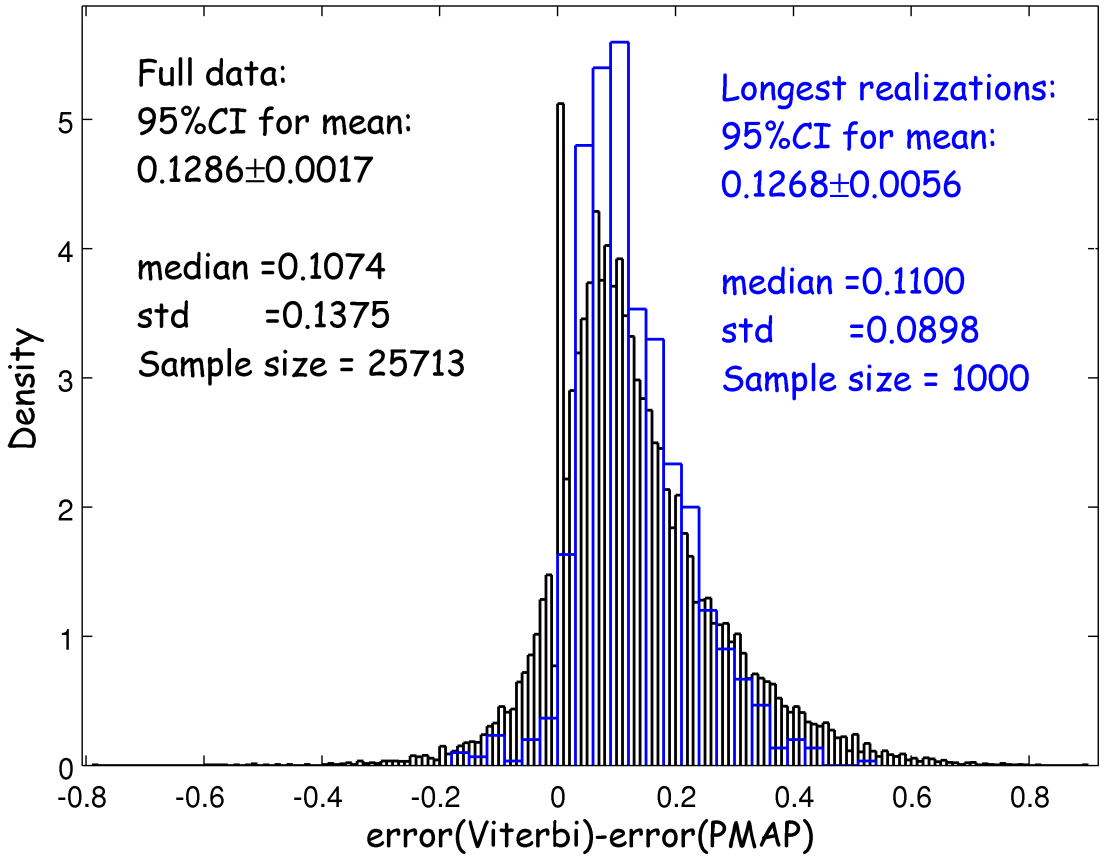


Figure 3: A histogram of the difference between the empirical error rates $\hat{e}(\text{Viterbi}, n) - \hat{e}(\text{PMAP}, n)$ obtained from the full data (black narrow bins) and the subsample consisting of 1000 longest realizations (blue wide bins). Although in 3.98% of the entire data set the two methods show the same accuracy (spike at 0), overall their performance appears to be notably different. The Viterbi decoder is more accurate in 10.67% of all the cases, and the PMAP decoder is more accurate in 85.35% of all the cases. The extreme differences (min $= -78.69\%$,max $=$ 89.74%) tend to be observed on short sequences (136 positions and shorter), but the subsample of the 1000 longest realizations (450-2060 positions) confirms the effect of the PMAP decoder being more accurate. In particular, on the longest sequences, the PMAP decoder can be 52.62% more accurate than the Viterbi decoder, whereas the latter can be at most 16.75% more accurate than the former.

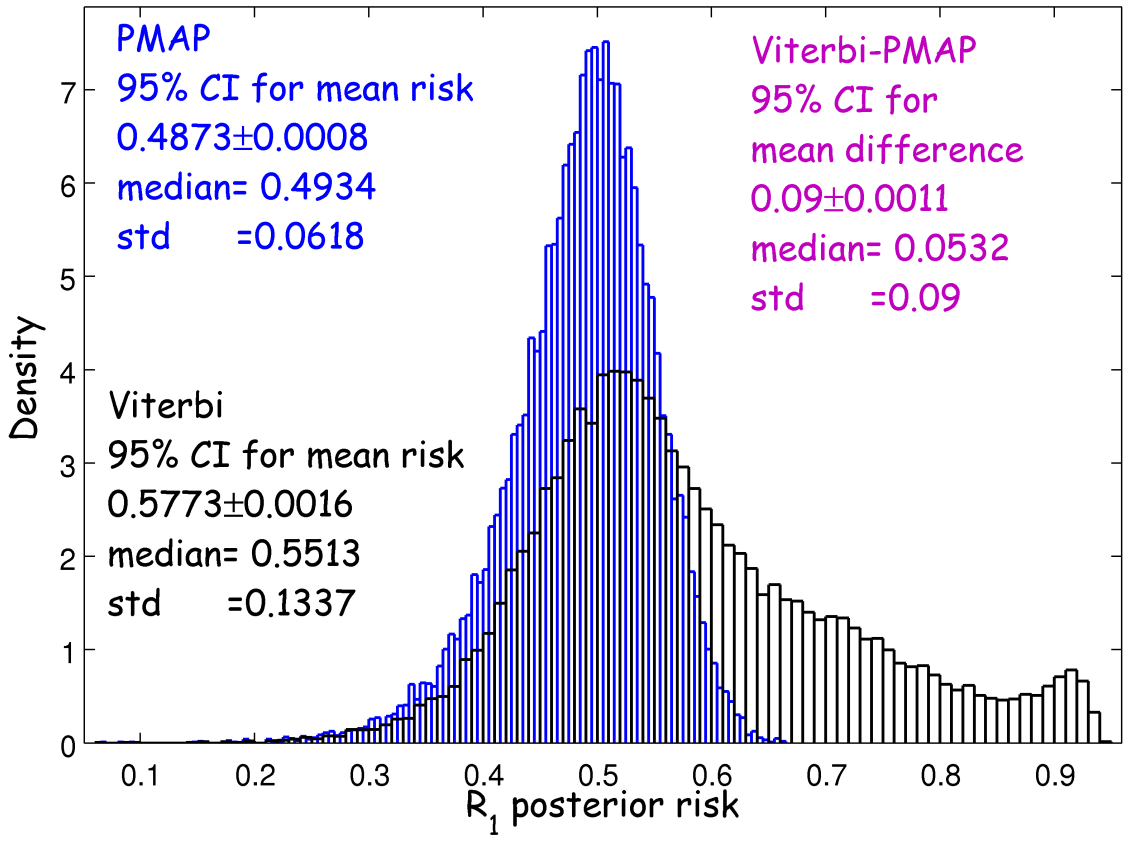


Figure 4: Histograms of the $R_1(\widehat{y^{T(n)}} \mid x^{T(n)}(n))$ risk of the Viterbi (black, more spread) and PMAP (blue, more peaked) decoders. Since the first order homogeneous HMM is only an approximation to the data source, the cross-validation averages of 48.73% (PMAP), 57.73% (Viterbi), and 9% (PMAP's gain over Viterbi) are likely to be biased as estimates of the respective pointwise error rates; see also Figure 3 for a model independent analysis.

5.3 The $\bar{R}_\infty$ Risk of the Viterbi, PMAP and Other Decoders

Next we look at the log-posterior probability rates $\log(P(\widehat{y^T} \mid x^T))/T = -\bar{R}_\infty(\widehat{y^T} \mid x^T)$ of the PMAP, Viterbi and other decoders. In 74.14% of the cases, the PMAP decoder returns an inadmissible path, that is, $\log(P(\widehat{y^T} \mid x^T))/T = -\infty$. To avoid dealing with an infinite range, we switch to the exponential scale. Thus, Figure 5 below displays histograms of the geometric rates $\sqrt[T]{P(\widehat{y^T} \mid x^T)}$.

The Rabiner 2-block decoder $\hat{y}(2)$ returns inadmissible paths in 70.94% of the cases. In 7.32% of the cases this decoder gives an inadmissible path even when the PMAP path (for the same realization) is admissible. This illustrates the violation of monotonicity (see

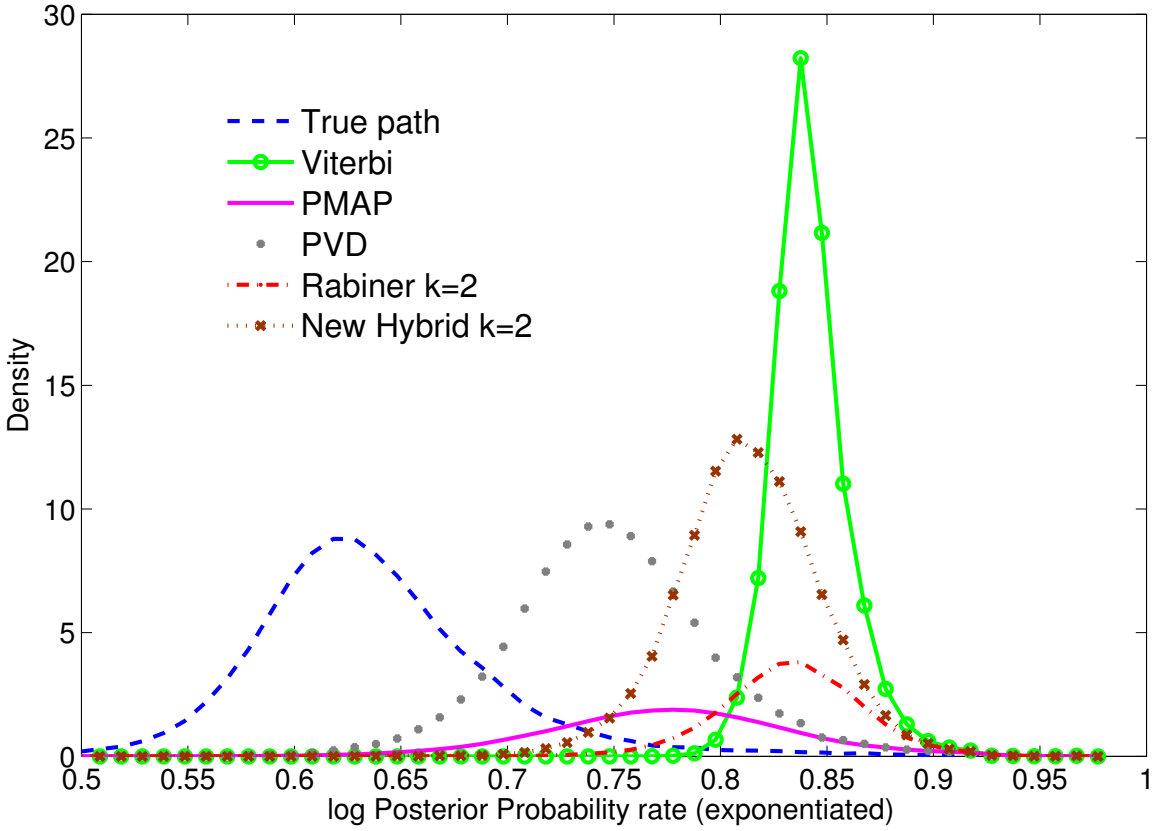


Figure 5: Distributions of the (geometric rates of the) posterior probabilities of selected decoders. The Constrained PMAP decoder is virtually indistinguishable from PVD, hence omitted. The PMAP and Rabiner 2-block (see Subsection 1.2.1) decoders return inadmissible paths in 74.14% and 70.94% of the cases (not shown), respectively (hence only 25.86% and 29.06% of the respective distributions are shown). Just like PVD and the Constrained PMAP decoder, the new hybrid 2-block posterior-Viterbi decoder (33) is guaranteed to produce admissible paths. Moreover, those paths would generally have a higher probability than the probabilities of the PVD and Constrained PMAP paths.

Subsection 1.2.1) in the path (posterior) probability when using Rabiner's suggestion to base decoding on the loss (30).

We also note that the posterior probabilities of the actual hidden paths (blue histogram) are notably lower than those of the admissible decodings, especially the Viterbi outputs. However, these effects are not out of line with the model-based simulations.

5.4 Summary of the Experiments

Figure 6 compares performance of these and other decoders as measured by the averaged error rate and the averaged (exponentiated) path log-posterior rate

$$\widehat{\sqrt[T]{P(\widehat{y^T} \mid x^T)}}_{CV} = \frac{1}{N}\sum_{n=1}^{N} \sqrt[T_n]{P(\widehat{y^{T_n}} \mid x^{T_n}(n))}. \tag{44}$$

Recall that the family of k-block posterior-Viterbi decoders is naturally parameterized by the block length k ($k = 1$ and $k \to \infty$ giving the PMAP and Viterbi decoders, respectively). We have also included the continuous re-parameterization (38) via $k = \frac{1}{1-\alpha}$ (and $\alpha = \frac{k-1}{k}$) which embeds these special cases into the generalized PVD Problem (18) via $C_1 = \alpha$, $C_2 = 1-\alpha$, $C_3 = C_4 = 0$.

Figure 6 displays performance of members of the generalized PVD and generalized PMAP (Problem 26) families with $C_1 = \alpha$, $C_2 = 1-\alpha$, $C_3 = C_4 = 0$ for a subset of values of α used in Figure 1 and Table 1. The point-wise maximum a priori ($C_1 = C_2 = C_4 = 0$, $C_3 = 1$) and the prior-based Viterbi ($C_1 = C_2 = C_3 = 0$, $C_4 = 1$) decoders are also included, showing identical performance on these data. Remarkably (but not very surprisingly given the crudeness of the hidden Markov model for these data), the accuracy of these "data-blind" decoders on average is still higher than that of the Viterbi (MAP) decoder. We reiterate that the hidden Markov model is rather crude as a model for the given data. Furthermore, the estimates of the model parameters used for decoding any given sequence are obtained from sequences that can generally have very different characteristics from the sequence being decoded. Therefore, the risks optimized under these conditions may be misleading, for example, a PMAP path need not have the lowest empirical error rate. Nonetheless, the empirical error rates of the generalized decoders are still found to follow the theoretical order of the posterior R_1 and $\bar{R}_1$ risks.

6. Algorithmic Approaches

It is also possible (at least when the Viterbi path is unique) to hybridize MAP and PMAP inferences without introduction of risk/loss functions. We discuss such approaches mainly because one such approach was taken by Brushe et al. (1998) in what appears to be the only publication dedicated to hybridization of the MAP and PMAP inferences in HMMs.

First note that the hybridization can be achieved by a suitable transformation of the forward and backward variables $\alpha_t(i)$ and $\beta_t(i)$ defined in (1). To make this concrete, consider the recursively applied power transformations with $\mu > 0$ given in (45) below

$$\begin{aligned}
\alpha_1(i;\mu) &:= \alpha_1(i); \qquad (45)\\
\alpha_t(i;\mu) &:= \left[\sum_{j=1}^{K} (\alpha_{t-1}(j;\mu)p_{ji})^{\mu}\right]^{\frac{1}{\mu}} f_i(x_t), \quad t = 2,3\ldots,T;\\
\beta_T(i;\mu) &:= \beta_T(i) = 1;\\
\beta_t(i;\mu) &:= \left[\sum_{j=1}^{K} (p_{ij}f_j(x_{t+1})\beta_{t+1}(j;\mu))^{\mu}\right]^{\frac{1}{\mu}}, \quad t = T-1, T-2, \ldots, 1,
\end{aligned}$$

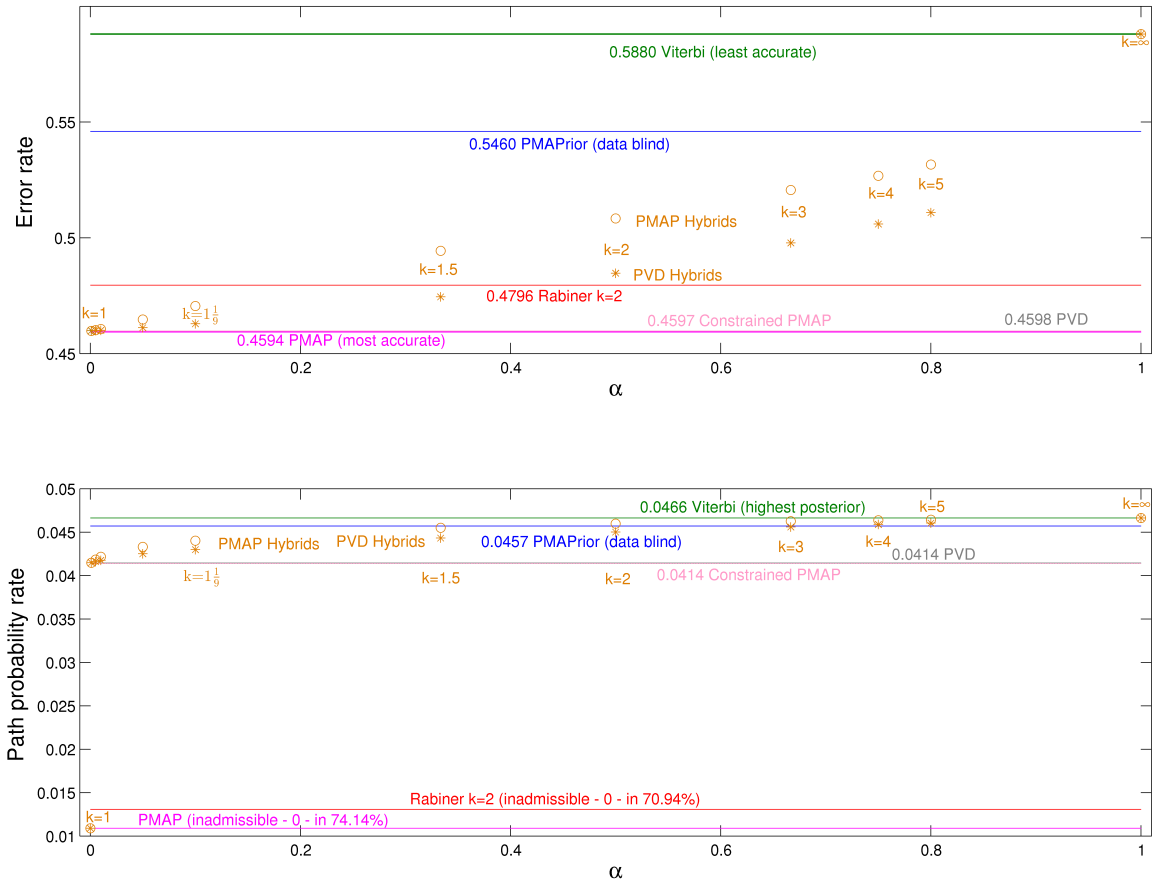


Figure 6: Empirical error (41) (top) and probability rates (44) (bottom) of the popular and some new members of the generalized PVD (asterisk) and PMAP (circle) families.

for all $i \in S$. Clearly, $\alpha_t(i;1) = \alpha_t(i)$ and $\beta_t(i;1) = \beta_t(i)$, for all $i \in S$ and all $t = 1, 2, \ldots, T$. Thus, $\mu = 1$ leads to the PMAP decoding, that is, at time t returning

$$\hat{y}_t(1) = \arg\max_{i \in S} \{\alpha_t(i;1)\beta_t(i;1)\}, \tag{46}$$

provided some tie-breaking rule.

Using induction on t and continuity of the power transform, it can also be seen that the following limits exist and are finite for all $i \in S$ and all $t = 1, 2, \ldots, T$: $\lim_{\mu\to\infty} \alpha_t(i;\mu) =: \alpha_t(i,\infty)$ and $\lim_{\mu\to\infty} \beta_t(i;\mu) =: \beta_t(i;\infty)$, where

$$\begin{aligned} \alpha_t(i;\infty) &= \max_{s^t : s_t = i} p(x^t, s^t), \quad t = 1, 2, \ldots, T, \\ &= \max_{j \in S} \left(\alpha_{t-1}(j;\infty) p_{ji}\right) f_i(x_t), \quad t = 2, 3, \ldots, T, \end{aligned} \tag{47}$$

$$\beta_t(i;\infty) = \max_{s_{t+1}^T \in S^{T-t}} p(x_{t+1}^T, s_{t+1}^T \mid Y_t = i), \quad t = T-1, T-2, \ldots, 1, \quad \text{and } \beta_T(i;\infty) = 1,$$
$$\max_{j \in S} \left(p_{ij} f_j(x_{t+1}) \beta_{t+1}(j;\infty)\right).$$

The above convergence follows from the following trivial observation, which we nonetheless prove below for reasons to become clear later on in the context of Equation (50).

Proposition 9 *Let $a_j(\mu)$, $j = 1, 2, \ldots, K$, be non-negative as functions of $\mu \in (0, \infty)$. Assume that $a_j(\mu)$ converges to some (finite) limit a_j as $\mu \to \infty$. Assume further that for any μ, at least some of the $a_j(\mu)$ are positive. Then we have*

$$\lim_{\mu \to \infty} \left(\sum_{j=1}^{K} a_j(\mu)^\mu \right)^{\frac{1}{\mu}} = \max_{1 \le j \le K} \{a_j\}.$$

Proof Let $M(\mu) = \max_{1 \le j \le K} \{a_j(\mu)\}$, and let $M = \max_{1 \le j \le K} \{a_j\}$. Write $\left(\sum_{j=1}^{K} a_j(\mu)^\mu \right)^{\frac{1}{\mu}} = M(\mu) \left(\sum_{j=1}^{K} \left(\frac{a_j(\mu)}{M(\mu)} \right)^\mu \right)^{\frac{1}{\mu}}$ and note that as $\mu \to \infty$, $M(\mu)$ converges to M. Also, we have

$$1 \le \left(\sum_{j=1}^{K} \left(\frac{a_j(\mu)}{M(\mu)} \right)^\mu \right)^{\frac{1}{\mu}} \le K^{\frac{1}{\mu}}.$$

Since $K^{\frac{1}{\mu}} \to 1$, by the Sandwich Theorem the middle term also converges to 1, yielding the proposed result. ■

Returning to (47), we note that any Viterbi path $\widehat{y^T}(\infty)$ satisfies the following property:

$$\hat{y}_t(\infty) = \arg\max_{i \in S} \{\alpha_t(i;\infty)\beta_t(i;\infty)\}. \tag{48}$$

The above property (48) has already been pointed out by Brushe et al. (1998). The main motivation of Brushe et al. (1998), however, seems to be the case of continuous emission distributions P_s, which might explain why the authors do not consider the fact that *not every path that satisfies* (48) *is necessarily Viterbi, or MAP*. Thus, ignoring potential non-uniqueness of the Viterbi paths, Brushe et al. (1998) state, based on (48), that the Viterbi path can be found *symbol-by-symbol*. As the following simple example shows, *when the Viterbi path is not unique, the attempt to implement the Viterbi decoding in the symbol-by-symbol fashion (based on Equation 48) can produce suboptimal (in the MAP sense), or even inadmissible, paths.*

Example 1 *Let $S = \{1, 2, 3\}$ and let $\{A, B, C, D\}$ be the emission alphabet. Let the initial distribution π, transition probability matrix $\mathbb{P}$, and the emission distributions f_s, $s \in S$, be defined as follows:*

$$\pi = \begin{pmatrix} 0.4 \\ 0.54 \\ 0.06 \end{pmatrix} \quad \mathbb{P} = \begin{pmatrix} 0.6 & 0.4 & 0 \\ 0.1 & 0.1 & 0.8 \\ 0 & 0.02 & 0.98 \end{pmatrix} \quad \begin{array}{lcccc} & A & B & C & D \\ f_1(\cdot) & 0.3 & 0.15 & 0.25 & 0.3 \\ f_2(\cdot) & 0.2 & 0.3 & 0.3 & 0.2 \\ f_3(\cdot) & 1/6 & 1/6 & 1/6 & 1/2 \end{array}.$$

Suppose the sequence $x^2 = (A, B)$ *has been observed. The (posterior) probabilities of all the nine paths* (i, j) *are then summarized in the matrix* $PP = (P(Y^2 = (i, j) \mid AB))$ *below:*

$$PP = \begin{pmatrix} 0.0108 & 0.0144 & 0 \\ 0.0016 & 0.0032 & 0.0144 \\ 0 & 0.0001 & 0.0016 \end{pmatrix},$$

hence there are two Viterbi paths in this case, namely $(1, 2)$ *and* $(2, 3)$*. Now,* $\alpha_1(i; \infty) = \pi_i f_i(A)$*,* $i \in S$*, and* $\beta_1(i; \infty) = \max_{j \in S} P(X_2 = B, Y_2 = j \mid Y_1 = i) = \max_{j \in S} f_j(B) p_{ij}$*, or, in the vector form:*

$$\begin{pmatrix} \alpha_1(1;\infty) \\ \alpha_1(2;\infty) \\ \alpha_1(3;\infty) \end{pmatrix} = \begin{pmatrix} 0.12 \\ 0.108 \\ 0.01 \end{pmatrix}, \quad \begin{pmatrix} \beta_1(1;\infty) \\ \beta_1(2;\infty) \\ \beta_1(3;\infty) \end{pmatrix} = \begin{pmatrix} 0.12 \\ 2/15 \\ 49/300 \end{pmatrix}, \quad \begin{pmatrix} \alpha_1(1;\infty)\beta_1(1;\infty) \\ \alpha_1(2;\infty)\beta_1(2;\infty) \\ \alpha_1(3;\infty)\beta_1(3;\infty) \end{pmatrix} = \begin{pmatrix} 0.0144 \\ 0.0144 \\ 49/30000 \end{pmatrix},$$

so we have $\hat{y}_1(\infty) = 1$ *or* $\hat{y}_1(\infty) = 2$*. On the other hand,* $\alpha_2(i; \infty) = \max_{j \in S} P(X^2 = (A, B), Y^2 = (j, i))$*, and* $\beta_2(i, \infty) = 1$ *for all* $i \in S$*. Therefore,*

$$\begin{pmatrix} \alpha_2(1;\infty) \\ \alpha_2(2;\infty) \\ \alpha_2(3;\infty) \end{pmatrix} = \begin{pmatrix} \alpha_2(1;\infty)\beta_2(1;\infty) \\ \alpha_2(2;\infty)\beta_2(2;\infty) \\ \alpha_2(3;\infty)\beta_2(3;\infty) \end{pmatrix} = \begin{pmatrix} \max\{0.0108, 0.0016, 0\} \\ \max\{0.0144, 0.0032, 0.0001\} \\ \max\{0, 0.0144, 0.0016\} \end{pmatrix} = \begin{pmatrix} 0.0108 \\ 0.0144 \\ 0.0144 \end{pmatrix}.$$

Therefore, $\hat{y}_2(\infty) = 2$ *or* $\hat{y}_2(\infty) = 3$*. However, the symbol-by-symbol decoding is not aware that gluing* $\hat{y}_1(\infty) = 1$ *and* $\hat{y}_2(\infty) = 3$ *is not only suboptimal, but is actually forbidden, that is, results in the inadmissible path* $(1, 3)$*.*

In contrast to Viterbi, the PMAP inference (in the absence of constraints) is by definition *point-wise*, or symbol-by-symbol, hence violation of admissibility is not surprising there regardless of the non-uniqueness issue.

All in all, the main idea of Brushe et al. (1998) is to consider "hybrid" decoders that use intermediate values of the interpolation parameter μ. That is, the hybrid decoder with parameter μ is defined as a decoder that at time t returns

$$\hat{y}_t(\mu) = \arg\max_{i \in S} \{\alpha_t(i; \mu)\beta_t(i; \mu)\}, \tag{49}$$

provided some tie-breaking rule.

Note also that in their attempt to hybridize PMAP with Viterbi in this manner, Brushe et al. (1998) instead of (45) use different transformations that are based on the following $(0, \infty) \to \mathbb{R}$ composite mapping

$$F(\mu, d_1(\mu), d_2(\mu), \ldots, d_N(\mu)) := \frac{1 + (N-1)\exp(-\mu)}{\mu} \log\left(\frac{1}{N}\sum_{j=1}^{N} \exp\left(\mu d_j\left(\mu\right)\right)\right), \tag{50}$$

where $N = K$ (in our notation) and functions $d_j\left(\mu\right)$ are continuous on $[0, \infty)$ with finite limits $d_j(\infty)$ as $\mu \to \infty$. It is then not hard to verify that as $\mu \to 0$, the function (50) converges to $\sum_{j=1}^{N} d_j(0)$ (based on Brushe et al., 1998, Proposition 1a). At the same time, as $\mu \to \infty$ the same function converges to $\max_{1 \le j \le N}\{d_j(\infty)\}$ (based on Brushe et al.,

1998, Proposition 1b). To establish the latter convergence, Brushe et al. (1998) refer to the Varadhan-Laplace Lemma, although the result can also be obtained with basic calculus, for example, by using continuity of the logarithmic function, taking the logarithm inside the limit in Proposition 9, and identifying $a_j(\mu)$ with $e^{d_j(\mu)}$.

This mapping is then applied recursively to $\alpha_t(i;\mu)$ and $\beta_t(i;\mu)$, the analogs of the forward and backward variables ($\kappa_t^\mu(i)$ and $\tau_t^\mu(i)$, respectively, in the notation of Brushe et al., 1998), to produce the correct end points/limits, that is, PMAP and Viterbi/MAP (when the latter is unique). Specifically, the transformed forward and backward variables would be re-defined as follows:

$$\begin{aligned}
\alpha_1(i;\mu) &:= \alpha_1(i); \qquad (51)\\
\alpha_t(i;\mu) &:= \frac{1+(N-1)\,e^{-\mu}}{\mu}\log\left(\frac{1}{N}\sum_{j=1}^{N} e^{\mu\alpha_{t-1}(j;\mu)p_{ji}}\right) f_i\left(x_t\right),\ t=2,3,\ldots,T;\\
\beta_T(i;\mu) &:= \beta_T(i)=1;\\
\beta_t(i;\mu) &:= \frac{1+(N-1)\,e^{-\mu}}{\mu}\log\left(\frac{1}{N}\sum_{j=1}^{N} e^{\mu\beta_{t+1}(j;\mu)p_{ij}f_j(x_{t+1})}\right),\ t=T-1,T-2,\ldots,1.
\end{aligned}$$

Above, we took the liberty to correct $\kappa_1^\mu(i)=\pi(i)$ ($\alpha_1(i;\mu)=\pi_i$ in our notation), which appears in the paper of Brushe et al. (1998) as Equation (22) and also in the proofs of parts (a) and (b) of their Lemma 1. Clearly, in order for $\kappa_1^\mu(i)$ ($\alpha_1(i;\mu)$ in our notation) to match $\alpha_1(i)=P(Y_1=i,X_1=x_1)$ (as claimed in their Lemma 1), $\kappa_1^\mu(i)$ has to equal $\pi(i)b_i(O_1)$ (which is $\pi_i f_i(x_1)$ in our notation). Note that Equation (15) of Brushe et al. (1998) leaves $\alpha_1(i)$ undefined, but instead introduces $\alpha_0(i)$, which is defined to be $\pi(i)$. If that was an implicit intention to introduce a "silent" state at $t=0$, then their Equation (22) and the relevant parts of the proof of Lemma 1 would also have to start with $t=0$ and not with $t=1$. If, on the other hand, $t=0$ in Equation (15) was simply a typing error and the intention was to have $t=1$, then the would-be definition of $\alpha_1(i)=\pi(i)$ contradicts an earlier equation just below their Equation (14), which gives $\alpha_1(i)=P(O_1,q_1=S_i)=\pi(i)b_1(O_1)$ (that is, $P(Y_1=i,X_1=x_1)=\pi_1 f_1(x_1)$ in our notation).

Returning to the essence of the approach, note that the only reason stated by Brushe et al. (1998) for choosing (51) as the family of interpolating transformations is the attainment of the required limits (that is, PMAP when $\mu\to 0$, and Viterbi when $\mu\to\infty$). It is therefore not clear if Brushe et al. (1998) realized that besides (51), there are other (single parameter) families of transformations, such as (45), with the same limiting behavior. Naturally, the resulting interpolation generally depends on the choice of the transformations used. In the absence of any special reason for using (51), (45) may have an appeal for its simplicity, should one really wish to pursue the idea of algorithmic hybridization. Moreover, we explain next (Subsection 6.1) *why the hybrid decoder defined by* (49) *and the transformations* (51) *does not work in practice except with trivial examples*, and we also show (Subsection 6.3) *how this decoder can be modified to become operational*. In contrast to this, we will show (Subsection 6.2) that *the hybrid decoder based on the transformations* (45) *becomes operational by modifying just the algorithm used for its computation, and not the decoder*. This makes the transformations (45) even more attractive as an alternative to (51).

6.1 The Hybrid Decoder Based on the Transformations (51) Does Not Work in Practice Except with Trivial Examples

The key point is that the transform-based algorithmic hybridization attempts to compute quantities which, at least for $\mu \approx 0$, are the same order of magnitude as the forward and backward probabilities $\alpha_t(i) = P(X^t = x^t, Y_t = i)$ and $\beta_t(i) = P(x_{t+1}^T \mid Y_t = i)$. These are well-known to vanish exponentially fast with T, see, for example, Bishop (2006, 13.2.4) who also note that "[f]or moderate lengths of chain (say 100 or so), the calculation of the $[\alpha_t(j)]$ will soon exceed the dynamic range of the computer, even if double precision floating point is used." The situation clearly gets worse as μ increases. Indeed, recall (47), and note that $\max_{s^t:s_t=i} p(x^t, s^t) = \alpha_t(i;\infty) \leq \sum_{s^t:s_t=i} p(x^t, s^t) = \alpha_t(i)$ (which is also $\alpha_t(j;1)$ in Equation 45 and $\alpha_t(j;0)$ in Equation 51). This easily leads to a collapse of computations already with chains as short as $T = 10$ (which indeed happens using the data and model from our experiments of Section 5 above).

We disagree with Brushe et al. (1998) in interpreting the nature of the above numerical problems when they divert the reader's attention to the computation of the `logsumexp` function used in their transforms (50), (51). We find this is misleading as the $\log(e^a + e^b) = \max\{a, b\} + \log\left(1 + e^{-|a-b|}\right)$ trick (alluded to by Brushe et al., 1998 in their Remark below Equation 25) is relevant to the problem of underflow only of the *intermediate values* (that is, $e^a + e^b$ when a or b is negative of a large magnitude, such as the logarithm of a very small probability). In the case of the transform (50), however, computations of the transformed, say, forward variable $\alpha_t(i;\mu)$ (51), do require $\mu d_j(\mu) = \mu\alpha_{t-1}(j;\mu)p_{ji}$ and not their logarithm. Thus, at some t underflow in $\alpha_t(i;\mu)$ occurs for some i, and then eventually for all i. In terms of the `logsumexp` function, this means that both e^a and e^b become 1 (and not zero!) but the logarithm of their average (the core of the transform 50) becomes 0, transferring the underflow to the next generation, that is, $\alpha_{t+1}(i;\mu)$. Thus, storing $\alpha_t(i;\mu)$ in the log-domain is irrelevant here since the transforms (50), (51) with or without the `logsumexp` trick, do require the actual value of $\alpha_t(i;\mu)$. One could conceivably introduce the `loglogsumexpexp` function to operate on $\log(\alpha_t(i;\mu))$ and resolve this problem in that way, but it is not clear if the goal is worth the effort.

Furthermore, insisting that "[t]he computational complexity and numerical implementation issues associated with the hybrid algorithm can be overcome using the Jacobian logarithm", Brushe et al. (1998, p. 3133) repeatedly refer to another paper, which proposes to compute the `logsumexp` function $\log(\sum_k \exp(a_k))$ via recursive application of $\log(e^a + e^b) = \max\{a, b\} + \log\left(1 + e^{-|a-b|}\right)$. Although this recursive implementation should indeed be generally more accurate (albeit also computationally more expensive) than the commonly used single-shift implementation $\log(\sum_k \exp(a_k)) = M + \log(\exp(a_k - M))$ ($M = \max_k\{a_k\}$), as we just explained above, it is irrelevant to the real problem of computing the transformed forward and backward variables $\alpha_t(i;\mu)$, $\beta_t(i;\mu)$ ($\kappa_t^\mu(i)$, $\tau_t^\mu(i)$, respectively, of Brushe et al., 1998). Thus, the approach of Brushe et al. (1998) *does not immediately provide an operational decoding algorithm* except for trivially short chains. For example, using the two-state HMM from the Example 2 and the 64-bit MATLAB (MATLAB, 2011) (but without The Symbolic Math Toolbox) installation on a (64-bit) Linux machine, the hybrid decoder based on (51) with $\mu = 1$ already fails for $T = 40$ (with or

without the `logsumexp` trick). For comparison, the hybrid decoder based on the power transform (45) ($\mu = 1$) survives an order of magnitude longer.

A natural question is then whether the transform-based algorithmic hybridization approach (using (51) or (45), or the like) can at all work in practice. The fact that no such example has been given by Brushe et al. (1998), or anyone else up to date, casts some doubt. Below we give reassuring answers, which have been verified to work on several realistic examples.

Indeed, it is well-known that in practice, to decode the t-th symbol the PMAP decoder uses the posterior probabilities $p_t(i \mid x^T)$ and not the vanishing joint probabilities $p_t(i \mid x^T)p(X^T = x^T) = P(x^T, Y_t = i) = \alpha_t(i)\beta_t(i)$. The posterior probabilities $p_t(i \mid x^T)$ are computed as $\tilde{\alpha}_t(i)\tilde{\beta}_t(i)$, where $\tilde{\alpha}_t(i) = P(Y_t = i \mid x^t)$ and $\tilde{\beta}_t(i) = P(x_{t+1}^T \mid Y_t = i)/p(x_{t+1}^T \mid x^t)$ are the scaled analogs of the forward and backward probabilities $\alpha_t(i)$ and $\beta_t(i)$ (Bishop, 2006, 13.2.4). This allows PMAP to bypass the aforementioned problem of numerical underflow.

6.2 The Hybrid Decoder (49) is Invariant to Rescaling of the Power-Transformed (45) Forward and Backward Variables $\alpha(\cdot;\mu)$, $\beta(\cdot;\mu)$.

Let us apply the same normalization approach to the transformed forward and backward variables, first, using the power transform (45) and then (51). First, recall (e.g., Bishop, 2006, 13.2.4) that $\tilde{\alpha}_t(i)$ are obtained by replacing the recursive definition

$$\alpha_t(i) = f_i(x_t) \sum_{j=1}^{K} \alpha_{t-1}(j) p_{ji}, \quad i = 1, 2, \ldots, K,$$

by the two-step self-normalized definition

$$\begin{aligned} p(x_t \mid x^{t-1})\tilde{\alpha}_t(i) &= f_i(x_t) \sum_{j=1}^{K} \tilde{\alpha}_{t-1}(j) p_{ji}, \quad i = 1, 2, \ldots, K, \\ \tilde{\alpha}_t(i) &= \frac{p(x_t \mid x^{t-1})\tilde{\alpha}_t(i)}{\sum_{s=1}^{K} p(x_t \mid x^{t-1})\tilde{\alpha}_t(s)}, \quad \text{for } t = 2, \ldots, T, \\ \text{where} \quad \tilde{\alpha}_1(i) &= \alpha_1(i)/c_1, \text{ and } c_1 := p(x_1) = \sum_{s=1}^{K} \alpha_1(s). \end{aligned}$$

Thus, for all $t = 2, 3 \ldots T$, and for all $i = 1, 2, \ldots, K$,

$$\tilde{\alpha}_t(i) = \frac{f_i(x_t) \sum_{j=1}^{K} \tilde{\alpha}_{t-1}(j) p_{ji}}{c_t}, \quad \text{where, also according to Bishop (2006, Equation 13.56),}$$

$$c_t := p(x_t \mid x^{t-1}) = \sum_{s=1}^{K} f_s(x_t) \sum_{j=1}^{K} \tilde{\alpha}_{t-1}(j) p_{js}.$$

Similarly, the rescaled backward variables are given by

$$\begin{aligned} \tilde{\beta}_T(i) &:= 1; \\ \tilde{\beta}_t(i) &:= \frac{\sum_{j=1}^{K} p_{ij} f_j(x_{t+1}) \tilde{\beta}_{t+1}(j)}{c_{t+1}}, \quad t = T-1, T-2, \ldots, 1. \end{aligned}$$

In the same manner, we normalize the $\alpha_t(i;\mu)$ and $\beta_t(i;\mu)$ (defined by equations 45) for any $\mu > 0$ as follows:

$$\begin{aligned}
\tilde{\alpha}_1(i;\mu) &:= \alpha_1(i)/c_1(\mu) = \tilde{\alpha}_1(i), \quad \text{where } c_1(\mu) := c_1 \text{ for all } \mu; \\
\tilde{\alpha}_t(i;\mu) &:= \frac{\left[\sum_{j=1}^K \left(\tilde{\alpha}_{t-1}(j;\mu)p_{ji}\right)^\mu\right]^{\frac{1}{\mu}} f_i(x_t)}{c_t(\mu)}, \quad t = 2,3,\ldots,T; \\
\tilde{\beta}_T(i;\mu) &:= \beta_T(i) = 1; \\
\tilde{\beta}_t(i;\mu) &:= \frac{\left[\sum_{j=1}^K \left(p_{ij}f_j(x_{t+1})\tilde{\beta}_{t+1}(j;\mu)\right)^\mu\right]^{\frac{1}{\mu}}}{c_{t+1}(\mu)}, \quad t = T-1, T-2, \ldots, 1,
\end{aligned} \tag{52}$$

where

$$c_t(\mu) := \sum_{s=1}^K \left[\sum_{j=1}^K \left(\tilde{\alpha}_{t-1}(j;\mu)p_{js}\right)^\mu\right]^{\frac{1}{\mu}} f_s(x_t), \quad t = 2,3,\ldots,T.$$

Thus, $c_t(1) = c_t$ for all $t = 1,2,\ldots,T$. Also note that, using induction on t and (47), $\lim_{\mu\to 1} c_t(\mu) = c_t(1)$, and the limits $c_t(\infty) := \lim_{\mu\to\infty} c_t(\mu)$ exist and are finite for all $t = 1,2,\ldots,T$.

Proposition 10 *For any $i \in S$, we have*

1) $\tilde{\alpha}_t(i;\mu) = \frac{\alpha_t(i;\mu)}{\sum_{s=1}^K \alpha_t(s;\mu)} = \frac{\alpha_t(i;\mu)}{\prod_{m=1}^t c_m(\mu)}$ *for all* $t = 1,2,\ldots,T$, *and* $\tilde{\beta}_t(i;\mu) = \frac{\beta_t(i;\mu)}{\prod_{m=t+1}^T c_m(\mu)}$ *for all* $t = 1,2,\ldots,T-1$ *and for all* $\mu > 0$*;*

2) $\lim_{\mu\to 1}\tilde{\alpha}_t(i;\mu) = \tilde{\alpha}_t(i)$, $\lim_{\mu\to 1}\tilde{\beta}_t(i;\mu) = \tilde{\beta}_t(i)$ *for all* $t = 1,2,\ldots,T$*;*

3) $\lim_{\mu\to\infty}\tilde{\alpha}_t(i;\mu) = \tilde{\alpha}_t(i;\infty) := \frac{\alpha_t(i;\infty)}{\sum_{s=1}^K \alpha_t(s,\infty)}$, *for all* $t = 1,2,\ldots,T$, *and* $\lim_{\mu\to\infty}\tilde{\beta}_t(i;\mu) =: \tilde{\beta}_t(i;\infty) = \frac{\beta_t(i;\infty)}{\prod_{m=t+1}^T c_m(\infty)}$, *for all* $t = 1,2,\ldots,T-1$, *and, finally,* $\lim_{\mu\to\infty}\tilde{\beta}_T(i;\mu) =: \tilde{\beta}_T(i;\infty) = 1$ *trivially;*

4) *The hybrid decoder* (49) *based on the transformations* (45) *and the hybrid decoder* (49) *based on the transformations* (52) *are one and the same decoder, provided that both use the same tie-breaking rule.*

Proof The first claim concerning the $\tilde{\alpha}_t$ is trivially true for $t = 1$ by definition of $\alpha_1(i;\mu)$, that is (45). Now, using induction on t, assume that the claim is true for $t-1$. Write $a_{t-1}(\mu)$ for $(\sum_{s=1}^K \alpha_{t-1}(s;\mu))^{-1}$ so that $a_{t-1}(\mu)\alpha_{t-1}(j;\mu) = \tilde{\alpha}_{t-1}(j;\mu)$ and $a_{t-1}(\mu) = (\prod_{m=1}^{t-1} c_m(\mu))^{-1}$. Then, using (52), we get

$$\tilde{\alpha}_t(i;\mu) = \frac{\left(\sum_{j=1}^K \left(a_{t-1}(\mu)\alpha_{t-1}(j;\mu)p_{ji}\right)^\mu\right)^{\frac{1}{\mu}} f_i(x_t)}{\sum_{s=1}^K \left(\sum_{j=1}^K \left(a_{t-1}(\mu)\alpha_{t-1}(j;\mu)p_{js}\right)^\mu\right)^{\frac{1}{\mu}} f_s(x_t)},$$

which, upon cancellation of the $a_{t-1}(\mu)$, yields the required result

$$\frac{\left(\sum_{j=1}^K (\alpha_{t-1}(j;\mu)p_{ji})^\mu\right)^{\frac{1}{\mu}} f_i(x_t)}{\sum_{s=1}^K \left(\sum_{j=1}^K (\alpha_{t-1}(j;\mu)p_{js})^\mu\right)^{\frac{1}{\mu}} f_s(x_t)} = \frac{\alpha_t(i;\mu)}{\sum_{s=1}^K \alpha_t(s;\mu)}.$$

To see that $\tilde{\alpha}_t(i;\mu)$ also equals $\frac{\alpha_t(i;\mu)}{\prod_{m=1}^t c_m(\mu)}$, write

$$\tilde{\alpha}_t(i;\mu) = \frac{\left(\sum_{j=1}^K (a_{t-1}(\mu)\alpha_{t-1}(j;\mu)p_{ji})^\mu\right)^{\frac{1}{\mu}} f_i(x_t)}{c_t(\mu)} = \frac{\left(\sum_{j=1}^K (\alpha_{t-1}(j;\mu)p_{ji})^\mu\right)^{\frac{1}{\mu}} f_i(x_t)}{(\prod_{m=1}^{t-1} c_m(\mu))c_t(\mu)},$$

which, recalling the original (unscaled) $\alpha_t(i;\mu)$ recursion, yields the result.

The β variables are handled analogously.

The second claim is then a straightforward consequence of the first claim and the continuity (with respect to μ, and in particular at $\mu = 1$) of the power transform; for example, to establish the result for the $\tilde{\beta}_t(i;\mu)$, observe that $\prod_{m=t+1}^T c_m(\mu) \to \prod_{m=t+1}^T c_m(1)$ when $\mu \to 1$. The third claim also immediately follows from the first one and Proposition 9, also noticing that $\prod_{m=t+1}^T c_m(\mu) \to \prod_{m=t+1}^T c_m(\infty)$ as $\mu \to \infty$. The fourth claim also immediately follows from the first claim as v_t maximizes $\alpha_t(i;\mu)\beta_t(i;\mu)$ if and only if it maximizes $\tilde{\alpha}_t(i;\mu)\tilde{\beta}_t(i;\mu)$. ■

In particular, we arrive at the following characterization of the Viterbi paths $\widehat{y^T}(\infty)$, which is now possible to compute in practice for a wide range of models and parameters in contrast to the condition (48):

Corollary 11 *For any $t = 1, 2, \ldots, T$, $\hat{y}_t(\infty) = \arg\max_{i\in S}\{\tilde{\alpha}_t(i;\infty)\tilde{\beta}_t(i;\infty)\}$.*

Recall (46), and thus note that the PMAP decoder also maximizes $\tilde{\alpha}_t(i;1)\tilde{\beta}_t(i;1)$. As a side note, consider also the following decoder $v(x^T;0)$ that extrapolates the normalized power-transformed decoder to $\mu \to 0$, that is "beyond" the PMAP decoding. Namely, for any $t = 1, 2, \ldots, T$, let $v_t = \arg\max_{i\in S}\{\tilde{\alpha}_t(i;0)\tilde{\beta}_t(i;0)\}$, where for any $i \in S$,

$$\begin{aligned}
\tilde{\alpha}_1(i;0) &:= \alpha_1(i)/c_1 = \tilde{\alpha}_1(i); \qquad (53)\\
\tilde{\alpha}_t(i;0) &:= \frac{\left[\prod_{j\in S_t(i)} \tilde{\alpha}_{t-1}(j;0)p_{ji}\right]^{\frac{1}{K_t(i)}} f_i(x_t)}{\sum_{s=1}^K \left[\prod_{j\in S_t(s)} \tilde{\alpha}_{t-1}(j;0)p_{js}\right]^{\frac{1}{K_t(s)}} f_s(x_t)}, \quad t = 2, 3, \ldots, T,\\
\text{where } S_t(i) &:= \{j \in S :\ \tilde{\alpha}_{t-1}(j;0)p_{ji} > 0\} \text{ and } K_t(i) := |S_t(i)|, \text{ that is size of } S_t(i);\\
\tilde{\beta}_T(i;0) &:= \beta_T(i) = 1;\\
\tilde{\beta}_t(i;0) &:= \frac{\left[\prod_{j\in S_t^*(i)} p_{ij} f_j(x_{t+1})\tilde{\beta}_{t+1}(j;0)\right]^{\frac{1}{K_t^*(i)}}}{\sum_{s=1}^K \left[\prod_{j\in S_{t+1}(s)} \tilde{\alpha}_t(j;0)p_{js}\right]^{\frac{1}{K_{t+1}(s)}} f_s(x_{t+1})}, \quad t = T-1, T-2, \ldots, 1,\\
\text{where } S_t^*(i) &:= \{j \in S :\ p_{ij} f_j(x_{t+1})\tilde{\beta}_{t+1}(j;0) > 0\} \text{ and } K_t^*(i) := |S_t^*(i)|.
\end{aligned}$$

Corollary 12 *Assume that* $\lim_{\mu\to 0}\tilde{\alpha}_t(i;\mu) > 0$ *and* $\lim_{\mu\to 0}\tilde{\beta}_t(i;\mu) > 0$ *for all* $i \in S$ *and all* $t = 1, 2, \ldots, T$. *Then* $\tilde{\alpha}_t(i;0) = \lim_{\mu\to 0}\tilde{\alpha}_t(i;\mu)$ *and* $\lim_{\mu\to 0}\tilde{\beta}_t(i;\mu) = \tilde{\beta}_t(i;0)$ *for all* $i \in S$ *and all* $t = 1, 2, \ldots, T$, *that is the decoder* (49) *based on the transformations* (52) *converges (upto the tie-breaking rule) to the decoder defined by* (53) *above.*

Proof This is a straightforward exercise in calculus, that is, using continuity of the exponential function and invoking Proposition 1a of Brushe et al. (1998), with the positivity assumption making all $K_t(i)$ and $K_t^*(i)$ equal to K. ■

Note also that the hybrid decoder (49) based on the original, that is, unnormalized variables (45), generally does not have a limit as $\mu \to 0$.

6.3 Rescaling of the Forward and Backward Variables $\alpha(\cdot;\mu)$ and $\beta(\cdot;\mu)$ Defined by (51) Alters the Hybrid Decoder (49).

In the same manner as in (52) above, we now normalize the $\alpha(\cdot;\mu)$ and $\beta(\cdot;\mu)$ variables transformed according to (51). Thus, for any $\mu > 0$ and for any $i \in S$, let

$$
\begin{aligned}
\check{\alpha}_1(i;\mu) &:= \alpha_1(i)/\sum_{s=1}^{K}\alpha_1(s) = \tilde{\alpha}_1(i); \qquad (54)\\
\check{\alpha}_t(i;\mu) &:= \frac{\log\left[\frac{1}{K}\sum_{j=1}^{K} e^{\mu\check{\alpha}_{t-1}(j;\mu)p_{ji}}\right] f_i(x_t)}{\sum_{s=1}^{K}\log\left[\frac{1}{K}\sum_{j=1}^{K} e^{\mu\check{\alpha}_{t-1}(j;\mu)p_{js}}\right] f_s(x_t)}, \quad t = 2, 3, \ldots, T;\\
\check{\beta}_T(i;\mu) &:= \beta_T(i) = 1, \quad t = T-1, T-2, \ldots, 1;\\
\check{\beta}_t(i;\mu) &:= \frac{\log\left[\frac{1}{K}\sum_{j=1}^{K} e^{\mu p_{ij} f_j(x_{t+1})\check{\beta}_{t+1}(j;\mu)}\right]}{\sum_{s=1}^{K}\log\left[\frac{1}{K}\sum_{j=1}^{K} e^{\mu\check{\alpha}_t(j;\mu)p_{js}}\right] f_s(x_{t+1})}, \quad t = T-1, T-2, \ldots, 1.
\end{aligned}
$$

Proposition 13 *For any* $i \in S$, *we have*

1) $\lim_{\mu\to 0}\check{\alpha}_t(i;\mu) = \tilde{\alpha}_t(i)$, $\lim_{\mu\to 0}\check{\beta}_t(i;\mu) = \tilde{\beta}_t(i)$ *for all* $t = 1, 2, \ldots, T$;

2) $\lim_{\mu\to\infty}\check{\alpha}_t(i;\mu) = \tilde{\alpha}_t(i;\infty)$ *and* $\lim_{\mu\to\infty}\check{\beta}_t(i;\mu) = \tilde{\beta}_t(i;\infty)$, *for all* $t = 1, 2, \ldots, T$.

3) *The hybrid decoder* (49) *based on the transformations* (51) *and the hybrid decoder* (49) *based on the transformations* (54) *are generally different, even if both use the same tie-breaking rule.*

Proof The first two claims are straightforward extensions of Lemmas 1 and 2 of Brushe et al. (1998). To see this, first restore the previously reduced factor $\frac{1+(K-1)e^{-\mu}}{\mu}$ in both the numerator and denominator of the expressions for $\check{\alpha}_t(i;\mu)$ and $\check{\beta}_t(i;\mu)$. Then apply induction on t (first in the forward manner for the α variables and then backward for the β variables). For example, assume that $\lim_{\mu\to\infty}\check{\beta}_{t+1}(i;\mu) = \tilde{\beta}_{t+1}(i;\infty)$. Then, as $\mu \to \infty$,

$$
\frac{1+(K-1)e^{-\mu}}{\mu}\log\left[\frac{1}{K}\sum_{j=1}^{K} e^{\mu p_{ij} f_j(x_{t+1})\check{\beta}_{t+1}(j;\mu)}\right] \;\to\; \max_{j\in S}\left(p_{ij} f_j(x_{t+1})\tilde{\beta}_{t+1}(j;\infty)\right),
$$

which is, according to claim 3 of Proposition 10,

$$\max_{j\in S}\left(p_{ij}f_j(x_{t+1})\beta_{t+1}(j;\infty)/\prod_{m=t+2}^{T}c_m(\infty)\right) \;=\; \max_{j\in S}\left(p_{ij}f_j(x_{t+1})\beta_{t+1}(j;\infty)\right)/\prod_{m=t+2}^{T}c_m(\infty).$$

Next, recalling (47), we get that the numerator in the expression for $\lim_{\mu\to\infty}\check{\beta}_t(i;\mu)$ is given by $\beta_t(i;\infty)/\prod_{m=t+2}^{T}c_m(\infty)$. Observing that the denominator is given by

$$\lim_{\mu\to\infty}\frac{1+(K-1)e^{-\mu}}{\mu}\sum_{s=1}^{K}\log\left[\frac{1}{K}\sum_{j=1}^{K}e^{\mu\check{\alpha}_t(j;\mu)p_{js}}\right]f_s(x_{t+1}) \;=\; \sum_{s=1}^{K}\max_{j\in S}\left(\tilde{\alpha}_t(j;\infty)p_{js}\right)f_s(x_{t+1}),$$

which is just $c_{t+1}(\infty)$, finally gives $\lim_{\mu\to\infty}\check{\beta}_t(i;\mu)=\beta_t(i;\infty)/\prod_{m=t+1}^{T}c_m(\infty)=\tilde{\beta}_t(i;\infty)$, as required.

As a counter-example proving the last claim, consider the simple HMM from The MathWorks, Inc. (2012, p. 1840).

Example 2 *Let $S=\{1,2\}$ and let $\{1,2,\ldots,6\}$ be the emission alphabet. Let the initial distribution π, transition probability matrix $\mathbb{P}$, and the emission distributions f_s, $s\in S$, be defined as follows:*

$$\pi=\begin{pmatrix}2/3\\1/3\end{pmatrix},\quad \mathbb{P}=\begin{pmatrix}0.95 & 0.05\\0.1 & 0.9\end{pmatrix},\ \pi^t\mathbb{P}=\pi^t,\qquad \begin{array}{lcccccc} & 1 & 2 & 3 & 4 & 5 & 6\\ f_1(\cdot) & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6\\ f_2(\cdot) & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.2\end{array}.$$

Suppose $x^5=(2,6,6,4,1)$ has been observed. Take $\mu=7$. Table 2 shows outputs of the original (top) and normalized (bottom) transformed decoders, respectively. Clearly, the decoders return different paths.

■

Note that unlike the normalized hybrid decoder based on the power-transform, this normalized hybrid decoder generally does not satisfy the first claim of Proposition 10. (Indeed, satisfying these conditions would contradict the third claim of the latter Proposition 13.)

We have also experimented with these normalized hybrid decoders using a subset of real data (and a realistic HMM with $K=6$ states) from our experimental Section 5 and can indeed confirm convergence of the hybrid decoder based (54) to the PMAP decoder with $\mu=0.001$ and to the Viterbi decoder with $\mu=10000$ for sequences of length $T=100$. Naturally, the above range of μ values would generally need to increase significantly with T.

Below, we summarize our views on the idea of purely algorithmic hybridization of MAP and PMAP.

1. The method presented by Brushe et al. (1998) need not work, that is, can fail to converge to the Viterbi path, when the Viterbi path is not unique, see Example 1 above.

2. Since the method depends on the transformation used, more work may be needed to understand which (if any) particular transformation/interpolation could be suitable for a specific application; the choice of (51) made by Brushe et al. (1998) seems to be rather arbitrary.

t	$\alpha_t(1;\mu)$	$\beta_t(1;\mu)$	$\alpha_t(2;\mu)$	$\beta_t(2;\mu)$	$\alpha_t(1;\mu)\beta_t(1;\mu)$	$\alpha_t(2;\mu)\beta_t(2;\mu)$
					10^{-6}	10^{-6}
1	0.11111	6.6968e-05	0.033333	0.00019826	**7.4409**	6.6088
2	0.010576	0.00071029	0.0091583	0.00085352	7.5121	**7.8168**
3	0.0009266	0.0083987	0.0022209	0.003471	**7.7823**	7.7088
4	9.201e-05	0.10141	0.00010268	0.058041	**9.3311**	5.9598
5	8.1481e-06	1	4.8559e-06	1	**8.1481**	4.8559

t	$\check{\alpha}_t(1;\mu)$	$\check{\beta}_t(1;\mu)$	$\check{\alpha}_t(2;\mu)$	$\check{\beta}_t(2;\mu)$	$\check{\alpha}_t(1;\mu)\check{\beta}_t(1;\mu)$	$\check{\alpha}_t(2;\mu)\check{\beta}_t(2;\mu)$
1	0.76923	0.30879	0.23077	0.97296	**0.23753**	0.22453
2	0.58963	0.55137	0.41037	0.55227	**0.32510**	0.22664
3	0.35383	1.15172	0.64617	0.39942	**0.40751**	0.25809
4	0.46886	1.03712	0.53114	0.59356	**0.48626**	0.31526
5	0.60611	1	0.39389	1	**0.60611**	0.39389

Table 2: $\mu = 7$. Top: Output from the original (unnormalized) transformed decoder based on the transformations (51); the optimal path is $(1, 2, 1, 1, 1)$. Bottom: Output from the normalized transformed decoder based on the transformations (54); the optimal path is $(1, 1, 1, 1, 1)$.

3. Also, the choice of (51) does not work in practice except with trivially short sequences; the underlying transformations can be normalized but this alters the decoder (Proposition 13). The choice of (45) is better in several aspects, mainly for its rescaling property (subsection 6.2), that is, the decoder is indeed ready to work in practice.

4. Algorithmically defined estimators are notoriously hard to analyze analytically (Winkler, 2003, pp. 25, 129-131). Indeed, it is not clear if the general members of the above interpolating families (regardless of the transformation used) satisfy any explicit optimality criteria; this makes it difficult to interpret such decoders. This may also discourage the use of such decoders in more complex inference cycles (that is, when any genuine model parameters are to be estimated as well, for example, as in Viterbi Training Koski, 2001; Lember and Koloydenko, 2008, 2010).

5. The point-wise hybridization scheme (49) can itself be altered. Indeed, other recursion schemes (see, for example, Koski, 2001, pp. 272-273 for Derin's formula) can also be applied for this purpose. However, now more than a decade after Brushe et al. (1998), we are not aware of any practical application of the idea of algorithmic hybridization of the MAP-PMAP inferences. Besides the plausible reasons already discussed in Subsection 1.2.1 (that actually extend to any type of MAP-PMAP hybridization), it is plausible that this particular type of hybridization has not yet seen application because of the lack of interpretation of its solutions, and possibly also because of the aforementioned difficulties with implementation of the original idea of Brushe et al. (1998).[3]

3. We recently attempted to contact the authors of that paper, but have not received any response by the time of sending this manuscript to the production editor.

Appendix D gives a pseudo-code to compute a decoded sequence $\widehat{y^T}(\mu)$ for any $\mu > 0$ using the power-transform approach (49) with scaling. Naturally, the decoding process can be parallelized over a range of μ values.

7. Asymptotic Risks

Given an arbitrary decoder g and a risk function R, the quantity $R(g(x^T) \mid x^T)$ evaluates the risk when g is applied to a given sequence x^T. Below we will write $R(x^T)$ for the minimum risk $\min_{s^T} R(s^T \mid x^T)$ which is achieved by the Bayes decoder v: $R(v(x^T) \mid x^T) = R(x^T)$. Besides $R(X^T)$, we are also interested in the random variables $R(g(X^T) \mid X^T)$ (depending on R and g). Thus, Kuljus and Lember (2012) have considered convergence of various risks of the Viterbi decoder $v(\cdot;\infty)$. Since Viterbi paths $v(x^T;\infty)$ and $v(x^{T+1};\infty)$ may differ significantly, asymptotic analysis of the Viterbi decoding is far from being trivial. Koloydenko and Lember (2008); Lember and Koloydenko (2008, 2010) constructed a well-defined process $v(X^\infty;\infty)$, named also after Viterbi, that for a wide class of HMMs extends *ad infinitum* finite Viterbi paths $v(x^T;\infty)$ and possesses useful ergodic properties. Based on the asymptotic theory of Viterbi processes $v(X^\infty;\infty)$, Kuljus and Lember (2012) have shown that under fairly general assumptions on the HMM, the random variables $R_k(v(X^T;\infty) \mid X^T)$, $\bar{R}_k(v(X^T;\infty) \mid X^T)$, where $k = 1, 2, \ldots$, and $\bar{R}_\infty(v(X^T;\infty) \mid X^T)$, as well as $\bar{R}_\infty(v(X^T;\infty))$ (see Equation 12), $\bar{R}_1(v(X^T;\infty))$ (see Equation 20), and $R_1(v(X^T;\infty))$ (see Equation 27) all converge (as $T \to \infty$) *a.s.* to constant (that is non-random) limits. Convergence of these risks implies *a.s.* convergence of

$$C_1\bar{R}_1(v(X^T;\infty) \mid X^T) + C_2\bar{R}_\infty(v(X^T;\infty) \mid X^T) + C_3\bar{R}_1(v(X^T;\infty)) + C_4\bar{R}_\infty(v(X^T;\infty)),$$

and

$$C_1R_1(v(X^T;\infty) \mid X^T) + C_2\bar{R}_\infty(v(X^T;\infty) \mid X^T) + C_3R_1(v(X^T;\infty)) + C_4\bar{R}_\infty(v(X^T;\infty)),$$

the risks appearing in the generalized problems (18) and (26), respectively. Actually, convergence of $\bar{R}_\infty(v(X^T;\infty), X^T)$ is also proved (and used in the proof of convergence of $\bar{R}_\infty(v(X^T;\infty) \mid X^T)$). Hence, the minimized risk in (19), evaluated at the Viterbi paths, converges as well.

The limits—*asymptotic risks*—are (deterministic) constants that depend only on the model, and help us assess the Viterbi inference in the following principled way. For example, let $R_1(k = \infty)$ be the limit (as $T \to \infty$) of $R_1(v(X^T;\infty) \mid X^T)$, which is the asymptotic misclassification rate of the Viterbi decoding. Thus, for large T, the Viterbi decoding makes about $TR_1(k = \infty)$ misclassification errors. The asymptotic risks might be, in principle, found theoretically, but in reality this can be rather difficult. However, since all these asymptotic results also hold in the L_1 sense, which implies convergences of expectations, the limiting risks can be estimated by simulations.

Lember (2011a,b) has also shown that under the same assumptions $R_1(X^T) = R_1(v(X^T;1) \mid X^T)$ converges to a constant limit, say R_1. Kuljus and Lember (2012) have at the same time also shown $\bar{R}_1(X^T) = \bar{R}_1(v(X^T;1) \mid X^T)$ to converge. Clearly $R_1(k = \infty) \geq R_1(1)$, and even if their difference is small, the total number of errors made by the Viterbi decoder in excess of PMAP in the long run can still be significant.

Presently, we are not aware of a universal method for proving (or improving upon) the limit theorems for these risks. Recall that convergence of the risks of the Viterbi decoding is possible due to the existence of the Viterbi process which has nice ergodic properties. The question whether infinite PMAP processes have similar properties, is still open. Therefore, convergence of $R_1(X^T)$ was proven with a completely different method based on the smoothing probabilities. In fact, all of the limit theorems obtained thus far have been proven with different methods. We conjecture that these different methods can be combined so that convergence of the minimized combined risk (18) or (26) could be proven as well. In summary, as mentioned before, convergence of the minimized combined risks has thus far been obtained for trivial combinations only, that is with three of the four constants being zero. Note that while convergence of the intermediate case (38) with its minimizer $v(x^T; k(\alpha))$ is an open question, (39) gives

$$0 \leq \bar{R}_\infty(v(x^T; k(\alpha)) \mid x^T) - \bar{R}_\infty(v(x^T; \infty) \mid x^T) \leq \frac{\bar{R}_1(v(x^T; \infty) \mid x^T)}{k-1}.$$

This, together with the *a.s.* convergence of $\bar{R}_1(v(X^T; \infty) \mid X^T)$, implies that in the long run, for most sequences x^T, $\bar{R}_\infty(v(x^T; k) \mid x^T)$ will not exceed $\bar{R}_\infty(v(x^T; \infty) \mid x^T)$ by more than $\frac{1}{k-1} \lim_{T\to\infty} \bar{R}_1(v(X^T; \infty) \mid X^T)$. Since this limit is finite, letting k increase with T, we get that $\bar{R}_\infty(v(X^T; k_T))$ approach $\lim_{T\to\infty} \bar{R}_\infty(v(X^T; \infty))$ *a.s.*, that is, as the intuition predicts, the likelihood of $v(X^T; k_T)$ approaches that of $v(X^T; \infty)$.

Finally, Lember and Koloydenko (2010); Lember et al. (2011) also outline possible applications of the above asymptotic risk theory. For example, if a certain number of the true labels $y_1, y_2, \ldots, y_T$ can be revealed (say, at some cost), the remaining labels would be computed by a constrained decoder, for example, the constrained Viterbi decoder. Having observed x^T, the user then needs to decide which positions are "most informative" and then acquires their labels. Assuming further that the HMM is stationary, the R_1-like risks $P(v(X^\infty; \infty)_t \neq Y_t \mid X_{t-m}^{t+m} \in A)$ (for any $m \geq 1$ and any measurable set $A \in \mathcal{X}^{2m+1}$), are independent of t (for $t = m+1, m+2, \ldots$), and could therefore be used in the above active learning protocol for the selection of the most informative positions. Specifically, if A is such that $P(v(X^\infty; \infty)_t \neq Y_t \mid X_{t-m}^{t+m} \in A)$ is high, then acquire labels at positions t of occurrence of A. Naturally, there are different ways to make this concrete. For one simple example, suppose only a batch of L labels can be acquired. Assuming $\mathcal{X}$ to be discrete, order all the $\mathcal{X}$ words A of length q (that is, $A \in \mathcal{X}^q$) by $P(v(X^\infty; \infty)_t \neq Y_t \mid X_{t-m}^{t+m} \in A)$. Finally, from the $\mathcal{X}$ of length q that occur in x^T, choose L with the highest $P(v(X^\infty; \infty)_t \neq Y_t \mid X_{t-m}^{t+m} \in A)$. The above asymptotic theory is crucial also for establishing $P(v(X^\infty; \infty)_t \neq Y_t \mid X_{t-m}^{t+m} \in A)$ as the *a.s.* limit of easily computable (e.g., via off-line simulations) empirical measures. In practice, these latter measures would be used as estimates of $P(v(X^\infty; \infty)_t \neq Y_t \mid X_{t-m}^{t+m} \in A)$ and first experiments along these lines are given by Lember et al. (2011, Section 4.4). It may also be of interest to test these ideas with other risks and decoders, such as members of the generalized hybrid families presented here.

8. Discussion

The point-wise symmetric zero-one loss $l(s,y) = \mathbb{I}_{\{s\neq y\}}$ in (4), (5), and consequently in the generalized PMAP hybrid decoding (26), can be easily replaced by a general loss $l(s,y) \geq 0$, $s, y \in S$. In computational terms, this would require multiplying the loss matrix $(l(s,y))_{s,y\in S}$ by the (prior or) posterior probability vectors $(p_t(1 \mid x^T), p_t(2 \mid x^T), \ldots, p_t(K \mid x^T))'$ to obtain the (prior or) posterior risk $(\rho_t(1 \mid x^T), \rho_t(2 \mid x^T), \ldots, \rho_t(K \mid x^T))'$ vectors (we use the apostrophe to denote vector transpose). The dynamic programming algorithm defined by (23) with (28) still stands provided $p_t(j \mid x^T)$ (or $p_t(j)$, or both) is replaced by $1 - \rho_t(j \mid x^T)$ (or $1 - \rho_t(j)$, or both respectively) in the definition of $\gamma_t(j)$. If all confusions of state y are equally undesirable, that is, $l(s,y)$ is of the form $l(y) \times \mathbb{I}_{\{s\neq y\}}$, then the above adjustment reduces to replacing $p_t(j \mid x^T)$ by $l(j)p_t(j \mid x^T)$ (for all $j \in S$), which we illustrated in Figure 2 when suppressing state 3. Similar adjustments can be made to the $\bar{R}_1$ risks of the generalized PVD family, which was also illustrated in Figure 2.

Using an asymmetric loss could be particularly valuable in practice when, for example, detection of a rare state or transition needs to be encouraged. Similar views have been most recently expressed also by Yau and Holmes (2010), who, staying within the additive risk framework, have proposed a general asymmetric form of the loss (30) with $k = 2$. Hybridizing this general asymmetric pairwise loss with the other losses considered in this work should provide additional flexibility to path inference. A way to incorporate this loss into our generalized framework is by vectorizing the chain $\{Y_t\}_{t\geq 1}$ as $\{(Y_t, Y_{t+1})\}_{t\geq 1}$ and then following the opening lines of this Section.

Also, using a range of perturbed versions of a loss function can help assess saliency of particular detections ("islands"). In fact, at the stage of data exploration one may more generally want to use a collection of outputs produced by using a range of different loss functions instead of a single one.

The logarithmic risks (3), (6), (12), (20) on the one hand, and the ordinary risks (2), (5), $R_\infty(s^T) = 1 - p(s^T)$, (27), on the other hand, can be respectively combined into a single parameter family of risks by using, for example, the power transformation as shown below with p for the moment standing for any probability distribution on S^T:

$$\begin{aligned} R_1(s^T;\beta) &= \begin{cases} -\frac{1}{T}\sum_{t=1}^T \frac{p_t(s_t)^\beta - 1}{\beta}, & \text{if } \beta \neq 0; \\ -\frac{1}{T}\sum_{t=1}^T \log p_t(s_t), & \text{if } \beta = 0; \end{cases} \\ R_\infty(s^T;\beta) &= \begin{cases} -\frac{1}{T}\frac{p(s^T)^\beta - 1}{\beta}, & \text{if } \beta \neq 0; \\ -\frac{1}{T}\log p(s^T), & \text{if } \beta = 0. \end{cases} \end{aligned} \tag{55}$$

Thus, the family of risk minimization problems given in (56) below

$$\min_{s^T} \Big[C_1 R_1(s^T \mid x^T;\beta_1) + C_2 R_\infty(s^T \mid x^T;\beta_2) + C_3 R_1(s^T;\beta_3) + C_4 R_\infty(s^T;\beta_4) \Big], \tag{56}$$

$C_i \geq 0$ and $\sum_{i=1}^4 C_i > 0$ unifies and generalizes problem (18) ($\beta_1 = \beta_2 = \beta_3 = \beta_4 = 0$) and problem (26) ($\beta_1 = \beta_3 = 1$, $\beta_2 = \beta_4 = 0$). Clearly, the dynamic programming approach of Theorem 4 immediately applies to any member of the above family (56) with $\beta_2 = \beta_4 = 0$. Also, computations of multiple decoders from this family (at least with $\beta_2 = \beta_4 = 0$) are readily parallelizable.

Next, Theorem 6 and Corollaries 7 and 8 obviously generalize to higher order Markov chains as can be seen from the following Proposition.

Proposition 14 *Let p represent a Markov chain of order m, $1 \leq m \leq T$, on S^T. Then for any $s^T \in S^T$ and for any $k \in \{m, m+1, \ldots\}$, we have*

$$\bar{R}_k(s^T) = \bar{R}_m(s^T) + (k-m)\bar{R}_\infty(s^T).$$

Proof This is a straightforward extension of the proof of Theorem 6. ■

The present risk-based discussion of HMM path inference also naturally extends to the problem of optimal *labeling* or *annotation* (already mentioned in Subsection 1.2). Namely, the state space S can be partitioned into subsets S_1, S_2, ..., S_Λ, for some $\Lambda \leq K$, in which case $\lambda(s)$ assigns label λ to every state $s \in S_\lambda$. The fact that the PMAP problem is as easily solved over the label space Λ^T as it is over S^T has already been used in practice. Indeed, Käll et al. (2005), who also add the constraint of admissibility with respect to the prior distribution, in effect average $p_t(s_t \mid x^T)$'s, for each t, within the label classes and then use recursions (15) to obtain the *optimal accuracy labeling* of *a priori* admissible state paths. This clearly corresponds to using the point loss $l(s, s') = \mathbb{I}_{\{\lambda(s) \neq \lambda(s')\}}$ in (4) when solving $\min_{s^T : p(s^T) > 0} R_1(s^T \mid x^T)$ (14). With our definition of admissibility (that is, positivity of the posterior path probability), the same approach (that is, replacing $p_t(s_t \mid x^T)$'s by their within class average $\bar{p}_t(s_t \mid x^T)$) extends to solve $\min_{s^T : p(s^T|x^T) > 0} R_1(s^T \mid x^T)$ (7) under the same loss $l(s, s') = \mathbb{I}_{\{\lambda(s) \neq \lambda(s')\}}$. Clearly, the generalized problem (56) also immediately incorporates the above pointwise label-level loss in either the prior $R_1(\cdot; \beta_3)$ or posterior risk $R_1(\cdot; \beta_1)$, or both. Since computationally these problems are essentially as light as recursion (24), (25), and since Käll et al. (2005) report their special case to be successful in practice, we believe that the above generalizations offer yet more possibilities that are potentially useful in practice.

Instead of using the same arithmetic averages $\bar{p}_t(s_t \mid x^T)$'s (or $\bar{p}_t(s_t)$'s) for the R_1 risks in (56) regardless of β, we can gain additional flexibility by replacing $\bar{p}_t(s_t)^\beta$ and $\log \bar{p}_t(s_t)$ in (55) ($\beta \neq 0$ and $\beta = 0$ respectively) with

$$\bar{p}_t(s; \beta) \quad \propto \quad \begin{cases} \left(\dfrac{\sum\limits_{s' \in S_{\lambda(s)}} p_t(s')}{|S_{\lambda(s)}|} \right)^\beta, & \text{if } \beta \neq 0; \\ \left(\prod\limits_{s' \in S_{\lambda(s)}} p_t(s') \right)^{\frac{1}{|S_{\lambda(s)}|}}, & \text{if } \beta = 0. \end{cases}$$

Certainly, the choice of the basic loss functions, inflection parameters β_i and weights C_i of the respective risks is application dependent, and can be tuned with the help of labeled data, using, for example, cross-validation.

Finally, these generalizations are presented for the standard HMM setting, and therefore extensions to more complex and practically more useful HMM-based settings (e.g., semi-Markov, autoregressive, coupled, etc.) could also be interesting.

Since the transform based approach, especially the newly proposed power-transform hybridization, has also generated some interest, it would be interesting to evaluate performance of the power-transform hybrids together with the risk-based families on multiple real applications and using various domain specific performance measures.

Acknowledgments

The first author has been supported by the Estonian Science Foundation Grant nr. 9288 and by targeted financing project SF0180015s12, which has also supported a research visit of the second author to Tartu University. The second author has also been supported by UK NIHR Grant i4i II-AR-0209-10012. The authors are also grateful to anonymous reviewers as well as to the action editor for their thorough reviews of this work, additional references, and comments and suggestions on improving this manuscript. The authors are also very thankful to Dr Dario Gasbarra and Dr Kristi Kuljus for reviewing earlier versions of the manuscript and pointing out two subtle mistakes, as well as to Ufuk Mat for pointing out some typing errors.

Appendix A. An Example of an Inadmissible Path of Positive Prior Probability

$$\pi = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}/9, \qquad P = \begin{pmatrix} 5 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 4 & 0 & 5 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 2 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 3 & 3 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 3 & 3 & 0 & 0 & 0 & 3 & 0 \end{pmatrix}/9.$$

To simplify the verifications, consider an emission alphabet with only four symbols, although the idea of constructing this example readily extends to larger alphabets (in particular, to more practically relevant situations where the emission alphabet is larger than the hidden state space, or the emission distributions are continuous altogether). Then take the following emission distributions:

$$\begin{pmatrix} P_1 & P_2 & P_3 & P_4 \\ 1/25 & 1/20 & 0 & 91/100 \\ 0 & 0 & 1/5 & 4/5 \\ 1/20 & 1/25 & 0 & 91/100 \\ 0 & 0 & 1/5 & 4/5 \\ 1/10 & 0 & 1/5 & 7/10 \\ 0 & 0 & 1/5 & 4/5 \\ 1/15 & 1/15 & 0 & 13/15 \\ 0 & 0 & 1/5 & 4/5 \\ 1/15 & 1/15 & 0 & 13/15 \end{pmatrix}.$$

Suppose now that a sequence $x^3 = (1, 2, 3)$ has been observed. It can then be verified that the (unconstrained) PMAP decoder returns any of the following paths $(5, 1, 5)$, $(5, 3, 5)$, $(5, 7, 5)$, or $(5, 9, 5)$, all of which having zero prior (and posterior) probabilities.

When the decoder is subject to the positivity constraint on the prior probabilities, it would return any of the following paths $(5, 2, 5)$, $(5, 4, 5)$, $(5, 5, 5)$, $(5, 6, 5)$, $(5, 8, 5)$, which, despite being of positive prior probabilities, all have zero posterior probabilities.

Finally, if the decoder is constrained to produce paths of positive posterior probability, it would then return any of the following paths $(5, 7, 2)$, $(5, 7, 6)$, $(3, 3, 5)$, $(9, 3, 5)$.

Appendix B. Proof of Remark 3

Proof Assume $C_3 = C_4 = 0$. For each $C_1, C_2 > 0$, let $\widehat{y^T}_{C_1,C_2} \in S^T$ be a solution to (18), and let $\widehat{y^T}_{PVD}$ be the output of PVD. Thus, we have

$$C_1\bar{R}_1(\widehat{y^T}_{C_1,C_2} \mid x^T) + C_2\bar{R}_\infty(\widehat{y^T}_{C_1,C_2} \mid x^T) \quad \leq \quad C_1\bar{R}_1(\widehat{y^T}_{PVD} \mid x^T) + C_2\bar{R}_\infty(\widehat{y^T}_{PVD} \mid x^T).$$

Then

$$0 \leq C_1(\bar{R}_1(\widehat{y^T}_{C_1,C_2} \mid x^T) - \bar{R}_1(\widehat{y^T}_{PVD} \mid x^T)) \leq C_2(\bar{R}_\infty(\widehat{y^T}_{PVD} \mid x^T) - \bar{R}_\infty(\widehat{y^T}_{C_1,C_2} \mid x^T))$$

holds for any $C_1, C_2 > 0$. Since $\bar{R}_\infty(\widehat{y^T}_{PVD} \mid x^T) - \bar{R}_\infty(\widehat{y^T}_{C_1,C_2} \mid x^T)$ is clearly bounded (and S^T is finite), we obtain $\bar{R}_1(\widehat{y^T}_{C_1,C_2} \mid x^T) = \bar{R}_1(\widehat{y^T}_{PVD} \mid x^T)$ for some sufficiently small C_2. Since $C_2 > 0$, all $\widehat{y^T}_{C_1,C_2}$ are admissible (Remark 1 above), therefore for such sufficiently small C_2, $\widehat{y^T}_{C_1,C_2}$ is also a solution to the PVD Problem (9).

The second statement is proved similarly, recalling Proposition 2 to establish admissibility of $\widehat{y^T}_{C_1,C_4}$ *almost surely.* ■

Appendix C. Supplementary Results on the Trade-Off between $\bar{R}_1$ and $\bar{R}_\infty$ Risks in Problem (18), and between R_1 and $\bar{R}_\infty$ Risks in Problem (26).

Corollary 15 *1. Let $\hat{y}$ and $\hat{y}'$ be solutions to Problem (18) with $C_1 \in [0, 1]$ and $C_2 = 1 - C_1$, $C_3 = C_4 = 0$ and $C'_1 \in [0, 1]$ and $C'_2 = 1 - C'_1$, $C'_3 = C'_4 = 0$, respectively. Assume $C_1 \leq C'_1$. Then $\bar{R}_1(\hat{y} \mid x^T) \geq \bar{R}_1(\hat{y}' \mid x^T)$ and $\bar{R}_\infty(\hat{y} \mid x^T) \leq \bar{R}_\infty(\hat{y}' \mid x^T)$.*

2. Let $\hat{y}$ and $\hat{y}'$ be solutions to Problem (18) with $C_3 \in [0, 1]$ and $C_4 = 1-C_3$, $C_1 = C_2 = 0$ and $C'_3 \in [0, 1]$ and $C'_4 = 1 - C'_3$, $C'_1 = C'_2 = 0$, respectively. Assume $C_3 \leq C'_3$. Then $\bar{R}_1(\hat{y}) \geq \bar{R}_1(\hat{y}')$ and $\bar{R}_\infty(\hat{y}) \leq \bar{R}_\infty(\hat{y}')$.

3. Let $\hat{y}$ and $\hat{y}'$ be solutions to Problem (26) with $C_1 \in [0, 1]$ and $C_2 = 1-C_1$, $C_3 = C_4 = 0$ and $C'_1 \in [0, 1]$ and $C'_2 = 1 - C'_1$, $C'_3 = C'_4 = 0$, respectively. Assume $C_1 \leq C'_1$. Then $R_1(\hat{y} \mid x^T) \geq R_1(\hat{y}' \mid x^T)$ and $\bar{R}_\infty(\hat{y} \mid x^T) \leq \bar{R}_\infty(\hat{y}' \mid x^T)$.

4. Let $\hat{y}$ and $\hat{y}'$ be solutions to Problem (26) with $C_3 \in [0, 1]$ and $C_4 = 1-C_3$, $C_1 = C_2 = 0$ and $C'_3 \in [0, 1]$ and $C'_4 = 1-C'_3$, $C'_1 = C'_2 = 0$. Assume $C_3 \leq C'_3$. Then $R_1(\hat{y}) \geq R_1(\hat{y}')$ and $\bar{R}_\infty(\hat{y}) \leq \bar{R}_\infty(\hat{y}')$.

Proof A straightforward application of Lemma 16 given below. ■

Lemma 16 *Let F and G be functions from a set A to the extended reals $\bar{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$. Let $\alpha_1, \alpha_2 \in [0,1]$ be such that $\alpha_1 \le \alpha_2$. Suppose $a_1, a_2 \in A$ are such that*

$$\alpha_i F(a_i) + (1-\alpha_i)G(a_i) \quad \le \quad \alpha_i F(x) + (1-\alpha_i)G(x), \quad i = 1,2, \quad \textit{for all } x \in A.$$

Then $F(a_1) \ge F(a_2)$ and $G(a_1) \le G(a_2)$.

Although the result is obvious, below we state its proof for completeness.

Proof Write a, b, c, and d for $F(a_1)$, $G(a_1)$, $F(a_2)$, and $G(a_2)$, respectively. Then we have

$$\begin{aligned} \alpha_1(a-c) &\le (1-\alpha_1)(d-b), \\ \alpha_2(a-c) &\ge (1-\alpha_2)(d-b), \end{aligned}$$

and therefore

$$\begin{aligned} \alpha_2\alpha_1(a-c) &\le \alpha_2(1-\alpha_1)(d-b), \\ \alpha_1\alpha_2(a-c) &\ge \alpha_1(1-\alpha_2)(d-b), \end{aligned}$$

which gives $\alpha_1(1-\alpha_2)(d-b) \le \alpha_2(1-\alpha_1)(d-b)$. Since $\alpha_1(1-\alpha_2) \le \alpha_2(1-\alpha_1)$, it follows that $d \ge b$, that is, $G(a_2) \ge G(a_1)$. The fact that $F(a_1) \ge F(a_2)$ is obtained similarly. ■

Appendix D. Pseudo-Code for Computing the Hybrid Decoders (49) Using the Power-Transform with Scaling (52), (53).

Finally, to output the decoded sequence $\widehat{y^T}(\mu)$, a simple tie-breaking rule may be as follows:

for $t = 1, 2, \ldots, T$ **do**
  $\hat{y}_t(\mu) \leftarrow \min \arg\max\{\tilde{\alpha}_t(i;\mu)\tilde{\beta}_t(i;\mu)\}$,
end for

whereas more elaborate rules may involve ordering of the entire state space S^T, or simply outputting all of the winning sequences. (Computations of the transformed and scaled α and β variables are summarized in Algorithms 1 and 2 respectively.)

Algorithm 1 The forward pass to compute $\tilde{\alpha}_t(i;\mu)$ and the scaling constants $c_t(\mu)$.

for $t = 1, 2, \ldots, T$ **do**
  $c_t(\mu) \leftarrow 0$
end for
for $i = 1, 2, \ldots, K$ **do**
  $\alpha_1(i) \leftarrow \pi_i f_i(x_1)$
  $c_1(\mu) \leftarrow c_1(\mu) + \pi_i f_i(x_1)$
end for
for $i = 1, 2, \ldots, K$ **do**
  $\tilde{\alpha}_1(i;\mu) \leftarrow \alpha_1(i)/c_1(\mu)$
end for
if $\mu = 0$ **then**
  for $t = 2, \ldots, T$ **do**
    for $i = 1, 2, \ldots, K$ **do**
      $S_t(i) \leftarrow \{j \in S : \ \tilde{\alpha}_{t-1}(j;\mu)p_{ji} > 0\}$
      $K_t(i) \leftarrow |S_t(i)|$
      $\tilde{\alpha}_t(i;\mu) \leftarrow \left[\prod_{j \in S_t(i)} \tilde{\alpha}_{t-1}(j;\mu)p_{ji}\right]^{\frac{1}{K_t(i)}} f_i(x_t)$
      $c_t(\mu) \leftarrow c_t(\mu) + \tilde{\alpha}_t(i;\mu)$
    end for
    for $i = 1, 2, \ldots, K$ **do**
      $\tilde{\alpha}_t(i;\mu) \leftarrow \tilde{\alpha}_t(i;\mu)/c_t(\mu)$
    end for
  end for
else
  for $t = 2, \ldots, T$ **do**
    for $i = 1, 2, \ldots, K$ **do**
      $\tilde{\alpha}_t(i;\mu) \leftarrow \left[\sum_{j=1}^{K} (\tilde{\alpha}_{t-1}(j;\mu)p_{ji})^{\mu}\right]^{\frac{1}{\mu}} f_i(x_t)$
      $c_t(\mu) \leftarrow c_t(\mu) + \tilde{\alpha}_t(i;\mu)$
    end for
    for $i = 1, 2, \ldots, K$ **do**
      $\tilde{\alpha}_t(i;\mu) \leftarrow \tilde{\alpha}_t(i;\mu)/c_t(\mu)$
    end for
  end for
end if

Algorithm 2 The backward pass to compute $\tilde{\beta}_t(i;\mu)$.

for $i = 1, 2, \ldots, K$ **do**
  $\tilde{\beta}_T(i;\mu) \leftarrow 1$
end for
if $\mu = 0$ **then**
  for $t = T-1, T-2, \ldots, 1$ **do**
    for $i = 1, 2, \ldots, K$ **do**
      $S_t^*(i) \leftarrow \{j \in S : \ f_j(x_{t+1}) p_{ij} \tilde{\beta}_{t+1}(j;\mu) > 0\}$
      $K_t^*(i) \leftarrow |S_t^*(i)|$
      $$\tilde{\beta}_t(i;\mu) \leftarrow \left[\prod_{j \in S_t^*(i)} f_j(x_{t+1}) p_{ij} \tilde{\beta}_{t+1}(j;\mu) \right]^{\frac{1}{K_t^*(i)}} / c_{t+1}(\mu)$$
    end for
  end for
else
  for $t = T-1, T-2, \ldots, 1$ **do**
    for $i = 1, 2, \ldots, K$ **do**
      $$\tilde{\beta}_t(i;\mu) \leftarrow \left[\sum_{j=1}^{K} \left(f_j(x_{t+1}) p_{ij} \tilde{\beta}_{t+1}(j;\mu) \right)^{\mu} \right]^{\frac{1}{\mu}} / c_{t+1}(\mu)$$
    end for
  end for
end if

Appendix E. Further Details of the Experiments from Section 5

Below are the estimates of the HMM parameters obtained from the entire data set as described in Section 5.

$$\hat{\pi} = \begin{pmatrix} 0.0016 & 0.0041 & 0.9929 & 0.0014 & 0.0000 & 0.0000 \end{pmatrix},$$

$$\widehat{\mathbb{P}} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{pmatrix} 0.8359 & 0.0034 & 0.1606 & 0 & 0 & 0 \\ 0.0022 & 0.8282 & 0.1668 & 0.0028 & 0 & 0 \\ 0.0175 & 0.0763 & 0.8607 & 0.0455 & 0 & 0 \\ 0 & 0 & 0 & 0.7500 & 0.2271 & 0.0229 \\ 0 & 0 & 0 & 0 & 0.8450 & 0.1550 \\ 0 & 0.0018 & 0.2481 & 0 & 0 & 0.7501 \end{pmatrix},$$

$$\hat{\pi}_{inv} = \begin{pmatrix} 0.0511 & 0.2029 & 0.4527 & 0.0847 & 0.1240 & 0.0847 \end{pmatrix},$$

	$\widehat{P_1}$	$\widehat{P_2}$	$\widehat{P_3}$	$\widehat{P_4}$	$\widehat{P_5}$	$\widehat{P_6}$
A	0.1059	0.0636	0.0643	0.1036	0.1230	0.1230
C	0.0107	0.0171	0.0135	0.0081	0.0111	0.0128
D	0.0538	0.0319	0.0775	0.0634	0.0415	0.0345
E	0.0973	0.0477	0.0620	0.1120	0.0852	0.0848
F	0.0436	0.0576	0.0330	0.0371	0.0386	0.0399
G	0.0303	0.0484	0.1133	0.0447	0.0321	0.0229
H	0.0203	0.0227	0.0259	0.0188	0.0197	0.0221
I	0.0564	0.1010	0.0372	0.0557	0.0694	0.0593
K	0.0672	0.0443	0.0574	0.0560	0.0671	0.0810
L	0.1227	0.1068	0.0674	0.0994	0.1279	0.1477
M	0.0240	0.0219	0.0181	0.0214	0.0293	0.0304
N	0.0299	0.0252	0.0561	0.0259	0.0338	0.0336
P	0.0333	0.0208	0.0757	0.0472	0.0067	0.0031
Q	0.0443	0.0270	0.0330	0.0469	0.0497	0.0472
R	0.0594	0.0464	0.0470	0.0522	0.0677	0.0697
S	0.0496	0.0496	0.0744	0.0485	0.0422	0.0491
T	0.0395	0.0641	0.0572	0.0465	0.0412	0.0375
V	0.0591	0.1386	0.0473	0.0685	0.0677	0.0545
W	0.0168	0.0172	0.0111	0.0135	0.0130	0.0124
Y	0.0359	0.0483	0.0286	0.0306	0.0332	0.0344

.

References

Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403 – 410, 1990.

Zafer Aydin, Yucel Altunbasak, and Mark Borodovsky. Protein secondary structure prediction for a single-sequence using hidden semi-Markov models. *BMC Bioinformatics*, 7 (1):178, 2006.

Lalit R. Bahl, John Cocke, Frederick Jelinek, and Josef Raviv. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). *IEEE Transactions on Information Theory*, 20(2):284–287, 1974.

Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B. Methodological*, 48(3):259–302, 1986.

Julian Besag and Peter J. Green. Spatial statistics and Bayesian computation. *Journal of the Royal Statistical Society. Series B. Methodological*, 55(1):25–37, 1993.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, 2006.

Matthew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden Markov models for complex action recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 994–999, S.Juan, Puerto Rico, 1997.

Broňa Brejová, Daniel G. Brown, and Tomáš Vinař. The most probable annotation problem in hmms and its application to bioinformatics. *Journal of Computer and System Sciences*, 73(7):1060 – 1077, 2007a.

Broňa Brejová, Daniel G. Brown, and Tomáš Vinař. Advances in hidden Markov models for sequence annotation. In Ion I. Măndoiu and Alexander Zelikovski, editors, *Bioinformatics Algorithms: Techniques and Applications*, pages 55–92. John Wiley & Sons, Inc., 2007b.

Gary D. Brushe, Robert E. Mahony, and John B. Moore. A soft output hybrid algorithm for ML/MAP sequence estimation. *IEEE Transactions on Information Theory*, 44(7): 3129–3140, 1998.

Chris Burge and Samuel Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268(1):78 – 94, 1997.

Olivier Cappé, Eric Moulines, and Tobias Rydén. *Inference in Hidden Markov Models*. Springer Series in Statistics. Springer, New York, 2005.

Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46 (2):255–308, 2009.

Luis E. Carvalho and Charles E. Lawrence. Centroid estimation in discrete high-dimensional spaces with applications in biology. *Proceedings of the National Academy of Sciences of the United States of America*, 105(9):3209–3214, 2008.

Christiane Cocozza-Thivent and Abdelkrim Bekkhoucha. Estimation in Pickard random fields and application to image processing. *Pattern Recognition*, 26(5):747–761, 1993.

Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

Sean Eddy. What is a hidden Markov model? *Nature Biotechnology*, 22(10):1315 – 1316, 2004.

Yariv Ephraim and Neri Merhav. Hidden Markov processes. *IEEE Transactions on Information Theory*, 48(6):1518–1569, June 2002.

Piero Fariselli, Pier Martelli, and Rita Casadio. A new decoding algorithm for hidden Markov models improves the prediction of the topology of all-beta membrane proteins. *BMC Bioinformatics*, 6(Suppl 4):S12, 2005.

Kuzman Ganchev, João V. Graça, and Ben Taskar. Better alignments = better translations? In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 986–993, Columbus, Ohio, 2008.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

Peter J. Green and Sylvia Richardson. Hidden Markov models and disease mapping. *Journal of the American Statistical Association*, 97(460):1055–1070, 2002.

Jeremiah F. Hayes, Thomas M. Cover, and Juan B. Riera. Optimal sequence detection and optimal symbol-by-symbol detection: similar algorithms. *IEEE Transactions on Communications*, 30(1):152–157, January 1982.

Ian Holmes and Richard Durbin. Dynamic programming alignment accuracy. *Journal of Computational Biology*, 5(3):493–504, 1998.

Xuedong Huang, Yasuo. Ariki, and Mervyn Jack. *Hidden Markov Models for Speech Recognition*. Edinburgh University Press, Edinburgh, UK, 1990.

Frederick Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64:532–556, April 1976.

Frederick Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts, 2001.

Dhiraj Joshi, Jia Li, and James Z. Wang. A computationally efficient approach to the estimation of two- and three-dimensional hidden Markov models. *IEEE Transactions on Image Processing*, 15(7):1871–1886, 2006.

Lukas Käll, Anders Krogh, and Erik L. L. Sonnhammer. An HMM posterior decoder for sequence feature prediction that includes homology information. *Bioinformatics*, 21 (suppl_1):i251–257, 2005.

Alexey A. Koloydenko and Jüri Lember. Infinite Viterbi alignments in the two state hidden Markov models. *Acta et Commentationes Universitatis Tartuensis de Mathematica*, (12): 109–124, 2008.

Timo Koski. *Hidden Markov Models for Bioinformatics*, volume 2 of *Computational Biology Series*. Kluwer Academic Publishers, Dordrecht, 2001.

Anders Krogh. Two methods for improving performance of an HMM and their application for gene finding. In *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, pages 179–186, Halkidiki, Greece, 1997.

Anders Krogh. An Introduction to Hidden Markov Models for Biological Sequences. In David B.Searls Steven L. Salzberg and Simon Kasif, editors, *Computational Methods in Molecular Biology*. Elsevier Science, first edition, 1998.

Kristi Kuljus and Jüri Lember. Asymptotic risks of Viterbi segmentation. *Stochastic Processes and Their Applications*, 122(9):3312–3341, 2012.

Hans Künsch, Stuart Geman, and Athanasios Kehagias. Hidden Markov random fields. *The Annals of Applied Probability*, 5(3):577–602, 1995.

Steffen L. Lauritzen. *Graphical models*, volume 17 of *Oxford Statistical Science Series*. Oxford University Press, New York, 1996.

Jüri Lember. On approximation of smoothing probabilities for hidden Markov models. *Statistics and Probability Letters*, 81(2):310–316, 2011a.

Jüri Lember. A correction on approximation of smoothing probabilities for hidden Markov models. *Statistics and Probability Letters*, 81(9):1463–1464, September 2011b.

Jüri Lember and Alexey A. Koloydenko. The Adjusted Viterbi training for hidden Markov models. *Bernoulli*, 14(1):180–206, 2008.

Jüri Lember and Alexey A. Koloydenko. A constructive proof of the existence of Viterbi processes. *IEEE Transactions on Information Theory*, 56(4):2017–2033, 2010.

Jüri Lember, Kristi Kuljus, and Alexey A. Koloydenko. Theory of segmentation. In Przemyslaw Dymarski, editor, *Hidden Markov Models, Theory and Applications*, Bioinformatics, pages 51–84. InTech, 2011.

Jia Li, Robert M. Gray, and Richard A. Olshen. Multiresolution image classification by hierarchical modeling with two-dimensional hidden Markov models. *IEEE Transactions on Information Theory*, 46(5):1826–1841, 2000.

Shu Lin and Daniel J. Costello Jr. *Error Control Coding: Fundamental and Applications*. Computer Applications in Electrical Engineering. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.

William H. Majoros and Uwe Ohler. Advancing the state of the art in computational gene prediction. In Sorin Istrail, Pavel Pevzner, and Michael Waterman, editors, *Knowledge Discovery and Emergent Complexity in Bioinformatics*, volume 4366 of *Lecture Notes in Computer Science*, pages 81–106. Springer Berlin / Heidelberg, 2007.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999.

Jose L. Marroquin, Edgar Arce Santana, and Salvador Botello. Hidden markov measure field models for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1380–1387, 2003.

Joshua Mason, Kathryn Watkins, Jason Eisner, and Adam Stubblefield. A natural language approach to automated cryptanalysis of two-time pads. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 235–244, Alexandria, Virginia, 2006.

MATLAB. *Version 7.13.0.564 (R2011b)*. The MathWorks, Inc., Natick, Massachusetts, 2011.

Erik McDermott and Timothy J. Hazen. Minimum classification error training of landmark models for real-time continuous speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Quebec, 2004.

Clare A. McGrory, D. Michael Titterington, Robert W. Reeves, and Anthony N. Pettitt. Variational Bayes for estimating the parameters of a hidden Potts model. *Statistics and Computing*, 19(3):329–340, 2009.

Hermann Ney, Volker Steinbiss, Reinhold Haeb-Umbach, B.-H. Tran, and Ute Essen. An overview of the Philips research system for large vocabulary continuous speech recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(1):33–70, 1994.

Mukund Padmanabhan and Michael A. Picheny. Large-vocabulary speech recognition algorithms. *Computer*, 35(4):42 – 50, 2002.

Lawrence Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1993.

Lawrence R. Rabiner, Jay G. Wilpon, and Biing-Hwang Juang. A segmental k-means training procedure for connected word recognition. *AT&T Technical Journal*, 65(3):21–31, 1986.

Patrick Robertson, Emmanuelle Villebrun, and Peter Hoeher. A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain. In *Proceedings of IEEE International Conference on Communications*, volume 2, pages 1009–1013, Seattle, Washington, 1995.

Havard Rue. New loss functions in Bayesian imaging. *Journal of the American Statistical Association*, 90(431):900–908, 1995.

Asaf A. Salamov and Victor V. Solovyev. Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments. *Journal of Molecular Biology*, 247(1):11 – 15, 1995.

Kengo Sato, Michiaki Hamada, Kiyoshi Asai, and Toutai Mituyama. Centroidfold: a web server for RNA secondary structure prediction. *Nucleic Acids Research*, 37(suppl 2): W277–W280, 2009.

Han Shu, I. Lee Hetherington, and James Glass. Baum-Welch training for segment-based speech recognition. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 43–48, St. Thomas, U. S. Virgin Islands, 2003.

Softberry, Inc. SSENVID: Protein secondary structure and environment assignment from atomic coordinates. `http://linux1.softberry.com/berry.phtml?topic=ssenvid&group=help&subgroup=propt`, 2001. Accessed: 15.10.2011.

Volker Steinbiss, Herman Ney, Xavier L. Aubert, Stefan Besling, Christian Dugast, Ute Essen, Daryl Geller, Reinhold Haeb-Umbach, Reinhard Kneser, Humberto G. Meier, Martin Oerder, and B.-H. Tran. The Philips research system for continuous-speech recognition. *Philips Journal of Research*, 49:317–352, 1995.

Nikko Ström, I. Lee Hetherington, Timothy J. Hazen, Eric Sandness, and James Glass. Acoustic modeling improvements in a segment-based speech recognizer. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 139–142, Keystone, Colorado, 1999.

The MathWorks, Inc. *Statistics Toolbox*$^{\text{TM}}$ *User's Guide*. Natick, Massachusetts, R2012a edition, 2012.

Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics*, volume 2, pages 836–841, Copenhagen, Denmark, 1996.

Gerhard Winkler. *Image Analysis, Random Fields and Markov chain Monte Carlo Methods*, volume 27 of *Applications of Mathematics (New York)*. Springer-Verlag, Berlin, second edition, 2003.

Christopher Yau and Chris C. Holmes. A decision theoretic approach for segmental classification using Hidden Markov models. *ArXiv e-prints*, 2010. URL `http://arxiv.org/abs/1007.4532`.

Journal of Machine Learning Research 15 (2014) 59-98 Submitted 3/12; Revised 5/13; Published 1/14

Fast SVM Training Using Approximate Extreme Points

Manu Nandan MNANDAN@UFL.EDU
Department of Computer and Information Science and Engineering
University of Florida
Gainesville, FL 32611, USA

Pramod P. Khargonekar PPK@ECE.UFL.EDU
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL 32611, USA

Sachin S. Talathi TALATHI@GMAIL.COM
Qualcomm Research Center
5775 Morehouse Dr
San Diego, CA 92121, USA

Editor: Sathiya Keerthi

Abstract

Applications of non-linear kernel support vector machines (SVMs) to large data sets is seriously hampered by its excessive training time. We propose a modification, called the approximate extreme points support vector machine (AESVM), that is aimed at overcoming this burden. Our approach relies on conducting the SVM optimization over a carefully selected subset, called the representative set, of the training data set. We present analytical results that indicate the similarity of AESVM and SVM solutions. A linear time algorithm based on convex hulls and extreme points is used to compute the representative set in kernel space. Extensive computational experiments on nine data sets compared AESVM to LIBSVM (Chang and Lin, 2011), CVM (Tsang et al., 2005) , BVM (Tsang et al., 2007), LASVM (Bordes et al., 2005), SVM$^{\text{perf}}$ (Joachims and Yu, 2009), and the random features method (Rahimi and Recht, 2007). Our AESVM implementation was found to train much faster than the other methods, while its classification accuracy was similar to that of LIBSVM in all cases. In particular, for a seizure detection data set, AESVM training was almost 500 times faster than LIBSVM and LASVM and 20 times faster than CVM and BVM. Additionally, AESVM also gave competitively fast classification times.

Keywords: support vector machines, convex hulls, large scale classification, non-linear kernels, extreme points

1. Introduction

Several real world applications require solutions of classification problems on large data sets. Even though SVMs are known to give excellent classification results, their application to problems with large data sets is impeded by the burdensome training time requirements. Recently, much progress has been made in the design of fast training algorithms (Fan et al., 2008; Shalev-Shwartz et al., 2011) for SVMs with the linear kernel (linear SVMs). However, many applications require SVMs with non-linear kernels for accurate classification. Training

time complexity for SVMs with non-linear kernels is typically quadratic in the size of the training data set (Shalev-Shwartz and Srebro, 2008). The difficulty of the long training time is exacerbated when grid search with cross-validation is used to derive the optimal hyper-parameters, since this requires multiple SVM training runs. Another problem that sometimes restricts the applicability of SVMs is the long classification time. The time complexity of SVM classification is linear in the number of support vectors and in some applications the number of support vectors is found to be very large (Guo et al., 2005).

In this paper, we propose a new approach for fast SVM training. Consider a two class data set of N data vectors, $\mathbf{X} = \{\mathbf{x}_i : \mathbf{x}_i \in \mathbb{R}^D, i = 1, 2, ..., N\}$, and the corresponding target labels $\mathbf{Y} = \{y_i : y_i \in [-1, 1], i = 1, 2, ..., N\}$. The SVM primal problem can be represented as the following unconstrained optimization problem (Teo et al., 2010; Shalev-Shwartz et al., 2011):

$$\begin{aligned} &\min_{\mathbf{w},b} F_1(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{i=1}^{N} l(\mathbf{w}, b, \phi(\mathbf{x}_i)), \qquad (1)\\ &\text{where } l(\mathbf{w}, b, \phi(\mathbf{x}_i)) = max\{0, 1 - y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b)\}, \forall \mathbf{x}_i \in \mathbf{X}\\ &\qquad \text{and } \phi : \mathbb{R}^D \to \mathbb{H}, b \in \mathbb{R}, \text{ and } \mathbf{w} \in \mathbb{H}, \text{ a Hilbert space.} \end{aligned}$$

Here $l(\mathbf{w}, b, \phi(\mathbf{x}_i))$ is the hinge loss of $\mathbf{x}_i$. Note that SVM formulations where the penalty parameter C is divided by N have been used extensively (Schölkopf et al., 2000; Franc and Sonnenburg, 2008; Joachims and Yu, 2009). These formulations enable better analysis of the scaling of C with N (Joachims, 2006). The problem in (1) requires optimization over N variables. In general, for SVM training algorithms, the training time will reduce if the size of the training data set is reduced.

In this paper, we present an alternative to (1), called approximate extreme points support vector machines (AESVM), that requires optimization over only a subset of the training data set. The AESVM formulation is:

$$\begin{aligned} &\min_{\mathbf{w},b} F_2(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{t=1}^{M} \beta_t l(\mathbf{w}, b, \phi(\mathbf{x}_t)), \qquad (2)\\ &\text{where } \mathbf{x}_t \in \mathbf{X}^*, \mathbf{w} \in \mathbb{H}, \text{ and } b \in \mathbb{R}. \end{aligned}$$

Here M is the number of vectors in the selected subset of $\mathbf{X}$, called the representative set $\mathbf{X}^*$. The constants β_t are defined in (9). We will prove in Section 3.2 that:

- $F_1(\mathbf{w}_1^*, b_1^*) - F_2(\mathbf{w}_2^*, b_2^*) \leq C\sqrt{C\epsilon}$, where $(\mathbf{w}_1^*, b_1^*)$ and $(\mathbf{w}_2^*, b_2^*)$ are the solutions of (1) and (2) respectively.
- Under the assumptions given in corollary 4, $F_1(\mathbf{w}_2^*, b_2^*) - F_1(\mathbf{w}_1^*, b_1^*) \leq 2C\sqrt{C\epsilon}$.
- The AESVM problem minimizes an upper bound of a low rank Gram matrix approximation of the SVM objective function.

Based on these results we claim that solving the problem in (2) yields a solution close to that of (1) for a small value of ϵ, the approximation error bound. As a by-product of the

reduction in size of the training set, AESVM is also observed to result in fast classification. Considering that the representative set will have to be computed several times if grid search is used to find the optimum hyper-parameter combination, we also propose fast algorithms to compute $\mathbf{Z}^*$. In particular, we present an algorithm of time complexity $O(N)$ and an alternative algorithm of time complexity $O(N \log_2 \frac{N}{P})$ to compute $\mathbf{Z}^*$, where P is a predefined large integer.

Our main contribution is the new AESVM formulation that can be used for fast SVM training. We develop and analyze our technique along the following lines:

- *Theoretical:* Theorems 1 and 2 and Corollaries 3 to 5 provide some theoretical basis for the use of AESVM as a computationally less demanding alternative to the SVM formulation.

- *Algorithmic:* The algorithm DeriveRS, described in Section 4, computes the representative set in linear time.

- *Experimental:* Our extensive experiments on nine data sets of varying characteristics illustrate the suitability of applying AESVM to classification on large data sets.

This paper is organized as follows: in Section 2, we briefly discuss recent research on fast SVM training that is closely related to this work. Next, we provide the definition of the representative set and discuss properties of AESVM. In Section 4, we present efficient algorithms to compute the representative set and analyze its computational complexity. Section 5 describes the results of our computational experiments. We compared AESVM to the widely used LIBSVM library, core vector machines (CVM), ball vector machines (BVM), LASVM, SVM$^{\text{perf}}$, and the random features method by Rahimi and Recht (2007). Our experiments used eight publicly available data sets and a data set on EEG from an animal model of epilepsy (Talathi et al., 2008; Nandan et al., 2010). We conclude with a discussion of the results of this paper in Section 6.

2. Related Work

Several methods have been proposed to efficiently solve the SVM optimization problem. SVMs require special algorithms, as standard optimization algorithms such as interior point methods (Boyd and Vandenberghe, 2004; Shalev-Shwartz et al., 2011) have large memory and training time requirements that make it infeasible for large data sets. In the following sections we discuss the most widely used strategies to solve the SVM optimization problem. We present a comparison of some of these methods to AESVM in Section 6. SVM solvers can be broadly divided into two categories as described below.

2.1 Dual Optimization

The SVM primal problem is a convex optimization problem with strong duality (Boyd and Vandenberghe, 2004). Hence its solution can be arrived at by solving its dual formulation

given below:

$$\max_{\alpha} \; L_1(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{3}$$

$$\text{subject to } 0 \leq \alpha_i \leq \frac{C}{N} \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0.$$

Here $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$, is the kernel product (Schölkopf and Smola, 2001) of the data vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, and α is a vector of all variables α_i. Solving the dual problem is computationally simpler, especially for non-linear kernels and a majority of the SVM solvers use dual optimization. Some of the major dual optimization algorithms are discussed below.

Decomposition methods (Osuna et al., 1997) have been widely used to solve (3). These methods optimize over a subset of the training data set, called the 'working set', at each algorithm iteration. SVM^{light} (Joachims, 1999) and SMO (Platt, 1999) are popular examples of decomposition methods. Both these methods have a quadratic time complexity for linear and non-linear SVM kernels (Shalev-Shwartz and Srebro, 2008). Heuristics such as shrinking and caching (Joachims, 1999) enable fast convergence of decomposition methods and reduce their memory requirements. LIBSVM (Chang and Lin, 2011) is a very popular implementation of SMO. A *dual coordinate descent* (Hsieh et al., 2008) SVM solver computes the optimal α value by modifying one variable α_i per algorithm iteration. Dual coordinate descent SVM solvers, such as LIBLINEAR (Fan et al., 2008), have been proposed primarily for the linear kernel.

Approximations of the Gram matrix (Fine and Scheinberg, 2002; Drineas and Mahoney, 2005), have been proposed to increase training speed and reduce memory requirements of SVM solvers. The Gram matrix is the NxN square matrix composed of the kernel products $K(\mathbf{x}_i, \mathbf{x}_j)$, $\forall \mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$. *Training set selection* methods attempt to reduce the SVM training time by optimizing over a selected subset of the training set. Several distinct approaches have been used to select the subset. Some methods use clustering based approaches (Pavlov et al., 2000) to select the subsets. In Yu et al. (2003), hierarchical clustering is performed to derive a data set that has more data vectors near the classification boundary than away from it. Minimum enclosing ball clustering is used in Cervantes et al. (2008) to remove data vectors that are unlikely to contribute to the SVM training. *Random sampling* of training data is another approach followed by approximate SVM solvers. Lee and Mangasarian (2001) proposed reduced support vector machines (RSVM), in which only a random subset of the training data set is used. Bordes et al. (2005) proposed the LASVM algorithm that uses *active selection* techniques to train SVMs on a subset of the training data set.

A *core set* (Clarkson, 2010) can be loosely defined as the subset of $\mathbf{X}$ for which the solution of an optimization problem such as (3) has a solution similar to that for the entire data set $\mathbf{X}$. Tsang et al. (2005) proved that the L2-SVM is a reformulation of the minimum enclosing ball problem for some kernels. They proposed core vector machine (CVM) that approximately solves the L2-SVM formulation using core sets. A simplified version of CVM called ball vector machine (BVM) was proposed in Tsang et al. (2007), where only an enclosing ball is computed. Gärtner and Jaggi (2009) proposed an algorithm to solve the L1-SVM problem, by computing the shortest distance between two polytopes (Bennett and

Bredensteiner, 2000) using core sets. However, there are no published results on solving L1-SVM with non-linear kernels using their algorithm.

Another method used to approximately solve the SVM problem is to map the data vectors into a *randomized feature space* that is relatively low dimensional compared to the kernel space $\mathbb{H}$ (Rahimi and Recht, 2007). Inner products of the projections of the data vectors are approximations of their kernel product. This effectively reduces the non-linear SVM problem into the simpler linear SVM problem, enabling the use of fast linear SVM solvers. This method is referred as RfeatSVM in the following sections of this document.

2.2 Primal Optimization

In recent years, linear SVMs have found increased use in applications with high-dimensional data sets. This has led to a surge in publications on efficient primal SVM solvers, which are mostly used for linear SVMs. To overcome the difficulties caused by the non-differentiability of the primal problem, the following methods are used.

Stochastic sub-gradient descent (Zhang, 2004) uses the sub-gradient computed at some data vector $\mathbf{x}_i$ to iteratively update $\mathbf{w}$. Shalev-Shwartz et al. (2011) proposed a stochastic sub-gradient descent SVM solver, Pegasos, that is reported to be among the fastest linear SVM solvers. *Cutting plane algorithms* (Kelley, 1960) solve the primal problem by successively tightening a piecewise linear approximation. It was employed by Joachims (2006) to solve linear SVMs with their implementation SVM^{perf}. This work was generalized in Joachims and Yu (2009) to include non-linear SVMs by approximately estimating $\mathbf{w}$ with arbitrary basis vectors using the fix-point iteration method (Schölkopf and Smola, 2001). Teo et al. (2010) proposed a related method for linear SVMs, that corrected some stability issues in the cutting plane methods.

3. Analysis of AESVM

As mentioned in the introduction, AESVM is an optimization problem on a subset of the training data set called the representative set. In this section we first define the representative set. Then we present some properties of AESVM. These results are intended to provide theoretical justifications for the use of AESVM as an approximation to the SVM problem (1).

3.1 Definition of the Representative Set

The convex hull of a set $\mathbf{X}$ is the smallest convex set containing $\mathbf{X}$ (Rockafellar, 1996) and can be obtained by taking all possible convex combinations of elements of $\mathbf{X}$. Assuming $\mathbf{X}$ is finite, the convex hull is a polygon. The extreme points of $\mathbf{X}$, $EP(\mathbf{X})$, are defined to be the vertices of the convex polygon formed by the convex hull of $\mathbf{X}$. Any vector $\mathbf{x}_i$ in $\mathbf{X}$ can be represented as a convex combination of vectors in $EP(\mathbf{X})$:

$$\mathbf{x}_i = \sum_{\mathbf{x}_t \in EP(\mathbf{X})} \pi_{i,t}\mathbf{x}_t, \text{ where } 0 \leq \pi_{i,t} \leq 1, \text{ and } \sum_{\mathbf{x}_t \in EP(\mathbf{X})} \pi_{i,t} = 1.$$

We can see that functions of any data vector in $\mathbf{X}$ can be computed using only $EP(\mathbf{X})$ and the convex combination weights $\{\pi_{i,t}\}$. The design of AESVM is motivated by the

intuition that the use of extreme points may provide computational efficiency. However, extreme points are not useful in all cases, as for some kernels all data vectors are extreme points in kernel space. For example, for the Gaussian kernel, $K(\mathbf{x}_i, \mathbf{x}_i) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_i) = 1$. This implies that all the data vectors lie on the surface of the unit ball in the Gaussian kernel space[1] and therefore are extreme points. Hence, we introduce the concept of *approximate extreme points*.

Consider the set of transformed data vectors:

$$\mathbf{Z} = \{\mathbf{z}_i : \mathbf{z}_i = \phi(\mathbf{x}_i), \forall \mathbf{x}_i \in \mathbf{X}\}. \tag{4}$$

Here, the explicit representation of vectors in kernel space is only for the ease of understanding and all the computations are performed using kernel products. Let V be a positive integer that is much smaller than N and ϵ be a small positive real number. For notational simplicity, we assume N is divisible by V. Let $\mathbf{Z}_l$ be subsets of $\mathbf{Z}$ for $l = 1, 2, ..., (\frac{N}{V})$, such that $\mathbf{Z} = \underset{l}{\cup}\mathbf{Z}_l$ and $\mathbf{Z}_l \cap \mathbf{Z}_m = \emptyset$ for $l \neq m$, where $m = 1, 2, ..., (\frac{N}{V})$. We require that the subsets $\mathbf{Z}_l$ satisfy $|\mathbf{Z}_l| = V, \forall l$ and

$$\forall \mathbf{z}_i, \mathbf{z}_j \in \mathbf{Z}_l, \text{ we have } y_i = y_j, \tag{5}$$

where $|\mathbf{Z}_l|$ denotes the cardinality of $\mathbf{Z}_l$. Let $\mathbf{Z}_{l_q}$ be an arbitrary subset of $\mathbf{Z}_l$, $\mathbf{Z}_{l_q} \subseteq \mathbf{Z}_l$. Next, for any $\mathbf{z}_i \in \mathbf{Z}_l$ we define:

$$f(\mathbf{z}_i, \mathbf{Z}_{l_q}) = \underset{\overline{\mu_i}}{\mathbf{min}} \|\mathbf{z}_i - \sum_{\mathbf{z}_t \in \mathbf{Z}_{l_q}} \mu_{i,t}\mathbf{z}_t\|^2, \tag{6}$$

$$\text{s.t. } 0 \leq \mu_{i,t} \leq 1, \text{ and } \sum_{\mathbf{z}_t \in \mathbf{Z}_{l_q}} \mu_{i,t} = 1.$$

A subset Z_l^* is said to be an ϵ - approximate extreme points subset of Z_l if:

$$\underset{\mathbf{z}_i \in \mathbf{Z}_l}{\mathbf{max}} f(\mathbf{z}_i, Z_l^*) \leq \epsilon.$$

We will drop the prefix ϵ for simplicity and refer to Z_l^* as approximate extreme points subset. Note that it is not unique. Intuitively, its cardinality will be related to computational savings obtained using the approach proposed in this paper. We have chosen to not use approximate extreme points subset of smallest cardinality to maintain flexibility.

It can be seen that $\mu_{i,t}$ for $\mathbf{z}_t \in \mathbf{Z}_l^*$ are analogous to the convex combination weights $\pi_{i,t}$ for $\mathbf{x}_t \in EP(\mathbf{X})$. The *representative set* $\mathbf{Z}^*$ of $\mathbf{Z}$ is the union of the sets of approximate extreme points of its subsets $\mathbf{Z}_l$.

$$\mathbf{Z}^* = \underset{l=1}{\overset{\frac{N}{V}}{\cup}} \mathbf{Z}_l^*.$$

The representative set has properties that are similar to $EP(\mathbf{X})$. Given any $\mathbf{z}_i \in \mathbf{Z}$, we can find $\mathbf{Z}_l$ such that $\mathbf{z}_i \in \mathbf{Z}_l$. Let $\gamma_{i,t} = \{\mu_{i,t} \text{ for } \mathbf{z}_t \in \mathbf{Z}_l^* \text{ and } \mathbf{z}_i \in \mathbf{Z}_l, \text{ and 0 otherwise}\}$. Now using (6), we can write:

$$\mathbf{z}_i = \sum_{\mathbf{z}_t \in \mathbf{Z}^*} \gamma_{i,t}\mathbf{z}_t + \tau_i. \tag{7}$$

1. We define the square of the distance of $\mathbf{x}$ from origin in kernel space as $K(\mathbf{x}, \mathbf{x})$.

Here τ_i is a vector that accounts for the approximation error $f(\mathbf{z}_i, \mathbf{Z}_{l_q})$ in (6). From (6) and (7) we can conclude that:

$$\|\tau_i\|^2 \leq \epsilon \; \forall \; \mathbf{z}_i \in \mathbf{Z}. \tag{8}$$

Since ϵ will be set to a very small positive constant, we can infer that τ_i is a very small vector. The weights $\gamma_{i,t}$ are used to define β_t in (2) as:

$$\beta_t = \sum_{i=1}^{N} \gamma_{i,t}. \tag{9}$$

For ease of notation, we refer to the set $\mathbf{X}^* := \{\mathbf{x}_t : \mathbf{z}_t \in \mathbf{Z}^*\}$ as the representative set of $\mathbf{X}$ in the remainder of this paper. For the sake of simplicity, we assume that all $\gamma_{i,t}, \beta_t, \mathbf{X},$ and $\mathbf{X}^*$ are arranged so that $\mathbf{X}^*$ is positioned as the first M vectors of $\mathbf{X}$, where $M = |\mathbf{Z}^*|$.

3.2 Properties of AESVM

Consider the following optimization problem.

$$\min_{\mathbf{w},b} F_3(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 \; + \frac{C}{N}\sum_{i=1}^{N} l(\mathbf{w}, b, \mathbf{u}_i), \tag{10}$$
$$\text{where } \mathbf{u}_i = \sum_{t=1}^{M} \gamma_{i,t}\mathbf{z}_t, \mathbf{z}_t \in \mathbf{Z}^*, \mathbf{w} \in \mathbb{H}, \text{ and } b \in \mathbb{R}.$$

We use the problem in (10) as an intermediary between (1) and (2). The intermediate problem (10) has a direct relation to the AESVM problem, as given in the following theorem. The properties of the *max* function given below are relevant to the following discussion:

$$max(0, A + B) \leq max(0, A) + max(0, B), \tag{11}$$
$$max(0, A - B) \geq max(0, A) - max(0, B), \tag{12}$$
$$\sum_{i=1}^{N} max(0, c_i A) = max(0, A)\sum_{i=1}^{N} c_i, \tag{13}$$

for $A, B, c_i \in \mathbb{R}$ and $c_i \geq 0$.

Theorem 1 *Let $F_3(\mathbf{w}, b)$ and $F_2(\mathbf{w}, b)$ be as defined in (10) and (2) respectively. Then,*

$$F_3(\mathbf{w}, b) \leq F_2(\mathbf{w}, b) \; , \forall \mathbf{w} \in \mathbb{H} \text{ and } b \in \mathbb{R}.$$

Proof Let $\mathcal{L}_2(\mathbf{w}, b, \mathbf{X}^*) = \frac{C}{N}\sum_{t=1}^{M} l(\mathbf{w}, b, \mathbf{z}_t)\sum_{i=1}^{N}\gamma_{i,t}$ and $\mathcal{L}_3(\mathbf{w}, b, \mathbf{X}^*) = \frac{C}{N}\sum_{i=1}^{N} l(\mathbf{w}, b, \mathbf{u}_i)$, where $\mathbf{u}_i = \sum_{t=1}^{M}\gamma_{i,t}\mathbf{z}_t$. From the properties of $\gamma_{i,t}$ in (6), and from (5) we get:

$$\begin{aligned}\mathcal{L}_3(\mathbf{w}, b, \mathbf{X}^*) &= \frac{C}{N}\sum_{i=1}^{N} max\left[0, \left\{1 - y_i(\mathbf{w}^T\sum_{t=1}^{M}\gamma_{i,t}\mathbf{z}_t + b)\right\}\right] \\ &= \frac{C}{N}\sum_{i=1}^{N} max\left[0, \sum_{t=1}^{M}\gamma_{i,t}\left\{1 - y_t(\mathbf{w}^T\mathbf{z}_t + b)\right\}\right].\end{aligned}$$

Using properties (11) and (13) we get:

$$\begin{aligned}\mathcal{L}_3(\mathbf{w}, b, \mathbf{X}^*) &\leq \frac{C}{N}\sum_{i=1}^{N}\sum_{t=1}^{M} max\left[0, \gamma_{i,t}\left\{1 - y_t(\mathbf{w}^T\mathbf{z}_t + b)\right\}\right] \\ &= \frac{C}{N}\sum_{t=1}^{M} max\left[0, 1 - y_t(\mathbf{w}^T\mathbf{z}_t + b)\right]\sum_{i=1}^{N}\gamma_{i,t} \\ &= \mathcal{L}_2(\mathbf{w}, b, \mathbf{X}^*).\end{aligned}$$

Adding $\frac{1}{2}\|\mathbf{w}\|^2$ to both sides of the inequality above we get

$$F_3(\mathbf{w}, b) \leq F_2(\mathbf{w}, b).$$

■

The following theorem gives a relationship between the SVM problem and the intermediate problem.

Theorem 2 *Let $F_1(\mathbf{w}, b)$ and $F_3(\mathbf{w}, b)$ be as defined in (1) and (10) respectively. Then,*

$$-\frac{C}{N}\sum_{i=1}^{N} max\left\{0, y_i\mathbf{w}^T\tau_i\right\} \leq F_1(\mathbf{w}, b) - F_3(\mathbf{w}, b) \leq \frac{C}{N}\sum_{i=1}^{N} max\left\{0, -y_i\mathbf{w}^T\tau_i\right\},$$

$\forall \mathbf{w} \in \mathbb{H}$ and $b \in \mathbb{R}$, where $\tau_i \in \mathbb{H}$ is the vector defined in (7).

Proof Let $\mathcal{L}_1(\mathbf{w}, b, \mathbf{X}) = \frac{C}{N}\sum_{i=1}^{N} l(\mathbf{w}, b, \mathbf{z}_i)$, denote the average hinge loss that is minimized in (1) and $\mathcal{L}_3(\mathbf{w}, b, \mathbf{X}^*)$ be as defined in Theorem 1. Using (7) and (1) we get:

$$\begin{aligned}\mathcal{L}_1(\mathbf{w}, b, \mathbf{X}) &= \frac{C}{N}\sum_{i=1}^{N} max\left\{0, 1 - y_i(\mathbf{w}^T\mathbf{z_i} + b)\right\} \\ &= \frac{C}{N}\sum_{i=1}^{N} max\left\{0, 1 - y_i(\mathbf{w}^T(\sum_{t=1}^{M}\gamma_{i,t}\mathbf{z}_t + \tau_i) + b)\right\}.\end{aligned}$$

From the properties of $\gamma_{i,t}$ in (6), and from (5) we get:

$$\mathcal{L}_1(\mathbf{w}, b, \mathbf{X}) = \frac{C}{N}\sum_{i=1}^{N} max\left\{0, \sum_{t=1}^{M}\gamma_{i,t}(1 - y_t(\mathbf{w}^T\mathbf{z}_t + b)) - y_i\mathbf{w}^T\tau_i\right\}. \tag{14}$$

Using (11) on (14), we get:

$$\begin{aligned}\mathcal{L}_1(\mathbf{w}, b, \mathbf{X}) &\leq \frac{C}{N}\sum_{i=1}^{N} max\left[0, \sum_{t=1}^{M}\gamma_{i,t}\left\{1 - y_t(\mathbf{w}^T\mathbf{z}_t + b)\right\}\right] + \frac{C}{N}\sum_{i=1}^{N} max\left\{0, -y_i\mathbf{w}^T\tau_i\right\} \\ &= \mathcal{L}_3(\mathbf{w}, b, \mathbf{X}^*) + \frac{C}{N}\sum_{i=1}^{N} max\left\{0, -y_i\mathbf{w}^T\tau_i\right\}.\end{aligned}$$

Using (12) on (14), we get:

$$\begin{aligned}\mathcal{L}_1(\mathbf{w}, b, \mathbf{X}) &\geq \frac{C}{N}\sum_{i=1}^{N} max\left[0, \sum_{t=1}^{M}\gamma_{i,t}\left\{1 - y_t(\mathbf{w}^T\mathbf{z}_t + b)\right\}\right] - \frac{C}{N}\sum_{i=1}^{N} max\left\{0, y_i\mathbf{w}^T\tau_i\right\} \\ &= \mathcal{L}_3(\mathbf{w}, b, \mathbf{X}^*) - \frac{C}{N}\sum_{i=1}^{N} max\left\{0, y_i\mathbf{w}^T\tau_i\right\}.\end{aligned}$$

From the two inequalities above we get,

$$\begin{aligned}\mathcal{L}_3(\mathbf{w}, b, \mathbf{X}^*) - \frac{C}{N}\sum_{i=1}^{N} max\left\{0, y_i\mathbf{w}^T\tau_i\right\} &\leq \mathcal{L}_1(\mathbf{w}, b, \mathbf{X}) \\ &\leq \mathcal{L}_3(\mathbf{w}, b, \mathbf{X}^*) + \frac{C}{N}\sum_{i=1}^{N} max\left\{0, -y_i\mathbf{w}^T\tau_i\right\}.\end{aligned}$$

Adding $\frac{1}{2}\|\mathbf{w}\|^2$ to the inequality above we get

$$F_3(\mathbf{w}, b) - \frac{C}{N}\sum_{i=1}^{N} max\left\{0, y_i\mathbf{w}^T\tau_i\right\} \leq F_1(\mathbf{w}, b) \leq F_3(\mathbf{w}, b) + \frac{C}{N}\sum_{i=1}^{N} max\left\{0, -y_i\mathbf{w}^T\tau_i\right\}.$$

■

Using the above theorems we derive the following corollaries. These results provide the theoretical justification for AESVM.

Corollary 3 *Let* $(\mathbf{w}_1^*, b_1^*)$ *be the solution of (1) and* $(\mathbf{w}_2^*, b_2^*)$ *be the solution of (2). Then,*

$$F_1(\mathbf{w}_1^*, b_1^*) - F_2(\mathbf{w}_2^*, b_2^*) \leq C\sqrt{C\epsilon}.$$

Proof It is known that $\|\mathbf{w}_1^*\| \leq \sqrt{C}$ (see Shalev-Shwartz et al., 2011, Theorem 1). It is straight forward to see that the same result also applies to AESVM, $\|\mathbf{w}_2^*\| \leq \sqrt{C}$. Based on (8) we know that $\|\tau_i\| \leq \sqrt{\epsilon}$. From Theorem 2 we get:

$$F_1(\mathbf{w}_2^*, b_2^*) - F_3(\mathbf{w}_2^*, b_2^*) \leq \frac{C}{N}\sum_{i=1}^{N} max\left\{0, -y_i \mathbf{w}_2^{*T}\tau_i\right\} \leq \frac{C}{N}\sum_{i=1}^{N}\|\mathbf{w}_2^*\|\|\tau_i\|$$
$$\leq \frac{C}{N}\sum_{i=1}^{N}\sqrt{C\epsilon} = C\sqrt{C\epsilon}.$$

Since $(\mathbf{w}_1^*, b_1^*)$ is the solution of (1), $F_1(\mathbf{w}_1^*, b_1^*) \leq F_1(\mathbf{w}_2^*, b_2^*)$. Using this property and Theorem 1 in the inequality above, we get:

$$F_1(\mathbf{w}_1^*, b_1^*) - F_2(\mathbf{w}_2^*, b_2^*) \leq F_1(\mathbf{w}_1^*, b_1^*) - F_3(\mathbf{w}_2^*, b_2^*)$$
$$\leq F_1(\mathbf{w}_2^*, b_2^*) - F_3(\mathbf{w}_2^*, b_2^*) \leq C\sqrt{C\epsilon}.$$

■

Now we demonstrate some properties of AESVM using the dual problem formulations of AESVM and the intermediate problem. The dual form of AESVM is given by:

$$\max_{\hat{\alpha}} \; L_2(\hat{\alpha}) = \sum_{t=1}^{M}\hat{\alpha}_t - \frac{1}{2}\sum_{t=1}^{M}\sum_{s=1}^{M}\hat{\alpha}_t\hat{\alpha}_s y_t y_s \mathbf{z}_t^T\mathbf{z}_s, \tag{15}$$
$$\text{subject to } 0 \leq \hat{\alpha}_t \leq \frac{C}{N}\sum_{i=1}^{N}\gamma_{i,t} \text{ and } \sum_{t=1}^{M}\hat{\alpha}_t y_t = 0.$$

The dual form of the intermediate problem is given by:

$$\max_{\breve{\alpha}} \; L_3(\breve{\alpha}) = \sum_{i=1}^{N}\breve{\alpha}_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\breve{\alpha}_i\breve{\alpha}_j y_i y_j \mathbf{u}_i^T\mathbf{u}_j, \tag{16}$$
$$\text{subject to } 0 \leq \breve{\alpha}_i \leq \frac{C}{N} \text{ and } \sum_{i=1}^{N}\breve{\alpha}_i y_i = 0.$$

Consider the mapping function $h : \mathbb{R}^N \to \mathbb{R}^M$, defined as

$$h(\breve{\alpha}) = \{\tilde{\alpha}_t : \tilde{\alpha}_t = \sum_{i=1}^{N}\gamma_{i,t}\breve{\alpha}_i\}. \tag{17}$$

It can be seen that the objective functions $L_2(h(\breve{\alpha}))$ and $L_3(\breve{\alpha})$ are identical.

$$L_2(h(\breve{\alpha})) = \sum_{t=1}^{M}\tilde{\alpha}_t - \frac{1}{2}\sum_{t=1}^{M}\sum_{s=1}^{M}\tilde{\alpha}_t\tilde{\alpha}_s y_t y_s \mathbf{z}_t^T\mathbf{z}_s$$
$$= \sum_{i=1}^{N}\breve{\alpha}_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\breve{\alpha}_i\breve{\alpha}_j y_i y_j \mathbf{u}_i^T\mathbf{u}_j$$
$$= L_3(\breve{\alpha}).$$

It is also straight forward to see that, for any feasible $\breve{\alpha}$ of (16), $h(\breve{\alpha})$ is a feasible point of (15) as it satisfies the constraints in (15). However, the converse is not always true. With that clarification, we present the following corollary.

Corollary 4 *Let $(\mathbf{w}_1^*, b_1^*)$ be the solution of (1) and $(\mathbf{w}_2^*, b_2^*)$ be the solution of (2). Let $\hat{\alpha}_2$ be the dual variable corresponding to $(\mathbf{w}_2^*, b_2^*)$. Let $h(\breve{\alpha}_2)$ be as defined in (17). If there exists an $\breve{\alpha}_2$ such that $h(\breve{\alpha}_2) = \hat{\alpha}_2$ and $\breve{\alpha}_2$ is a feasible point of (16), then,*

$$F_1(\mathbf{w}_2^*, b_2^*) - F_1(\mathbf{w}_1^*, b_1^*) \leq 2C\sqrt{C\epsilon}.$$

Proof Let $(\mathbf{w}_3^*, b_3^*)$ be the solution of (10) and $\breve{\alpha}_3$ the solution of (16). We know that $L_3(\breve{\alpha}_2) = L_2(\hat{\alpha}_2) = F_2(\mathbf{w}_2^*, b_2^*)$ and $L_3(\breve{\alpha}_3) = F_3(\mathbf{w}_3^*, b_3^*)$. Since $L_3(\breve{\alpha}_3) \geq L_3(\breve{\alpha}_2)$, we get

$$F_3(\mathbf{w}_3^*, b_3^*) \geq F_2(\mathbf{w}_2^*, b_2^*).$$

But, from Theorem 1 we know $F_3(\mathbf{w}_3^*, b_3^*) \leq F_3(\mathbf{w}_2^*, b_2^*) \leq F_2(\mathbf{w}_2^*, b_2^*)$. Hence

$$F_3(\mathbf{w}_3^*, b_3^*) = F_3(\mathbf{w}_2^*, b_2^*).$$

From the above result we get

$$F_3(\mathbf{w}_2^*, b_2^*) - F_3(\mathbf{w}_1^*, b_1^*) \leq 0. \tag{18}$$

From Theorem 2 we have the following inequalities:

$$-\frac{C}{N}\sum_{i=1}^{N} max\left\{0, y_i \mathbf{w}_1^{*T}\tau_i\right\} \leq F_1(\mathbf{w}_1^*, b_1^*) - F_3(\mathbf{w}_1^*, b_1^*), \text{ and} \tag{19}$$

$$F_1(\mathbf{w}_2^*, b_2^*) - F_3(\mathbf{w}_2^*, b_2^*) \leq \frac{C}{N}\sum_{i=1}^{N} max\left\{0, -y_i \mathbf{w}_2^{*T}\tau_i\right\}. \tag{20}$$

Adding (19) and (20) we get:

$$F_1(\mathbf{w}_2^*, b_2^*) - F_1(\mathbf{w}_1^*, b_1^*) \leq R + \frac{C}{N}\sum_{i=1}^{N}\left[max\left\{0, -y_i \mathbf{w}_2^{*T}\tau_i\right\} + max\left\{0, y_i \mathbf{w}_1^{*T}\tau_i\right\}\right], \tag{21}$$

where $R = F_3(\mathbf{w}_2^*, b_2^*) - F_3(\mathbf{w}_1^*, b_1^*)$. Using (18) and the properties $\|\mathbf{w}_2^*\| \leq \sqrt{C}$ and $\|\mathbf{w}_1^*\| \leq \sqrt{C}$ in (21) we get

$$\begin{aligned} F_1(\mathbf{w}_2^*, b_2^*) - F_1(\mathbf{w}_1^*, b_1^*) &\leq \frac{C}{N}\sum_{i=1}^{N}\left[max\left\{0, -y_i \mathbf{w}_2^{*T}\tau_i\right\} + max\left\{0, y_i \mathbf{w}_1^{*T}\tau_i\right\}\right] \\ &\leq \frac{C}{N}\sum_{i=1}^{N}\|\mathbf{w}_2^*\|\|\tau_i\| + \|\mathbf{w}_1^*\|\|\tau_i\| \\ &\leq \frac{C}{N}\sum_{i=1}^{N} 2\sqrt{C\epsilon} = 2C\sqrt{C\epsilon}. \end{aligned}$$

■

Now we prove a relationship between AESVM and the Gram matrix approximation methods mentioned in Section 2.1.

Corollary 5 *Let $L_1(\alpha)$, $L_3(\breve{\alpha})$, and $F_2(\mathbf{w}, b)$ be the objective functions of the SVM dual (3), intermediate dual (16) and AESVM (2) respectively. Let $\mathbf{z}_i$, τ_i, and $\mathbf{u}_i$ be as defined in (4), (7), and (10) respectively. Let $\mathbf{G}$ and $\tilde{\mathbf{G}}$ be the NxN matrices with $\mathbf{G}_{ij} = \mathbf{y}_i\mathbf{y}_j\mathbf{z}_i^T\mathbf{z}_j$ and $\tilde{\mathbf{G}}_{ij} = \mathbf{y}_i\mathbf{y}_j\mathbf{u}_i^T\mathbf{u}_j$ respectively. Then for any feasible $\breve{\alpha}, \alpha, \mathbf{w}$, and b:*

1. *Rank of $\tilde{\mathbf{G}} = M, L_1(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\alpha\mathbf{G}\alpha^T, L_3(\breve{\alpha}) = \sum_{i=1}^{N}\breve{\alpha}_i - \frac{1}{2}\breve{\alpha}\tilde{\mathbf{G}}\breve{\alpha}^T$, and*

$$\text{Trace}(\mathbf{G} - \tilde{\mathbf{G}}) \leq N\epsilon + 2\sum_{t=1}^{M}\mathbf{z}_t^T\sum_{i=1}^{N}\gamma_{i,t}\tau_i.$$

2. *$F_2(\mathbf{w}, b) \geq L_3(\breve{\alpha})$.*

Proof Using $\mathbf{G}$, the SVM dual objective function $L_1(\alpha)$ can be represented as:

$$L_1(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\alpha\mathbf{G}\alpha^T.$$

Similarly, $L_3(\breve{\alpha})$ can be represented using $\tilde{\mathbf{G}}$ as:

$$L_3(\breve{\alpha}) = \sum_{i=1}^{N}\breve{\alpha}_i - \frac{1}{2}\breve{\alpha}\tilde{\mathbf{G}}\breve{\alpha}^T.$$

Applying $\mathbf{u}_i = \sum_{t=1}^{M}\gamma_{i,t}\mathbf{z}_t,\ \forall\mathbf{z}_t \in \mathbf{Z}^*$ to the definition of $\tilde{\mathbf{G}}$, we get:

$$\tilde{\mathbf{G}} = \Gamma\mathbf{A}\Gamma^T.$$

Here $\mathbf{A}$ is the MxM matrix comprised of $\mathbf{A}_{ts} = \mathbf{y}_t\mathbf{y}_s\mathbf{z}_t^T\mathbf{z}_s,\ \forall\mathbf{z}_t, \mathbf{z}_s \in \mathbf{Z}^*$ and Γ is the NxM matrix with the elements $\Gamma_{it} = \gamma_{i,t}$. Hence the rank of $\tilde{\mathbf{G}} = M$ and intermediate dual problem (16) is a low rank approximation of the SVM dual problem (3).

The Gram matrix approximation error can be quantified using (7) and (8) as:

$$\begin{aligned}\text{Trace}(\mathbf{G} - \tilde{\mathbf{G}}) &= \sum_{i=1}^{N}\left[\mathbf{z}_i^T\mathbf{z}_i - (\sum_{t=1}^{M}\gamma_{i,t}\mathbf{z}_t)^T(\sum_{s=1}^{M}\gamma_{i,s}\mathbf{z}_s)\right] \\ &= \sum_{i=1}^{N}\left[\tau_i^T\tau_i + 2\sum_{t=1}^{M}\gamma_{i,t}\mathbf{z}_t^T\tau_i\right] \leq N\epsilon + 2\sum_{t=1}^{M}\mathbf{z}_t^T\sum_{i=1}^{N}\gamma_{i,t}\tau_i.\end{aligned}$$

By the principle of duality, we know that $F_3(\mathbf{w}, b) \geq L_3(\breve{\alpha}),\ \forall\mathbf{w} \in \mathbb{H}$ and $b \in \mathbb{R}$, where $\breve{\alpha}$ is any feasible point of (16). Using Theorem 1 on the inequality above, we get

$$F_2(\mathbf{w}, b) \geq L_3(\breve{\alpha}),\ \forall\mathbf{w} \in \mathbb{H}, b \in \mathbb{R} \text{ and feasible } \breve{\alpha}.$$

Thus the AESVM problem minimizes an upper bound $F_2(\mathbf{w}, b)$, of a rank M Gram matrix approximation of $L_1(\alpha)$. ■

Based on the theoretical results in this section, it is reasonable to suggest that for small values of ϵ, the solution of AESVM is close to the solution of SVM.

4. Computation of the Representative Set

In this section, we present algorithms to compute the representative set. *The AESVM formulation can be solved with any standard SVM solver such as SMO and hence we do not discuss methods to solve it.* As described in Section 3.1, we require an algorithm to compute approximate extreme points in kernel space. Osuna and Castro (2002) proposed an algorithm to derive extreme points of the convex hull of a data set in kernel space. Their algorithm is computationally intensive, with a time complexity of $O(N\ S(N))$, and is unsuitable for large data sets as $S(N)$ typically has a super-linear dependence on N. The function $S(N)$ denotes the time complexity of a SVM solver (required by their algorithm), to train on a data set of size N. We next propose two algorithms leveraging the work by Osuna and Castro (2002) to compute the representative set in kernel space $\mathbf{Z}^*$ with much smaller time complexities.

We followed the divide and conquer approach to develop our algorithms. The data set is first divided into subsets $\mathbf{X}_q, q = 1, 2, .., Q$, where $|\mathbf{X}_q| < P$, $Q \geq \frac{N}{P}$ and $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, .., \mathbf{X}_Q\}$. The parameter P is a predefined large integer. It is desired that each subset $\mathbf{X}_q$ contains data vectors that are more similar to each other than data vectors in other subsets. Our notion of similarity of data vectors in a subset, is that the distances between data vectors within a subset is less than the distances between data vectors in distinct subsets. Since performing such a segregation is computationally expensive, heuristics are used to greatly simplify the process. Instead of computing the distance of all data vectors from each other, only the distance from a few selected data vectors are used to segregate the data in the methods FLS2 and SLS described below.

The first level of segregation is followed by another level of segregation. We can regard the first level of segregation as coarse segregation and the second as fine segregation. Finally, the approximate extreme points of the subsets obtained after segregation, are computed. The two different algorithms to compute the representative set differ only in the first level of segregation as described below.

4.1 First Level of Segregation

We propose the methods, FLS1 and FLS2 given below to perform a first level of segregation. In the following description we use arrays Δ' and Δ'_2 of N elements. Each element of Δ' (Δ'_2), δ_i (δ_i^2) , contains the index in $\mathbf{X}$ of the last data vector of the subset to which $\mathbf{x}_i$ belongs. It is straight forward to replace this N element array with a smaller array of size equal to the number of subsets. We use a N element array for ease of description. The set $\mathbf{X}'$ denotes any set of data vectors.

1. FLS1($\mathbf{X}', P$)

For some applications, such as anomaly detection on sequential data, data vectors are found to be homogeneous within intervals. For example, the atmospheric conditions typically do not change within a few minutes and hence weather data is homogeneous for a short span. For such data sets it is enough to segregate the data vectors based on its position in the training data set. The same method can also be used on very large data sets without any homogeneity, in order to reduce computation time. The complexity of this method is $O(N')$, where $N' = |\mathbf{X}'|$.

$[\mathbf{X}', \Delta'] = \text{FLS1}(\mathbf{X}', P)$

---

1. For outerIndex = 1 ***to*** ceiling($\frac{|\mathbf{X}'|}{P}$)
2. For innerIndex = (outerIndex - 1)P ***to*** min((outerIndex)P,$|\mathbf{X}'|$)
3. Set $\delta_{innerIndex} = min((outerIndex)P, |\mathbf{X}'|)$

---

2. FLS2($\mathbf{X}', P$)

When the data set is not homogeneous within intervals or it is not excessively large we use the more sophisticated algorithm, FLS2, of time complexity $O(N' \log_2 \frac{N'}{P})$ given below. In step 1 of FLS2, the distance d_i in kernel space of all $\mathbf{x}_i \in \mathbf{X}'$ from $\mathbf{x}_j$ is computed as $d_i = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 = k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{x}_j, \mathbf{x}_j) - 2k(\mathbf{x}_i, \mathbf{x}_j)$. The algorithm FLS2($\mathbf{X}', P$), in effect builds a binary search tree, with each node containing the data vector $\mathbf{x}_k$ selected in step 2 that partitions a subset of the data set into two. The size of the subsets successively halve, on downward traversal from the root of the tree to the other nodes. When the size of all the subsets at a level become $\leq P$ the algorithm halts. The complexity of FLS2 can be derived easily when the algorithm is considered as an incomplete binary search tree building method. The last level of such a tree will have $O(\frac{N'}{P})$ nodes and consequently the height of the tree is $O(\log_2 \frac{N'}{P})$. At each level of the tree the calls to the BFPRT algorithm (Blum et al., 1973) and the rearrangement of the data vectors in steps 2 and 3 are of $O(N')$ time complexity. Hence the overall time complexity of FLS2($\mathbf{X}', P$) is $O(N' \log_2 \frac{N'}{P})$.

4.2 Second Level of Segregation

After the initial segregation, another method SLS($\mathbf{X}', V, \Delta'$) is used to further segregate each set $\mathbf{X}_q$ into smaller subsets $\mathbf{X}_{q_r}$ of maximum size V, $\mathbf{X}_q = \{\mathbf{X}_{q_1}, \mathbf{X}_{q_2}, ..., \mathbf{X}_{q_R}\}$, where V is predefined ($V < P$) and $R = ceiling(\frac{|\mathbf{X_q}|}{V})$. The algorithm SLS($\mathbf{X}', V, \Delta'$) is given below. In step 2.b, $\mathbf{x}_t$ is the data vector in $\mathbf{X}_q$ that is farthest from the origin in the space of the data vectors. For some kernels, such as the Gaussian kernel, all data vectors are equidistant from the origin in kernel space. If the algorithm chooses $\mathbf{a}_l$ in step 2.b based on distances in such kernel spaces, the choice would be arbitrary and such a situation is avoided here. Each iteration of the For loop in step 2 involves several runs of the BFPRT algorithm, with each run followed by a rearrangement of $\mathbf{X}_q$. Specifically, the BFPRT algorithm is first run on P data vectors, then on $P-V$ data vectors, then on $P-2V$ data vectors and so on. The time complexity of each iteration of the For loop including the BFPRT algorithm run and the rearrangement of data vectors is: $O(P+(P-V)+(P-2V)+..+V) \Rightarrow O(\frac{P^2}{V})$. The overall

$[\mathbf{X}', \Delta'] = \text{FLS2}(\mathbf{X}', P)$

1. Compute distance d_i in kernel space of all $\mathbf{x}_i \in \mathbf{X}'$ from the first vector $\mathbf{x}_j$ in $\mathbf{X}'$
2. Select $\mathbf{x}_k$ such that there exists $\frac{|\mathbf{X}'|}{2}$ data vectors $\mathbf{x}_i \in \mathbf{X}'$ with $d_i < d_k$, using the linear time BFPRT algorithm
3. Using $\mathbf{x}_k$, rearrange $\mathbf{X}'$ as $\mathbf{X}' = \{\mathbf{X}^1, \mathbf{X}^2\}$, where $\mathbf{X}^1 = \{\mathbf{x}_i : d_i < d_k, \mathbf{x}_i \in \mathbf{X}'\}$ and $\mathbf{X}^2 = \{\mathbf{x}_i : \mathbf{x}_i \in \mathbf{X}' \text{ and } \mathbf{x}_i \notin \mathbf{X}^1\}$
4. If $\frac{|\mathbf{X}'|}{2} \leq P$

   For i where $\mathbf{x}_i \in \mathbf{X}^1$, set $\delta_i =$ index of last data vector in $\mathbf{X}^1$.

   For i where $\mathbf{x}_i \in \mathbf{X}^2$, set $\delta_i =$ index of last data vector in $\mathbf{X}^2$.
5. If $\frac{|\mathbf{X}'|}{2} > P$

   Run FLS2($\mathbf{X}^1, P$) and FLS2($\mathbf{X}^2, P$)

complexity of SLS($\mathbf{X}', V, \Delta'$) considering the Q For loop iterations is $O(\frac{N'}{P}\frac{P^2}{V}) \Rightarrow O(\frac{N'P}{V})$, since $Q = O(\frac{N'}{P})$.

$[\mathbf{X}', \Delta'_2] = \text{SLS}(\mathbf{X}', V, \Delta')$

1. Initialize $l = 1$
2. For q = 1 ***to*** Q
   (a) Identify subset $\mathbf{X}_q$ of $\mathbf{X}'$ using Δ'
   (b) Set $\mathbf{a}_l = \phi(\mathbf{x}_t)$, where $\mathbf{x}_t \in \underset{i}{\textbf{argmax}}\ \|\mathbf{x}_i\|^2, \mathbf{x}_i \in \mathbf{X}_q$
   (c) Compute distance d_i in kernel space of all $\mathbf{x}_i \in \mathbf{X}_q$ from $\mathbf{a}_l$
   (d) Select $\mathbf{x}_k$ such that, there exists V data vectors $\mathbf{x}_i \in \mathbf{X}_q$ with $d_i < d_k$, using the BFPRT algorithm
   (e) Using $\mathbf{x}_k$, rearrange $\mathbf{X}_q$ as $\mathbf{X}_q = \{\mathbf{X}^1, \mathbf{X}^2\}$, where $\mathbf{X}^1 = \{\mathbf{x}_i : d_i < d_k, \mathbf{x}_i \in \mathbf{X}_q\}$ and $\mathbf{X}^2 = \{\mathbf{x}_i : \mathbf{x}_i \in \mathbf{X}_q \text{ and } \mathbf{x}_i \notin \mathbf{X}^1\}$
   (f) For i where $\mathbf{x}_i \in \mathbf{X}^1$, set $\delta_i^2 =$ index of last data vector in $\mathbf{X}^1$, where δ_i^2 is the i^{th} element of Δ'_2
   (g) Remove $\mathbf{X}^1$ from $\mathbf{X}_q$
   (h) If $|\mathbf{X}^2| > V$

      Set: $l = l + 1$ and $\mathbf{a}_l = \mathbf{x}_k$

      Repeat steps 2.c to 2.h
   (i) If $|\mathbf{X}^2| \leq V$

      For i where $\mathbf{x}_i \in \mathbf{X}^2$, set $\delta_i^2 =$ index of last data vector in $\mathbf{X}^2$

4.3 Computation of the Approximate Extreme Points

After computing the subsets $\mathbf{X}_{q_r}$, the algorithm DeriveAE is applied to each $\mathbf{X}_{q_r}$ to compute its approximate extreme points. The algorithm DeriveAE is described below. DeriveAE uses three routines. SphereSet($\mathbf{X}_{q_r}$) returns all $\mathbf{x}_i \in \mathbf{X}_{q_r}$ that lie on the surface of the smallest hypersphere in kernel space that contains $\mathbf{X}_{q_r}$. It computes the hypersphere as a hard margin support vector data descriptor (SVDD) (Tax and Duin, 2004). SphereSort($\mathbf{X}_{q_r}$) returns data vectors $\mathbf{x}_i \in \mathbf{X}_{q_r}$ sorted in descending order of distance in the kernel space from the center of the SVDD hypersphere. CheckPoint($\mathbf{x}_i, \Psi$) returns TRUE if $\mathbf{x}_i$ is an approximate extreme point of the set Ψ in kernel space. The operator $A \backslash B$ indicates a set operation that returns the set of the members of A excluding $A \cap B$. The matrix $\mathbf{X}^*_{q_r}$ contains the approximate extreme points of $\mathbf{X}_{q_r}$ and $\overline{\beta_{q_r}}$ is a $|\mathbf{X}^*_{q_r}|$ sized vector.

$[\mathbf{X}^*_{q_r}, \overline{\beta_{q_r}}] = \text{DeriveAE}(\mathbf{X}_{q_r})$

---

1. Initialize: $\mathbf{X}^*_{q_r} = \text{SphereSet}(\mathbf{X}_{q_r})$ and $\Psi = \emptyset$
2. Set $\zeta = \text{SphereSort}(\mathbf{X}_{q_r} \backslash \mathbf{X}^*_{q_r})$
3. For each $\mathbf{x}_i$ taken in order from ζ, call the routine CheckPoint($\mathbf{x}_i, \mathbf{X}^*_{q_r} \cup \Psi$)

   If it returns $FALSE$, then set $\Psi = \Psi \cup \mathbf{x}_i$
4. Initialize a matrix Γ of size $|\mathbf{X}_{q_r}| \text{x} |\mathbf{X}^*_{q_r}|$ with all elements set to 0

   Set $\mu_{k,k} = 1 \; \forall \mathbf{x}_k \in \mathbf{X}^*_{q_r}$, where $\mu_{i,j}$ is the element in the i^{th} row and j^{th} column of Γ
5. For each $\mathbf{x}_i \in \mathbf{X}_{q_r}$ and $\mathbf{x}_i \notin \mathbf{X}^*_{q_r}$, execute CheckPoint($\mathbf{x}_i, \mathbf{X}^*_{q_r}$)

   Set the i^{th} row of $\Gamma = \overline{\mu_i}$, where $\overline{\mu_i}$ is the result of CheckPoint($\mathbf{x}_i, \mathbf{X}^*_{q_r}$)
6. For j = 1 ***to*** $|\mathbf{X}^*_{q_r}|$

   Set $\beta^j_{q_r} = \sum_{k=1}^{|\mathbf{X}_{q_r}|} \mu_{k,j}$

CheckPoint($\mathbf{x}_i, \Psi$) is computed by solving the following quadratic optimization problem:

$$\min_{\overline{\mu_i}} \; p(\mathbf{x}_i, \Psi) = \|\phi(\mathbf{x}_i) - \sum_{t=1}^{|\Psi|} \mu_{i,t} \phi(\mathbf{x}_t)\|^2,$$
$$\text{s.t. } \mathbf{x}_t \in \Psi, 0 \leq \mu_{i,t} \leq 1 \text{ and } \sum_{t=1}^{|\Psi|} \mu_{i,t} = 1,$$

where $\|\phi(\mathbf{x}_i) - \sum_{t=1}^{|\Psi|} \mu_{i,t} \phi(\mathbf{x}_t)\|^2 = K(\mathbf{x}_t, \mathbf{x}_t) + \sum_{t=1}^{|\Psi|} \sum_{s=1}^{|\Psi|} \mu_{i,t} \mu_{i,s} K(\mathbf{x}_t, \mathbf{x}_s) - 2 \sum_{t=1}^{|\Psi|} \mu_{i,t} K(\mathbf{x}_i, \mathbf{x}_t)$. If the optimized value of $p(\mathbf{x}_i, \Psi) \leq \epsilon$, CheckPoint($\mathbf{x}_i, \Psi$) returns TRUE and otherwise it returns FALSE. It can be seen that the formulation of $p(\mathbf{x}_i, \Psi)$ is similar to (6). The value of $\overline{\mu_i}$ computed by CheckPoint($\mathbf{z}_i, \Psi_0$), is used in step 5 of DeriveAE.

Now we compute the time complexity of DeriveAE. We use the fact that the optimization problem in CheckPoint($\mathbf{x}_i, \Psi$) is essentially the same as the dual optimization problem of SVM given in (3). Since DeriveAE solves several SVM training problems in steps 1,3, and 5, it is necessary to know the training time complexity of a SVM. As any SVM solver method can be used, we denote the training time complexity of each step of DeriveAE that solves an SVM problem as $O(S(A_{q_r}))$. Here A_{q_r} is the largest value of $\mathbf{X}^*_{q_r} \cup \Psi$ during the run of DeriveAE($\mathbf{X}_{q_r}$). This enables us to derive a generic expression for the complexity of DeriveAE, independent of the SVM solver method used. Hence the time complexity of step 1 is $O(S(A_{q_r}))$. The time complexity of steps 3 and 5 are $O(V\ S(A_{q_r}))$ and $O(A_{q_r}\ S(A_{q_r}))$ respectively. The time complexity of step 2 is $O(V\ |\Psi_1| + V\ \log_2 V)$, where $\Psi_1 = \text{SphereSet}(\mathbf{X}_{q_r})$. Hence the time complexity of DeriveAE is $O(V\ |\Psi_1| + V\ \log_2 V + V\ S(A_{q_r})$. Since $|\Psi_1|$ is typically very small, we denote the time complexity of DeriveAE by $O(V\ \log_2 V + V\ S(A_{q_r}))$. For SMO based implementations of DeriveAE, such as the implementation we used for Section 5, typically $S(A_{q_r}) = O(A^2_{q_r})$.

4.4 Combining All the Methods to Compute X^*

To derive X^*, it is required to first rearrange $\mathbf{X}$, so that data vectors from each class are grouped together as $\mathbf{X} = \{\mathbf{X}^+, \mathbf{X}^-\}$. Here $\mathbf{X}^+ = \{\mathbf{x}_i : y_i = 1, \mathbf{x}_i \in \mathbf{X}\}$ and $\mathbf{X}^- = \{\mathbf{x}_i : y_i = -1, \mathbf{x}_i \in \mathbf{X}\}$. Then the selected segregation methods are run on $\mathbf{X}^+$ and $\mathbf{X}^-$ separately. The algorithm DeriveRS given below, combines all the algorithms defined earlier in this section with a few additional steps, to compute the representative set of $\mathbf{X}$. The complexity of DeriveRS[2] can easily be computed by summing the complexities of its steps. The complexity of steps 1 and 6 is O(N). The complexity of step 2 is $O(N)$ if FLS1 is run or $O(N\ \log_2 \frac{N}{P})$ if FLS2 is run. In step 3, the $O(\frac{NP}{V})$ method SLS is run. In steps 4 and 5, DeriveAE is run on all the subsets $\mathbf{X}_{q_r}$ giving a total complexity of $O(N\ \log_2 V + V \sum_{q=1}^{Q} \sum_{r=1}^{R} S(A_{q_r}))$. Here we use the fact that the number of subsets $\mathbf{X}_{q_r}$ is $O(\frac{N}{V})$. Thus the complexity of DeriveRS is $O(N(\frac{P}{V} + \log_2 V) + V \sum_{q=1}^{Q} \sum_{r=1}^{R} S(A_{q_r}))$ when FLS1 is used and $O(N(\log_2 \frac{N}{P} + \frac{P}{V} + \log_2 V) + V \sum_{q=1}^{Q} \sum_{r=1}^{R} S(A_{q_r}))$ when FLS2 is used.

5. Experiments

We focused our experiments on an SMO (Fan et al., 2005) based implementation of AESVM and DeriveRS. We evaluated the classification performance of AESVM using the nine data sets, described below. Next, we present an evaluation of the algorithm DeriveRS, followed by an evaluation of AESVM.

2. We present DeriveRS as one algorithm in spite of its two variants that use FLS1 or FLS2, for simplicity and to conserve space.

$[\mathbf{X}^*, \mathbf{Y}^*, \overline{\beta}]$ = DeriveRS($\mathbf{X}$,$\mathbf{Y}$,P,V)

1. Set $\mathbf{X}^+ = \{\mathbf{x}_i : \mathbf{x}_i \in \mathbf{X}, y_i = 1\}$ and $\mathbf{X}^- = \{\mathbf{x}_i : \mathbf{x}_i \in \mathbf{X}, y_i = -1\}$
2. Run $[\mathbf{X}^+, \Delta^+]$ = FLS($\mathbf{X}^+$,P) and $[\mathbf{X}^-, \Delta^-]$ = FLS($\mathbf{X}^-$,P), where FLS is FLS1 or FLS2
3. Run $[\mathbf{X}^+, \Delta_2^+]$ = SLS($\mathbf{X}^+$,V,Δ^+) and $[\mathbf{X}^-, \Delta_2^-]$ = SLS($\mathbf{X}^-$,V,Δ^-)
4. Using Δ_2^+, identify each subset $\mathbf{X}_{q_r}$ of $\mathbf{X}^+$ and run $[\mathbf{X}^*_{q_r}, \overline{\beta_{q_r}}]$ = DeriveAE($\mathbf{X}_{q_r}$)
   Set N^{+*} = sum of number of data vectors in all $\mathbf{X}^*_{q_r}$ derived from $\mathbf{X}^+$
5. Using Δ_2^-, identify each subset $\mathbf{X}_{q_r}$ of $\mathbf{X}^-$ and run $[\mathbf{X}^*_{q_r}, \overline{\beta_{q_r}}]$ = DeriveAE($\mathbf{X}_{q_r}$)
   Set N^{-*} = sum of number of data vectors in all $\mathbf{X}^*_{q_r}$ derived from $\mathbf{X}^-$
6. Combine in the same order, all $\mathbf{X}^*_{q_r}$ to obtain $\mathbf{X}^*$ and all $\overline{\beta_{q_r}}$ to obtain $\overline{\beta}$
   Set $\mathbf{Y}^* = \{y_i : y_i = 1 \text{ for } i = 1, 2, .., N^{+*}$; and $y_i = -1$ for $i = 1 + N^{+*}, 2 + N^{+*}, .., N^{-*} + N^{+*}\}$

5.1 Data Sets

Nine data sets of varied size, dimensionality and density were used to evaluate DeriveRS and our AESVM implementation. For data sets D2, D3 and D4, we performed five fold cross validation. We did not perform five fold cross-validation on the other data sets, because they have been widely used in their native form with a separate training and testing set.

D1 *KDD'99 intrusion detection data set:*[3] This data set is available as a training set of 4898431 data vectors and a testing set of 311027 data vectors, with forty one features ($D = 41$). As described in Tavallaee et al. (2009), a huge portion of this data set is comprised of repeated data vectors. Experiments were conducted only on the distinct data vectors. The number of distinct training set vectors was $N = 1074974$ and the number of distinct testing set vectors was $N = 77216$. The training set density = 33%.

D2 *Localization data for person activity:*[4] This data set has been used in a study on agent-based care for independent living (Kaluža et al., 2010). It has $N = 164860$ data vectors of seven features. It is comprised of continuous recordings from sensors attached to five people and can be used to predict the activity that was performed by each person at the time of data collection. In our experiments we used this data set to validate a binary problem of classifying the activities 'lying' and 'lying down' from the other activities. Features 3 and 4, that gives the time information, were not used in our experiments. Hence for this data set $D = 5$. The data set density = 96%.

3. D1 is available for download at `http://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data`.
4. D2 is available for download at `http://archive.ics.uci.edu/ml/datasets/Localization+Data+for+Person+Activity`.

D3 *Seizure detection data set*: This data set has $N = 982863$ data vectors, three features ($D = 3$) and density = 100%. It is comprised of continuous EEG recordings from rats induced with status epilepticus and is used to evaluate algorithms that classify seizure events from seizure-free EEG. An important characteristic of this data set is that it is highly unbalanced, the total number of data vectors corresponding to seizures is minuscule compared to the remaining data. Details of the data set can be found in Nandan et al. (2010), where it is used as data set A.

D4 *Forest cover type data set*:[5] This data set has $N = 581012$ data vectors and fifty four features ($D = 54$) and density = 22%. It is used to classify the forest cover of areas of 30mx30m size into one of seven types. We followed the method used in Collobert et al. (2002), where a classification of forest cover type 2 from the other cover types was performed.

D5 *IJCNN1 data set*:[6] This data set was used in IJCNN 2001 generalization ability challenge (Chang and Lin, 2001). The training set and testing set have 49990 ($N = 49990$) and 91701 data vectors respectively. It has 22 features ($D = 22$) and training set density = 59%

D6 *Adult income data set*:[7] This data set derived from the 1994 Census database, was used to classify incomes over $50000 from those below it. The training set has $N = 32561$ with $D = 123$ and density = 11%, while the testing set has 16281 data vectors. The data is pre-processed as described in Platt (1999).

D7 *Epsilon data set*:[8] This is a data set that was used for 2008 Pascal large scale learning challenge and in Yuan et al. (2011). It is comprised of 400000 data vectors that are 100% dense with $D = 2000$. Since this is too large for our experiments, we used the first 10% of the training set[9] giving $N = 40000$. The testing set has 100000 data vectors.

D8 *MNIST character recognition data set*:[10] The widely used data set (Lecun et al., 1998) of hand written characters has a training set of $N = 60000$, $D = 780$ and density = 19%. We performed the binary classification task of classifying the character '0' from the others. The testing set has 10000 data vectors.

5. D4 is available for download at `http://archive.ics.uci.edu/ml/datasets/Covertype`.
6. D5 is available for download at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#ijcnn1`.
7. D6 is available for download at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#a9a`.
8. D7 is available for download at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#epsilon`.
9. AESVM and the other SVM solvers are fully capable of training on this data set. However, the excessive training time makes it impractical to train the solvers on the entire data set for this paper.
10. D8 is available for download at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#mnist`.

D9 *w8a data set*:[11] This artificial data set used in Platt (1999) was randomly generated and has $D = 300$ features. The training set has $N = 49749$ with a density = 4% and the testing set has 14951 data vectors.

5.2 Evaluation of DeriveRS

We began our experiments with an evaluation of the algorithm DeriveRS, described in Section 4. The performance of the two methods FLS1 and FLS2 were compared first. DeriveRS was run on D1, D2, D4 and D5 with the parameters $P = 10^4, V = 10^3, \epsilon = 10^{-2}$, and $g = [2^{-4}, 2^{-3}, 2^{-2}, ..., 2^2]$, first with FLS1 and then FLS2. For D2, DeriveRS was run on the entire data set for this particular experiment, instead of performing five fold cross-validation. This was done because, D2 is a small data set and the difference between the two first level segregation methods can be better observed when the data set is as large as possible. The relatively small value of $P = 10^4$ was also chosen considering the small size of D2 and D5. To evaluate the effectiveness of FLS1 and FLS2, we also ran DeriveRS with FLS1 and FLS2 after randomly reordering each data set. The results are shown in Figure 1.

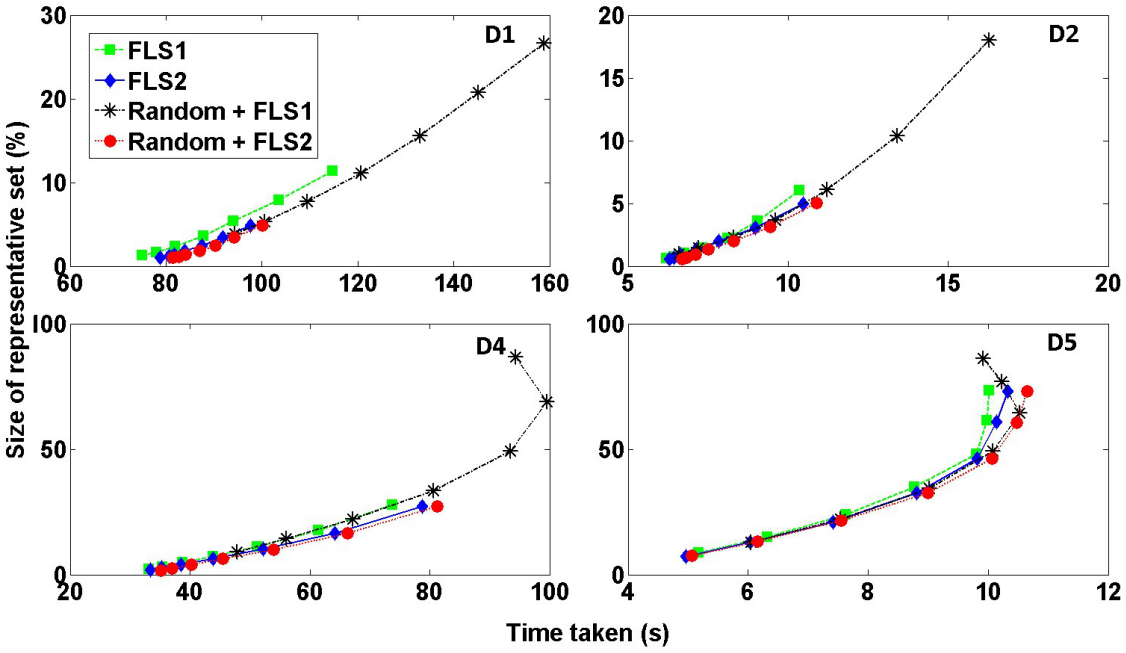


Figure 1: Performance of variants of DeriveRS with $g = [2^{-4}, 2^{-3}, 2^{-2}, ..., 2^2]$, for data sets D1, D2, D4, and D5. The results of DeriveRS with FLS1 and FLS2, after randomly reordering the data sets are shown as Random+FLS1 and Random+FLS2, respectively

11. D9 is available for download at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#w8a`.

For all data sets, FLS2 gave smaller representative sets than FLS1. For D1, DeriveRS with FLS2 was significantly faster and gave much smaller results than FLS1. For D2, D4 and D5, even though the representative sets derived by FLS1 and FLS2 are almost equal in size, FLS1 took noticeably less time. The results of DeriveRS obtained after randomly rearranging the data sets, indicate the utility of FLS2. For all the data sets, the results of FLS2 after random reordering was seen to be significantly better than the results of FLS1 after random rearrangement. Hence we can infer that the good results obtained with FLS2 are not caused by any pre-existing order in the data sets. A sharp increase was observed in representative set sizes and computation times for FLS1, when the data sets were randomly rearranged.

Next we investigated the impact of changes in the values of the parameters P and V on the performance of DeriveRS. All combinations of $P = \{10^4, 5\text{x}10^4, 10^5, 2\text{x}10^5\}$ and $V = \{10^2, 5\text{x}10^2, 10^3, 2\text{x}10^3, 3\text{x}10^3\}$ were used to compute the representative set of D1. The computations were performed for $\epsilon = 10^{-2}$ and $g = 1$. The method FLS2 was used for the first level segregation in DeriveRS. The results are shown in Table 1. As expected for an algorithm of time complexity $O(N(\log_2 \frac{N}{P} + \frac{P}{V} + \log_2 V) + V \sum_{q=1}^{Q} \sum_{r=1}^{R} S(A_{q_r}))$, the computation time was generally observed to increase for an increase in the value of V or P. It should be noted that our implementation of DeriveRS was based on SMO and hence $S(A_{q_r}) = O(A_{q_r}^2)$. In some cases the computation time decreased when P or V increased. This is caused by a decrease in the value of $O(\sum_{q=1}^{Q} \sum_{r=1}^{R} A_{q_r}^2)$, which is inferred from the observed decrease of the size of the representative set M ($M \approx \sum_{q=1}^{Q} \sum_{r=1}^{R} A_{q_r}$). A sharp decrease in M was observed when V was increased. The impact of increasing P on the size of the representative set was found to be less drastic. This observation indicates that DeriveAE selects fewer approximate extreme points when V is larger.

$\frac{M}{N}$x100% (Computation time in seconds)					
P	$V = 10^2$	$V = 5\text{x}10^2$	$V = 10^3$	$V = 2\text{x}10^3$	$V = 3\text{x}10^3$
10^4	7(27)	3(51)	2.5(87)	2.2(161)	2.1(233)
$5\text{x}10^4$	6.9(66)	2.9(59)	2.4(92)	2.1(166)	2(239)
10^5	7(121)	2.9(69)	2.3(98)	2.1(169)	1.9(248)
$2\text{x}10^5$	6.9(237)	2.9(94)	2.3(110)	2(176)	1.9(250)

Table 1: The impact of varying P and V on the result of DeriveRS

As described in Section 5.3, we compared several SVM training algorithms with our implementation of AESVM. We performed a grid search with all combinations of the SVM hyper-parameters $C' = \{2^{-4}, 2^{-3}, ..., 2^6, 2^7\}$ and $g = \{2^{-4}, 2^{-3}, 2^{-2}, ..., 2^1, 2^2\}$. The hyper-parameter C' is related to the hyper-parameter C as $C' = \frac{C}{N}$. We represent the grid in terms of C' as it is used in several SVM solvers such as LIBSVM, LASVM, CVM and BVM. Furthermore, the use of C' enables the application of the same hyper-parameter grid to all data sets. To train AESVM with all the hyper-parameter combinations in the grid,

the representative set has to be computed using DeriveRS for all values of kernel hyper-parameter g in the grid. This is because the kernel space varies when the value of g is varied. For all the computations, the input parameters were set as $P = 10^5$ and $V = 10^3$. The first level segregation in DeriveRS was performed using FLS2. Three values of the tolerance parameter ϵ were investigated, $\epsilon = 10^{-2}, 10^{-3}$ or 10^{-4}.

The results of the computation for data sets D1 - D5, are shown in the Table 2. The percentage of data vectors in the representative set was found to increase with increasing values of g. This is intuitive, as when g increases the distance between the data vectors in kernel space increases. With increased distances, more data vectors $\mathbf{x}_i$ become approximate extreme points. The increase in the number of approximate extreme points with g causes the rising trend of computation time shown in Table 2. For a decrease in the value of ϵ, M increases. This is because, for smaller ϵ fewer $\mathbf{x}_i$ would satisfy the condition: optimized $p(\mathbf{x}_i, \Psi) \leq \epsilon$ in CheckPoint$(\mathbf{x}_i, \Psi)$. This results in the selection of a larger number of approximate extreme points in DeriveAE.

$\frac{M}{N}$x100% (Computation time in seconds)								
ϵ	Data set	g = $\frac{1}{2^4}$	g = $\frac{1}{2^3}$	g = $\frac{1}{2^2}$	g = $\frac{1}{2}$	g = 1	g = 2^1	g = 2^2
10^{-2}	D1	0.9(139)	1(138)	1.3(140)	1.7(147)	2.4(151)	3.3(157)	4.6(163)
	D2	0.6(12)	0.7(13)	0.8(13)	1.2(13)	1.8(14)	2.8(15)	4.7(17)
	D3	0.6(79)	0.6(80)	0.6(80)	0.6(79)	0.6(79)	0.6(79)	0.6(78)
	D4	1.3(55)	1.9(58)	3.1(61)	5.1(68)	8.5(78)	14.5(91)	25.2(111)
	D5	5.6(7)	10.4(8)	17.7(10)	28.1(12)	42.1(14)	58(15)	71(15)
10^{-3}	D1	1.6(142)	2.2(149)	3(160)	4.2(168)	6(188)	8.5(208)	12.1(231)
	D2	1.3(13)	1.8(14)	2.6(16)	3.8(19)	5.7(23)	8.8(29)	14.4(35)
	D3	0.6(80)	0.6(79)	0.6(79)	0.6(79)	0.5(80)	0.5(80)	0.6(81)
	D4	5.5(71)	8.6(86)	13(106)	19.9(136)	31.1(172)	48.7(203)	71.3(204)
	D5	25.8(15)	36.4(19)	49.5(22)	63.5(23)	76.2(22)	86.1(21)	93.5(19)
10^{-4}	D1	3.8(189)	5.4(217)	7.7(253)	10.9(304)	15.2(358)	20.4(418)	26.8(479)
	D2	3.8(21)	5.1(28)	6.9(40)	9.6(52)	14.3(61)	22.8(79)	35.8(100)
	D3	0.5(78)	0.5(79)	0.5(80)	0.6(81)	0.7(83)	0.9(86)	1.2(90)
	D4	19.4(175)	27.1(249)	38.1(333)	54.3(394.3)	75.5(387)	92.6(310)	98.8(244)
	D5	56.9(40)	69.1(43)	80.1(41)	88.6(38)	94.9(32)	98.3(26)	99.7(22)

Table 2: The percentage of the data vectors in $\mathbf{X}^*$ (given by $\frac{M}{N}$x100) and its computation time for data sets D1-D5

The results of applying DeriveRS to the high-dimensional data sets D6-D9 are shown in Table 3. It was observed that $\frac{M}{N}$ was much larger for D6-D9 than for the other data sets. We computed the representative set with $\epsilon = 10^{-2}$ only, as for smaller values of ϵ we expect the representative set to be close to 100% of the training set. The increasing trend of the size of the representative set with increasing g values can be observed in Table 3 also.

$\frac{M}{N}$x100% (Computation time in seconds)							
Data set	g = $\frac{1}{2^4}$	g = $\frac{1}{2^3}$	g = $\frac{1}{2^2}$	g = $\frac{1}{2}$	g = 1	g = 2^1	g = 2^2
D6	83.1(12)	83.1(12)	83.1(13)	83.1(12)	83.1(9)	82.7(9)	86(9)
D7	97.2(317)	99.7(309)	100(325)	100(332)	100(360)	100(330)	100(280)
D8	100(97)	100(75)	100(62)	100(63)	100(67)	100(64)	100(64)
D9	72.2(21)	72.2(22)	72.2(21)	72.7(17)	72.8(15)	74.4(14)	76.1(15)

Table 3: The percentage of data vectors in $\mathbf{X}^*$ and its computation time for data sets D6-D9 with $\epsilon = 10^{-2}$

5.3 Comparison of AESVM to SVM Solvers

To judge the accuracy and efficiency of AESVM, its classification performance was compared with the SMO implementation in LIBSVM, ver. 3.1. We chose LIBSVM because it is a state-of-the-art SMO implementation that is routinely used in similar comparison studies. To compare the efficiency of AESVM to other popular approximate SVM solvers we chose CVM, BVM, LASVM, SVM$^{\text{perf}}$, and RfeatSVM. A description of these methods is given in Section 2. We chose these methods because they are widely cited, their software implementations are freely available and other studies (Shalev-Shwartz et al., 2011) have reported fast SVM training using some of these methods. LASVM is also an efficient method for online SVM training. However, since we do not investigate online SVM learning in this paper, we did not test the online SVM training performance of LASVM. We compared AESVM with CVM and BVM even though they are L2-SVM solvers, as they has been reported to be faster alternatives to SVM implementations such as LIBSVM.

The implementation of AESVM and DeriveRS were built upon the LIBSVM implementation. All methods except SVM$^{\text{perf}}$ were allocated a cache of size 600 MB. The parameters for DeriveRS were $P = 10^5$ and $V = 10^3$, and the first level segregation was performed using FLS2. To reflect a typical SVM training scenario, we performed a grid search with all eighty four combinations of the SVM hyper-parameters $C' = \{2^{-4}, 2^{-3}, ..., 2^6, 2^7\}$ and $g = \{2^{-4}, 2^{-3}, 2^{-2}, ..., 2^1, 2^2\}$. As mentioned earlier, for data sets D2, D3 and D4, five fold cross-validation was performed. The results of the comparison have been split into sub-sections given below, due to the large number of SVM solvers and data sets used.

5.3.1 Comparison to CVM, BVM, LASVM and LIBSVM

First we present the results of the performance comparison for D2 in Figures 2 and 3. For ease of representation, only the results of grid points corresponding to combinations of $C' = \{2^{-4}, 2^{-2}, 1, 2^2, 2^4, 2^6\}$ and $g = \{2^{-4}, 2^{-2}, 1, 2^2\}$ are shown in Figures 2 and 3. Figure 2 shows the graph between training time and classification accuracy for the five algorithms. Figure 3 shows the graph between the number of support vectors and classification accuracy. We present classification accuracy as the ratio of the number of correct classifications to the total number of classifications performed. Since the classification time of an SVM algorithm is directly proportional to the number of support vectors, we represent it in terms of the

number of support vectors. It can be seen that, AESVM generally gave more accurate results for a fraction of the training time of the other algorithms, and also resulted in less classification time. The training time and classification times of AESVM increased when ϵ was reduced. This is expected given the inverse relation of M to ϵ shown in Tables 2 and 3. The variation in accuracy with ϵ is not very noticeable.

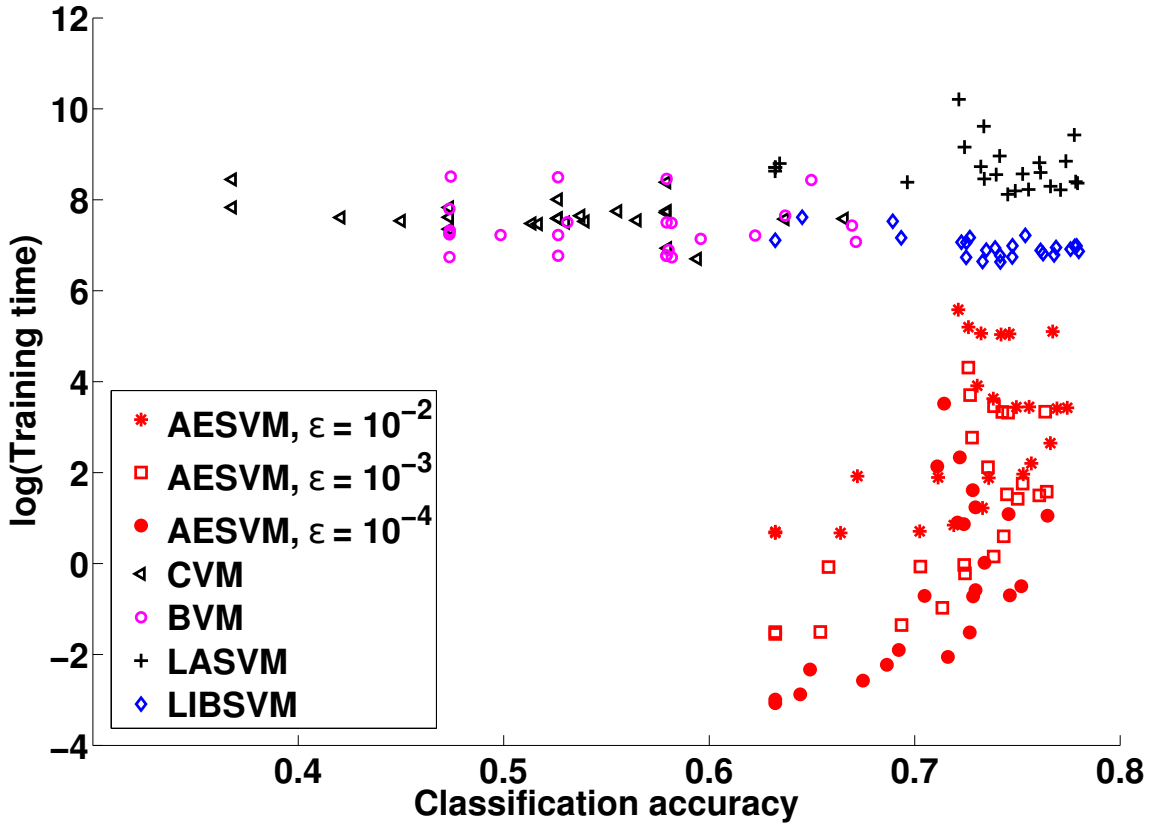


Figure 2: Plot of training time against classification accuracy of the SVM algorithms on D2

Figures 2 and 3 indicate that AESVM gave better results than the other algorithms for SVM training and classification on D2, in terms of standard metrics. To present a more quantitative and easily interpretable comparison of the algorithms, we define the seven performance metrics given below. These metrics combine the results of all runs of each algorithm into a single value, for each data set. For the first five metrics, we take LIBSVM as a baseline of comparison, as it gives the most accurate solution among the tested methods. Furthermore, an important objective of these experiments is to show the similarity of the results of AESVM and LIBSVM. In the description given below, $\mathbb{F}$ can refer to any SVM algorithm such as AESVM, CVM, LASVM etc.

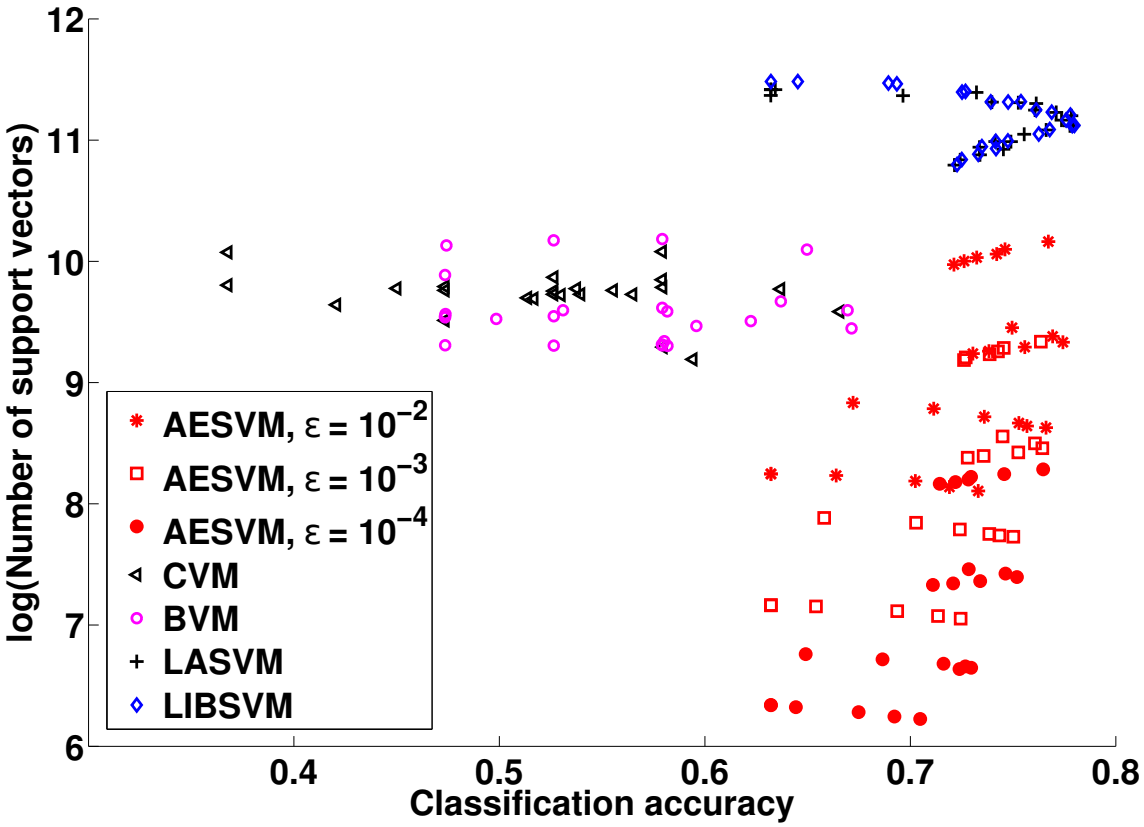


Figure 3: Plot of classification time, represented by the number of support vectors, against classification accuracy of the SVM algorithms on D2

1. *Expected training time speedup, ETS*: The expected speedup in training time is indicated by:

$$ETS = \frac{1}{RS}\sum_{r=1}^{R}\sum_{s=1}^{S}\frac{TL_s^r}{T\mathbb{F}_s^r}.$$

Here TL_s^r and $T\mathbb{F}_s^r$ are the training times of LIBSVM and $\mathbb{F}$ respectively, in the s^{th} cross-validation fold with the r^{th} set of hyper-parameters of grid search.

2. *Overall training time speedup, OTS*: It indicates overall training time speedup for the entire grid search with cross-validation, including the time taken to compute the representative set. The total time taken by DeriveRS to compute the representative set for all values of g is represented as $\mathbb{TX}^*$. For methods other than AESVM and RfeatSVM2 (see Section 5.3.3), $\mathbb{TX}^* = 0$.

$$OTS = \frac{\sum_{r=1}^{R}\sum_{s=1}^{S}TL_s^r}{\sum_{r=1}^{R}\sum_{s=1}^{S}T\mathbb{F}_s^r + \mathbb{TX}^*}.$$

3. *Expected classification time speedup, ECS*: The expected speedup in classification time is indicated by:
$$ECS = \frac{1}{RS}\sum_{r=1}^{R}\sum_{s=1}^{S}\frac{NL_s^r}{N\mathbb{F}_s^r}.$$
Here NL_s^r and $N\mathbb{F}_s^r$ are the number of support vectors in the solution of LIBSVM and $\mathbb{F}$ respectively.

4. *Classification time speedup for optimal hyper-parameters, CTS*: The speedup in classification time for the optimal hyper-parameters (hyper-parameters that result in maximum classification accuracy) chosen by grid search is indicated by:
$$CTS = \frac{\mathbf{max}_r \sum_{s=1}^{S} NL_s^r}{\mathbf{max}_r \sum_{s=1}^{S} N\mathbb{F}_s^r}.$$

5. *Root mean squared error of classification accuracy, RMSE*: The similarity of the solution of $\mathbb{F}$ to LIBSVM, in terms of its classification accuracy, is indicated by:
$$RMSE = \left(\frac{1}{RS}\sum_{r=1}^{R}\sum_{s=1}^{S}(CL_s^r - C\mathbb{F}_s^r)^2\right)^{0.5}.$$
Here CL_s^r and $C\mathbb{F}_s^r$ are the classification accuracy of LIBSVM and $\mathbb{F}$ respectively.

6. *Maximum classification accuracy*: It gives the best classification results of an SVM solver, for the set of SVM hyper-parameters that are tested.
$$\text{max. acc.} = \mathbf{max}_r \frac{1}{S}\sum_{s=1}^{S} C\mathbb{F}_s^r.$$

7. *Mean and standard deviation of classification accuracies*: It indicates the classification performance of an SVM solver, that can be expected for arbitrary hyper-parameter values.
$$\text{mean acc.} = \frac{1}{RS}\sum_{r=1}^{R}\sum_{s=1}^{S} C\mathbb{F}_s^r, \text{ and } \quad \text{std. acc.} = \sqrt{\frac{1}{R}\sum_{r=1}^{R}\left(\frac{1}{S}\sum_{s=1}^{S} C\mathbb{F}_s^r - \text{mean acc.}\right)^2}.$$

The results of the classification performance comparison on data sets D1-D5, are shown in Table 4. It was observed that for all tested values of ϵ, AESVM resulted in large reductions in training and classification times when compared to LIBSVM for a very small difference in classification accuracy. Most notably, for D3 the expected and overall training time speedups were 41728.8 and 488.5 respectively, which is outstanding. Comparing the results of AESVM for different ϵ values, we see that $RMSE$ generally improves by decreasing when ϵ decreases, while the metrics improve by increasing when ϵ increases. The increase in ETS and OTS is of a larger order than the increase in $RMSE$ when ϵ increases.

Data set	Solver	*ETS*	*OTS*	*ECS*	*CTS*	*RMSE* (x10^2)	max. acc. (x10^2)	mean & std. acc. (x10^2)
D1	AESVM1	**1188.9**	**156**	**5.8**	**3.3**	0.22	94.2	92.4, 0.8
	AESVM2	314.8	50.4	3.8	2.6	0.14	93.6	92.3, 0.7
	AESVM3	72.7	14.7	2.4	1.8	**0.06**	93.8	92.4, 0.8
	CVM	8.9	6.2	1.2	2.3	0.44	94.1	**92.7**, 0.8
	BVM	28.6	21.6	2	1.9	0.6	**94.4**	92.6, 0.9
	LASVM	0.8	0.8	1.1	1	0.12	94.3	92.5, 0.8
	LIBSVM						93.9	92.4, 0.8
D2	AESVM1	**6067.6**	**134.5**	**77.7**	**17.8**	3.85	76.5	71.1, 3.3
	AESVM2	1202.5	86.1	29	9.4	2.43	76.7	72.4, 3.6
	AESVM3	164.5	21.8	10.9	6.2	**1.73**	77.4	73.1, 3.6
	CVM	0.7	0.5	4.7	4.3	26.59	70.3	52.2, 0.8
	BVM	0.8	0.5	5	5.6	24.06	67.1	54.6, 0.7
	LASVM	0.2	0.1	1	1	2.18	78.1	73.5, 0.5
	LIBSVM						**78.2**	**74.1**, 3.5
D3	AESVM1	**41728.8**	**488.5**	**71.5**	**64.4**	0.2	99.9	**99.8**, 0.1
	AESVM2	21689.3	468	39.5	51.5	0.1	99.9	**99.8**, 0.1
	AESVM3	12792	429.9	17.1	36	**0.09**	99.9	**99.8**, 0.1
	CVM	60.4	23.9	0.4	0.1	0.33	99.9	**99.8**, 0.2
	BVM	76.8	22.8	0.6	0.2	0.39	99.9	**99.8**, 0.2
	LASVM	0.9	0.5	0.6	0.7	55.2	99.9	69.3, 29.9
	LIBSVM						99.9	**99.8**, 0.1
D4	AESVM1	**962**	**34.6**	**24.5**	**72.8**	1.5	**68.3**	**61.6**, 3.1
	AESVM2	68.8	6.1	6.3	17.1	0.7	68.1	61, 3.3
	AESVM3	6.7	2.3	2.3	5	**0.3**	68.1	60.8, 3.2
	CVM	8	6.2	12.4	28	9.4	63.7	55.5, 3.1
	BVM	6.6	4.4	12.1	8.9	9.44	62.3	54.9, 3.4
	LASVM	-	-	-	-	-	-	-
	LIBSVM						68.2	60.6, 3.2
D5	AESVM1	**26.6**	**4.1**	**3.3**	**1.6**	0.5	98.8	96.2, 2.6
	AESVM2	3.1	1.8	1.5	0.9	0.39	98.9	96.3, 2.6
	AESVM3	1.3	1.1	1.1	0.9	0.25	99	96.4, 2.6
	CVM	0.3	0.2	0.8	0.6	0.74	99	96.6, 2.5
	BVM	0.5	0.3	1	0.9	0.84	99.1	**97**, 2
	LASVM	0.6	0.5	1	1.1	**0.13**	**99.2**	**97**, 2
	LIBSVM						99	96.6, 2.4

Table 4: Performance comparison of AESVM, CVM, BVM, LASVM and LIBSVM on data sets D1-D5. AESVM1, AESVM2 and AESVM3 represent the results of AESVM with $\epsilon = 10^{-2}, 10^{-3},$ and 10^{-4} respectively.

Comparing AESVM to CVM, BVM and LASVM, we see that AESVM in general gave the least values of $RMSE$ and the largest values of ETS, OTS, ECS and CTS. In a few cases LASVM gave low $RMSE$ values. However, in all our experiments LASVM took longer to train than the other algorithms including LIBSVM. *We could not complete the evaluation of LASVM for D4 due to its large training time, which was more than 40 hours for some hyper-parameter combinations.* The five algorithms under comparison were found to give similar maximum classification accuracies for D1, D3 and D5. For D2 and D4, CVM and BVM gave significantly smaller maximum classification accuracies. Another interesting result is that for D3, the mean and standard deviation of classification accuracy of LASVM was found to be widely different from the other algorithms. For all the tested values of ϵ the maximum, mean and standard deviation of the classification accuracies of AESVM were found to be similar.

Next we present the results of performance comparison of CVM, BVM, LASVM, AESVM, and LIBSVM on the high-dimensional data sets D6-D9. As described in Section 5.2, DeriveRS was run with only $\epsilon = 10^{-2}$ for these data sets. The results of the performance comparison are shown in Table 5. *CVM was found to take longer than 40 hours to train on D6, D7 and D8 with some hyper-parameter values and hence we could not complete its evaluation for those data sets. BVM also took longer than 40 hours to train on D7 and it was also not evaluated for D7.* AESVM consistently reported ETS, OTS, ECS and CTS values that are larger than 1 unlike the other algorithms, except for D9 where the CTS value for AESVM was 0.6. However it should be noted that the other methods also had similarly low CTS values for D9. Similar to the results in Table 4, LASVM and BVM resulted in very large $RMSE$ values for some data sets. The maximum classification accuracies of all algorithms were similar. On some data sets, BVM and LASVM were observed to give significantly lower mean and higher standard deviation of classification accuracy.

5.3.2 Comparison to SVM$^{\text{perf}}$

SVM$^{\text{perf}}$ differs from the other SVM solvers in its ability to compute a solution close to the SVM solution for a given number of support vectors (k). The algorithm complexity depends on k as $O(k^2)$ per iteration. We first used a value of $k = 1000$ for our experiments, as it has been reported to give good performance (Joachims and Yu, 2009). SVM$^{\text{perf}}$ was tested on data sets D1, D4, D5, D6, D8 and D9, with the Gaussian kernel[12] and the same hyper-parameter grid as described earlier. The results of the grid search are presented in Table 6. The results of our experiments on AESVM (with $\epsilon = 10^{-2}$) and LIBSVM are repeated in Table 6 for ease of reference. The maximum, mean and standard deviation of classification accuracies are represented as max. acc., mean & std. acc. respectively.

Based on the results obtained for $k = 1000$, other values of k were also tested. For data sets D1, D4 and D5, though SVM$^{\text{perf}}$ gave classification accuracies similar to the that of LIBSVM and AESVM, the training times were similar to or higher than the training times of LIBSVM. To test the ability of SVM$^{\text{perf}}$ to give fast training, we also tested it with $k = 400$ for D1, D4 and D5. For the high dimensional data sets (D6, D8 and D9), the $RMSE$ values were significantly higher for SVM$^{\text{perf}}$, while the mean classification accuracy was noticeably lower than AESVM. Considering the possibility that the value of $k = 1000$ is insufficient to

12. We used the software parameters '-t 2 -w 9 –i 2 –b 0' as suggested in the author's website.

Data set	Solver	ETS	OTS	ECS	CTS	$RMSE$ (x10^2)	max. acc. (x10^2)	mean & std. acc. (x10^2)
D6	AESVM	**1.5**	**1.4**	1.1	**1.2**	**0**	85.1	**81.4**, 2.8
	CVM	-	-	-	-	-	-	-
	BVM	0.6	0.6	**1.5**	**1.2**	7.8	**85.2**	80.2, 8.9
	LASVM	0.8	0.5	1	1.1	0.85	85	81.1, 2.9
	LIBSVM						85.1	**81.4**, 2.8
D7	AESVM	**1**	**1**	**1**	**1.1**	**0.01**	88.3	85.3, 5.7
	CVM	-	-	-	-	-	-	-
	BVM	-	-	-	-	-	-	-
	LASVM	0.9	0.7	**1**	0.9	2.37	88.4	85.2, 6.2
	LIBSVM						**88.6**	**85.7**, 4.8
D8	AESVM	1	1	1	1	**0**	99.7	**92.3**, 3.6
	CVM	-	-	-	-	-	-	-
	BVM	**4.7**	**2.6**	**3.2**	**3.1**	17.55	99.7	88.5, 18.1
	LASVM	1	0.9	1	1	0	99.7	**92.3**, 3.6
	LIBSVM						99.7	**92.3**, 3.6
D9	AESVM	1.4	1.3	1.1	**0.6**	**0**	99.5	98.8, 0.8
	CVM	1.4	1.2	1.8	0.3	1	99.5	**98.9** , 0.8
	BVM	**17.5**	**16.9**	**4.9**	**0.6**	0.09	99.5	**98.9** , 0.8
	LASVM	0.6	0.5	2.3	0.1	27.5	99.5	85.5, 23.9
	LIBSVM						99.5	98.8, 0.8

Table 5: Performance comparison of AESVM (with $\epsilon = 10^{-2}$), CVM, BVM, LASVM and LIBSVM on data sets D6-D9

result in an accurate solution for these data sets, we tested D6 and D9 with $k = 2000$ and D8 with $k = 3000$. Even though the training time increased significantly with an increase in k, the values of $RMSE$ and the mean and standard deviation of accuracies did not improve significantly. The training time speedup values of SVM$^{\text{perf}}$ are much lower than AESVM for all tested k values for all data sets, except for D8. The maximum accuracies of all the algorithms were similar. Due to the ability of SVM$^{\text{perf}}$ to approximate $\mathbf{w}$ with a small set of k vectors, the classification time speedups of SVM$^{\text{perf}}$ are significantly higher than AESVM. However, this approximation comes at the cost of increased training time and sometimes results in a loss of accuracy, as illustrated in Table 6.

5.3.3 Comparison to RfeatSVM

Rahimi and Recht (2007) proposed a promising method to approximate non-linear kernel SVM solutions using simpler linear kernel SVMs. This is accomplished by first projecting the training data set into a randomized feature space and then using any SVM solver with the linear kernel on the projected data set. We first investigated the classification accuracy of the solution of RfeatSVM and its similarity to the SVM solution. LIBSVM with the

Data set	Solver	ETS	OTS	ECS	CTS	$RMSE$ (x10^2)	max. acc. (x10^2)	mean & std. acc. (x10^2)
D1	AESVM	**1188.9**	**156**	5.8	3.3	**0.22**	**94.2**	92.4, 0.8
	SVM$^{\text{perf}}$ k = 400	6.7	1.6	**17**	**6.6**	0.89	93.9	**92.7**, 0.4
	SVM$^{\text{perf}}$ k = 1000	3.7	0.9	2.6	2.6	0.74	94	92.7, 0.5
	LIBSVM						93.9	92.4, 0.8
D4	AESVM	**962**	**34.6**	24.5	72.8	**1.5**	68.3	61.6, 3.1
	SVM$^{\text{perf}}$ k = 400	10.2	3.7	**467.1**	**694.3**	3.7	**68.4**	**62.9**, 2.2
	SVM$^{\text{perf}}$ k = 1000	3.1	1.2	186.8	277.7	2.14	68.1	61.8, 2.7
	LIBSVM						68.2	60.6, 3.2
D5	AESVM	**26.6**	**4.1**	3.3	1.6	0.5	98.8	96.2, 2.6
	SVM$^{\text{perf}}$ k = 400	0.8	0.4	**14.6**	**8.2**	2.9	98.8	96.5, 2.4
	SVM$^{\text{perf}}$ k = 1000	0.2	0.1	5.8	3.3	**0.26**	**99**	**96.7**, 2.4
	LIBSVM						**99**	96.6, 2.4
D6	AESVM	**1.5**	**1.4**	1.1	1.2	**0**	85.1	**81.4**, 2.8
	SVM$^{\text{perf}}$ k = 1000	1.1	0.9	**20**	**12.1**	9.39	**85.2**	79.6, 10.7
	SVM$^{\text{perf}}$ k = 2000	0.3	0.2	10	6	6.5	85.1	80.1, 7.8
	LIBSVM						85.1	**81.4**, 2.8
D8	AESVM	1	1	1	1	**0**	99.7	**92.3**, 3.6
	SVM$^{\text{perf}}$ k = 1000	**37.6**	**23.8**	**49**	**9.9**	54.2	**99.9**	55.7, 42.3
	SVM$^{\text{perf}}$ k = 3000	3.5	1.2	16.3	3.3	51.4	99.8	59.2, 41.6
	LIBSVM						99.7	**92.3**, 3.6
D9	AESVM	**1.4**	**1.3**	1.1	0.6	**0**	**99.5**	**98.8**, 0.8
	SVM$^{\text{perf}}$ k =1000	1.2	0.9	**21.3**	**3**	22.6	99.2	86.1, 18.8
	SVM$^{\text{perf}}$ k =2000	0.4	0.3	10.7	1.5	20.6	99.4	87.3, 17.3
	LIBSVM						**99.5**	**98.8**, 0.8

Table 6: Performance comparison of SVM$^{\text{perf}}$, AESVM (with $\epsilon = 10^{-2}$), and LIBSVM

linear kernel was used to compute the RfeatSVM solution on the projected data sets. This combination of RfeatSVM and LIBSVM is denoted as RfeatSVM1. We used LIBSVM,

in spite of the availability of faster linear SVM implementations, as it is an exact SVM solver. Hence only the performance metrics related to accuracy were used to compare the performance of AESVM, LIBSVM and RfeatSVM1. The random Fourier features method, described in Algorithm 1 of Rahimi and Recht (2007), was used to project the data sets D1, D5, D6 and D9 into a randomized feature space of dimension E.

Data set	Solver	$RMSE$ (x10^2)	max. acc. (x10^2)	mean & std. acc. (x10^2)
D1	AESVM	**0.25**	93.5	92.2,0.9
	RfeatSVM1 E = 100	56.18	37.8	36.1,1.3
	LIBSVM		**93.6**	**92.3**,0.9
D5	AESVM	**0.9**	98.6	95.7,2.8
	RfeatSVM1 E = 100	5.3	94.7	91.6,1.4
	LIBSVM		**98.9**	**96.2** ,2.7
D6	AESVM	**0.16**	**85.1**	81.2,2.9
	RfeatSVM1 E = 1000	4	81.6	78,2.2
	LIBSVM		85	**81.3**,3
D9	AESVM	**0.15**	99.3	98.6,0.8
	RfeatSVM1 E = 1000	0.6	98.7	97.4,0.6
	LIBSVM		**99.5**	**98.8**,0.9

Table 7: Performance comparison of RfeatSVM1 (RfeatSVM solved using LIBSVM), AESVM (with $\epsilon = 10^{-2}$), and LIBSVM

The results of the accuracy comparison are given in Table 7. We used a smaller hyper-parameter grid of all twenty four combinations of $C' = \{2^{-4}, 2^{-2}, 1, 2^2, 2^4, 2^6\}$ and $g = \{2^{-4}, 2^{-2}, 1, 2^2\}$ for our experiments. The results reported in Table 7 for AESVM and LIBSVM were computed for this smaller grid. We selected the number of dimensions (E) of the randomized feature space for D1 and D6 based on Rahimi and Recht (2007). The maximum accuracy for RfeatSVM1 was found to be much less than AESVM and LIBSVM for all data sets. The $RMSE$ values for RfeatSVM1 were significantly higher than AESVM and mean accuracy noticeably lower for most data sets, especially for D1 and D6.

Next we investigated the training and classification time requirements of RfeatSVM by solving it using the fast linear SVM solver LIBLINEAR (Fan et al., 2008), referred to as RfeatSVM2 in the remainder of this paper. The entire hyper-parameter grid used in the previous sections were used in this experiment. The results of the performance comparison of RfeatSVM2, AESVM and LIBSVM are presented in Table 8. The **classification time** shown in Table 8 is the time taken for classification when the SVM solver was trained with

Data set	Solver	ETS	OTS	Classification time (s)	$RMSE$ (x10^2)	max. acc. (x10^2)	mean & std. acc. (x10^2)
D1	AESVM	**1188.9**	**156**	6.1	**0.22**	**94.2**	**92.4**,0.8
	RfeatSVM2 E = 100	176.3	56.4	**0.9**	50.3	63.5	43.7,12.9
	RfeatSVM2 E = 500	77.5	47.7	4.4	43.4	89.3	56,24.1
	LIBSVM			15		93.9	**92.4**,0.8
D5	AESVM	26.6	4.1	9.7	**0.5**	98.8	96.2,2.6
	RfeatSVM2 E = 100	**80.7**	**9.2**	**0.9**	38.6	90.5	64.4,20
	RfeatSVM2 E = 500	33.2	6.5	4.5	30.9	90.5	70.8,15.5
	RfeatSVM2 E = 1000	18.4	3.6	13.8	31.5	90.5	70.2,17.8
	RfeatSVM2 E = 5000	3.9	0.85	64.5	33.8	90.5	70.2,19.8
	LIBSVM			16.8		**99**	**96.6** ,2.4
D6	AESVM	1.5	1.4	16	**0**	**85.1**	**81.4**,2.8
	RfeatSVM2 E = 1000	**205.7**	**43.9**	**2.1**	27.8	75.3	54.9,9.7
	RfeatSVM2 E = 5000	48.8	8.9	10.7	29.1	76.4	53.1,8.1
	RfeatSVM2 E = 10000	24.8	5.1	30.9	28.5	76.4	54,9.2
	LIBSVM			30.5		**85.1**	**81.4**,2.8
D9	AESVM	1.4	1.3	10.5	**0**	**99.5**	**98.8**,0.8
	RfeatSVM2 E = 1000	**245.1**	**50**	**2.9**	36.9	92.8	63.3,9.9
	RfeatSVM2 E = 5000	57.4	12	15.3	39	95.1	61.5,11.2
	RfeatSVM2 E = 10000	28.9	6.5	45.5	37.4	96.3	63.8,12.9
	LIBSVM			5.1		**99.5**	**98.8**,0.8

Table 8: Performance comparison of RfeatSVM2 (RfeatSVM solved using LIBLINEAR), AESVM (with $\epsilon = 10^{-2}$), and LIBSVM

its optimal hyper-parameters. For RfeatSVM2 the classification time includes the time taken to derive the random Fourier features of the test vectors.

The classification time for RfeatSVM2 was generally less than AESVM, for small values of E. Moreover, it was found that RfeatSVM2 has significantly higher training time

speed-ups than AESVM for small values of E, except for D1 where AESVM was much faster. However, with increasing E the classification time and training time increased to more than AESVM for most data sets. For all data sets, the $RMSE$, and maximum, mean and standard deviation of accuracy of RfeatSVM2 were significantly worse than AESVM. Increasing the number of dimensions E, resulted in only a slight improvement in the classification performance of RfeatSVM2. An important observation was that the projected data sets were found to be almost 100% dense, which results in large memory requirements for RfeatSVM1 and RfeatSVM2. Even though, technically the value of E can be increased arbitrarily, its value is practically limited by the memory requirements of RfeatSVM.

5.4 Performance with the Polynomial Kernel

To validate our proposal of AESVM as a fast alternative to SVM for all non-linear kernels, we performed a few experiments with the polynomial kernel, $k(\mathbf{x}_1, \mathbf{x}_2) = (1 + \mathbf{x}_1^T \mathbf{x}_2)^d$. The hyper-parameter grid composed of all twelve combinations of $C' = \{2^{-4}, 2^{-2}, 1, 2^2\}$ and $d = \{2, 3, 4\}$ was used to compute the solutions of AESVM and LIBSVM on the data sets D1, D4 and D6. The results of the computation of the representative set using DeriveRS are shown in Table 9. The parameters for DeriveRS were $P = 10^5$, $V = 10^3$ and $\epsilon = 10^{-2}$, and the first level segregation was performed using FLS2. The performance comparison of AESVM and LIBSVM with the polynomial kernel is shown in Table 10. Like in the case of the Gaussian kernel, we found that AESVM gave results similar to LIBSVM with the polynomial kernel, while taking shorter training and classification times.

$\frac{M}{N}$x100% (Computation time in seconds)			
Data set	d = 2	d = 3	d = 4
D1	8(109)	13.2(199)	26(638)
D4	20.1(67)	48(260.1)	81.3(1166.4)
D6	87.8(11)	84(12.5)	91(13.7)

Table 9: Results of DeriveRS for the polynomial kernel

6. Discussion

AESVM is a new problem formulation that is almost identical to, but less complex than, the SVM primal problem. AESVM optimizes over only a subset of the training data set called the representative set, and consequently, is expected to give fast convergence with most SVM solvers. In contrast, the other studies mentioned in Section 2 are mostly algorithms that solve the SVM primal or related problems. Methods such as RSVM also use different problem formulations. However, they require special algorithms to solve, unlike AESVM. In fact, AESVM can be solved using many of the methods in Section 2. As described in Corollary 5, there are some similarities between AESVM and the Gram matrix approximation methods discussed earlier. It would be interesting to see a comparison of AESVM, with the core set based method proposed by Gärtner and Jaggi (2009). However, due to the

Data set	Solver	*ETS*	*OTS*	*ECS*	*CTS*	*RMSE* (x10^2)	max. acc. (x10^2)	mean & std. acc. (x10^2)
D1	AESVM	21.1	6.4	2.7	2.6	0.13	93.9	93.4, 0.4
	LIBSVM						**94.1**	**93.5**, 0.4
D4	AESVM	7	1.6	2.6	1.9	0.8	**64.9**	**61.2**, 2.7
	LIBSVM						64.5	60.7, 2.5
D6	AESVM	3.8	5.3	1.1	1.1	0.04	**84.6**	**81**, 2.4
	LIBSVM						**84.6**	**81**, 2.3

Table 10: Performance comparison of AESVM (with $\epsilon = 10^{-2}$), and LIBSVM with the polynomial kernel

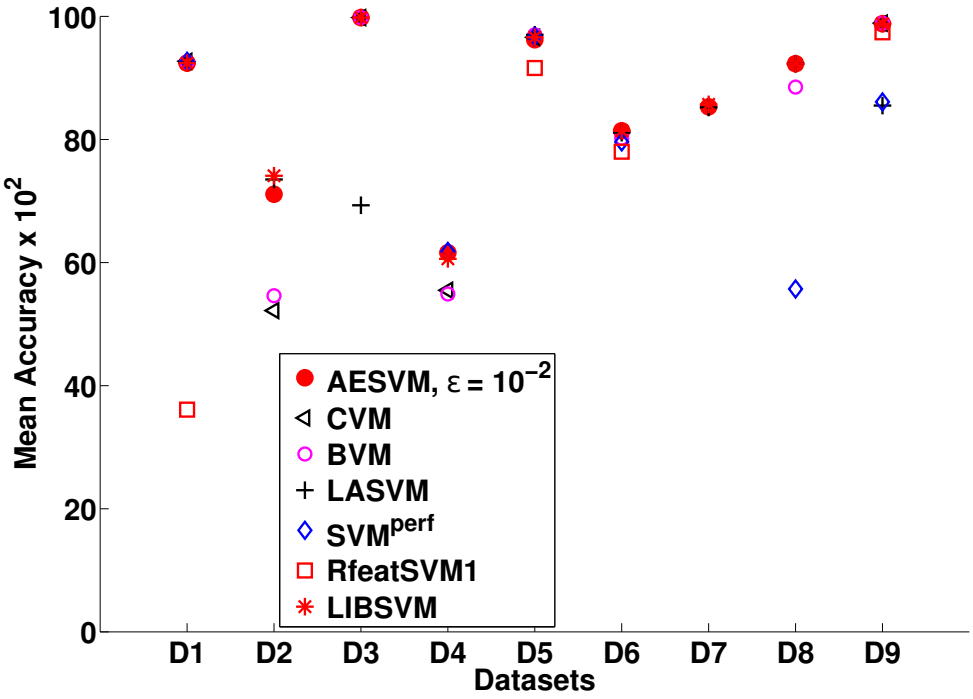


Figure 4: Plot of mean classification accuracy of all SVM solvers

lack of availability of a software implementation and of published results on L1-SVM with non-linear kernels using their approach, the authors find such a comparison study beyond the scope of this paper.

The theoretical and experimental results presented in this paper demonstrate that the solutions of AESVM and SVM are similar in terms of the resulting classification accuracy. A summary of the experiments in Section 5, that compared an SMO based AESVM implementation, CVM, BVM, LASVM, LIBSVM, SVM$^{\text{perf}}$ (with k = 1000) and RfeatSVM1, is presented in Figures 4 to 7. The results of RfeatSVM2 are omitted from Figures 4 to 7, for ease of representation. *It can be seen that AESVM typically gave classification per-*

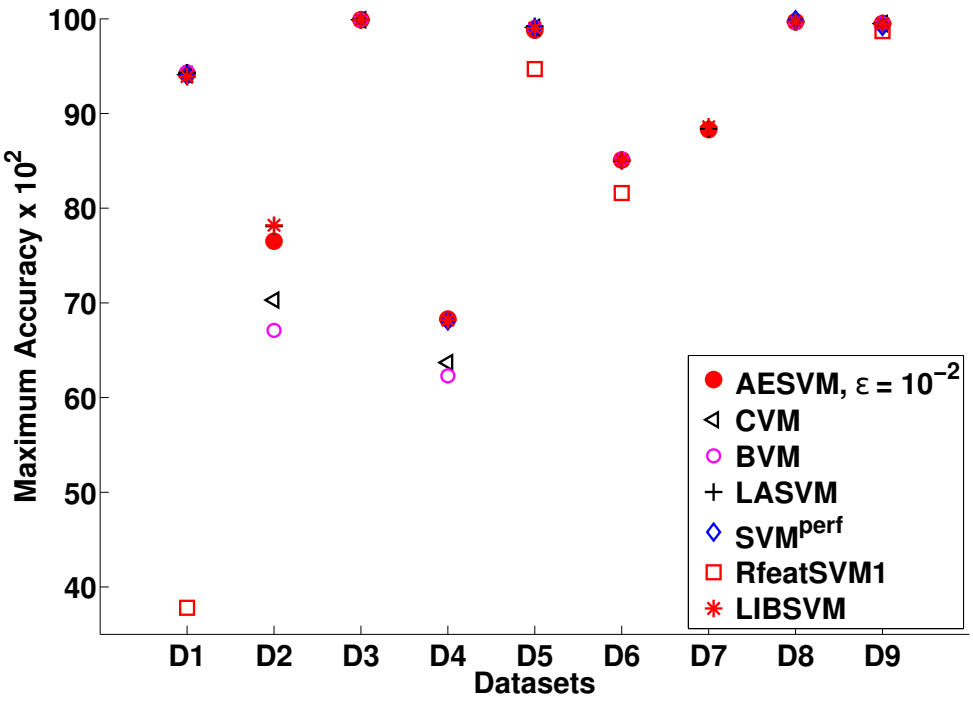


Figure 5: Plot of maximum classification accuracy of all SVM solvers

formance similar to LIBSVM, while giving highest overall training time speedup (OTS). Even though RfeatSVM2 gave higher *OTS* values in some cases, the degradation in classification accuracy was worse than in RfeatSVM1 as shown in Tables 7 and 8. *AESVM also gave competitively high classification time speedup for the optimal hyper-parameters (CTS) in comparison with the other algorithms except SVM^{perf} and RfeatSVM2.* It was found that the maximum classification accuracies of all the algorithms except RfeatSVM1 and RfeatSVM2 were similar. RfeatSVM1 and RfeatSVM2, and in some cases CVM and BVM, gave lower maximum classification accuracies. Apart from the excellent experimental results for AESVM with the Gaussian kernel, AESVM also gave good results with the polynomial kernel as described in Section 5.4.

The algorithm DeriveRS was generally found to be efficient, especially for the lower dimensional data sets D1-D5. For the high dimensional data sets D6-D9, the representative set was almost the same size as the training data set, resulting in small gains in training and classification time speedups for AESVM. In particular, for D7 and D8 the representative set computed by DeriveRS was almost 100% of the training set. A similar result was reported for this data set in Beygelzimer et al. (2006), where a divide and conquer method was used to speed up nearest neighbor search. Data set D8 is reported to have resulted in nearly no speedup, compared to a speedup of almost one thousand for other data sets when their method was used. Their analysis found that the data vectors in D8 were very distant from each other in comparison with the other data sets.[13] This observation can explain the performance of DeriveRS on D8, as data vectors that are very distant from each other

13. This is indicated by the large expansion constant for D8 illustrated in Beygelzimer et al. (2006).

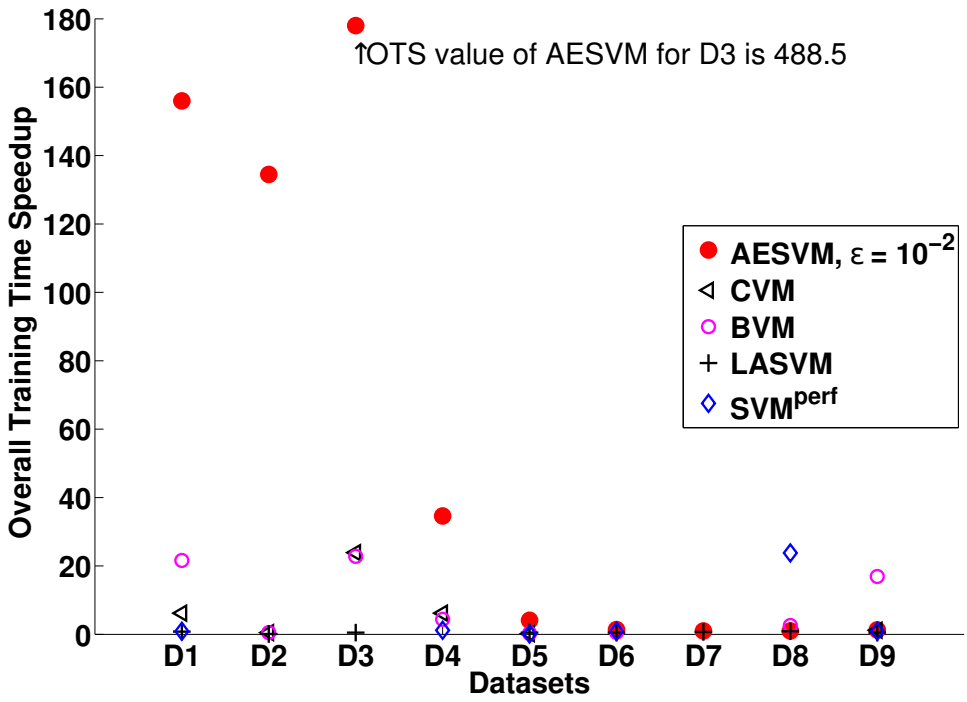


Figure 6: Plot of overall training time speedup (compared to LIBSVM) of all SVM solvers

are expected to have large representative sets. It should be noted that irrespective of the dimensionality of the data sets, AESVM always resulted in excellent performance in terms of classification accuracy. There seems to be no relation between data set density and the performance of DeriveRS and AESVM.

The authors will provide the software implementation of AESVM and DeriveRS upon request. Based on the presented results, we suggest the parameters $\epsilon = 10^{-2}$, $P = 10^5$ and $V = 10^3$ for DeriveRS. A possible extension of this paper is to apply the idea of the representative set to other SVM variants and support vector clustering. It would be interesting to investigate AESVM solvers implemented using methods other than SMO. Modifications to DeriveRS using the methods in Section 2 might improve its performance on high dimensional data sets. The authors will investigate improvements to DeriveRS and the application of AESVM to the linear kernel in their future work.

Acknowledgments

Dr. Khargonekar acknowledges support from the Eckis professor endowment at the University of Florida. Dr. Talathi was partially supported by the Children's Miracle Network, and the Wilder Center of Excellence in Epilepsy Research. The authors acknowledge Mr. Shivakeshavan R. Giridharan, for providing assistance with computational resources.

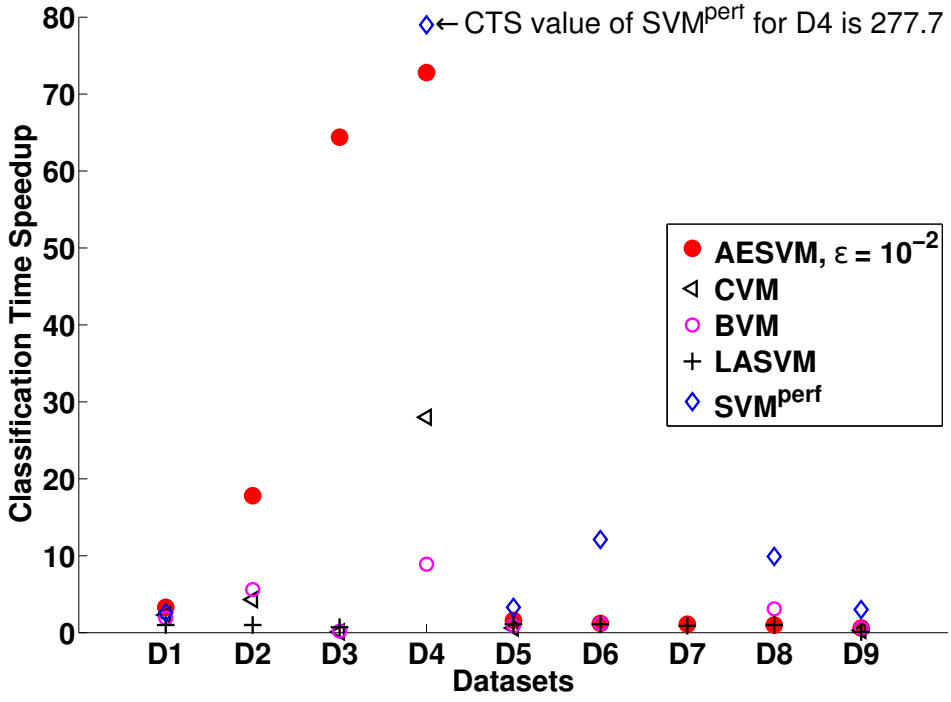


Figure 7: Plot of classification time speedup for optimal hyper-parameters (compared to LIBSVM) of all SVM solvers

References

K. P. Bennett and E. J. Bredensteiner. Duality and geometry in SVM classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 57–64, 2000.

A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 97–104, 2006.

M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7:448–461, August 1973.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, December 2005.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

J. Cervantes, X. Li, W. Yu, and K. Li. Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing*, 71:611–619, January 2008.

C. C. Chang and C. J. Lin. IJCNN 2001 challenge: Generalization ability and text decoding. In *Proceedings of International Joint Conference on Neural Networks*, volume 2, pages 1031 –1036, 2001.

C.C Chang and C.J Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:1–27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

K. L. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Transaction on Algorithms*, 6(4):63:1–63:30, September 2010.

R. Collobert, S. Bengio, and Y. Bengio. A parallel mixture of SVMs for very large scale problems. *Neural Computing*, 14(5):1105–1114, 2002.

P. Drineas and M. W. Mahoney. On the Nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, December 2005.

R. E. Fan, P. H. Chen, and C. J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, June 2008.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2002.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th International Conference on Machine Learning*, pages 320–327, 2008.

B. Gärtner and M. Jaggi. Coresets for polytope distance. In *Proceedings of the 25th Annual Symposium on Computational Geometry*, pages 33–42, 2009.

J. Guo, N. Takahashi, and T. Nishi. A learning algorithm for improving the classification speed of support vector machines. In *Proceedings of the 2005 European Conference on Circuit Theory and Design*, volume 3, pages 381 – 384, 2005.

C. J. Hsieh, K. W. Chang, C. J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th International Conference on Machine Learning*, pages 408–415, 2008.

T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods*, pages 169–184. MIT Press, 1999.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226. ACM, 2006.

T. Joachims and C. N. J. Yu. Sparse kernel SVMs via cutting-plane training. *Machine Learning*, 76:179–193, September 2009.

B. Kaluža, V. Mirchevska, E. Dovgan, M. Luštrek, and M. Gams. An agent-based approach to care in independent living. In *Ambient Intelligence*, pages 177–186. Springer, 2010.

J. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278 –2324, 1998.

Y. J. Lee and O. L. Mangasarian. Rsvm: Reduced support vector machines. In *Proceedings of the First SIAM International Conference on Data Mining*, pages 5–7. SIAM Philadelphia, 2001.

M. Nandan, S. S. Talathi, S. Myers, W. L. Ditto, P. P. Khargonekar, and P. R. Carney. Support vector machines for seizure detection in an animal model of chronic epilepsy. *Journal of Neural Engineering*, 7(3), 2010.

E. Osuna and O. Castro. Convex hull in feature space for support vector machines. In *Proceedings of the 8th Ibero-American Conference on AI: Advances in Artificial Intelligence*, pages 411–419, 2002.

E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 130 –136, 1997.

D. Pavlov, D. Chudova, and P. Smyth. Towards scalable support vector machines using squashing. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 295–299. ACM, 2000.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods*, pages 185–208. MIT Press, 1999.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, pages 1177–1184, 2007.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1996.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, 2000.

S. Shalev-Shwartz and N. Srebro. SVM optimization: Inverse dependence on training set size. In *Proceedings of the 25th International Conference on Machine Learning*, pages 928–935, 2008.

S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated subgradient solver for SVM. *Mathematical Programming*, 127:3–30, March 2011.

S. S. Talathi, D. U. Hwang, M. L. Spano, J. Simonotto, M. D. Furman, S. M. Myers, J. T. Winters, W. L. Ditto, and P. R. Carney. Non-parametric early seizure detection in an animal model of temporal lobe epilepsy. *Journal of Neural Engineering*, 5:85–98, 2008.

M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the 2009 IEEE Symposium Computational Intelligence for Security and Defense Applications*, pages 53–58, 2009.

D. Tax and R. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

C. H. Teo, S. V. N. Vishwanthan, A. J. Smola, and Q. V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, 2010.

I. W. Tsang, J. T. Kwok, P. Cheung, and N. Cristianini. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

I. W. Tsang, A. Kocsor, and J. T. Kwok. Simpler core vector machines with enclosing balls. In *Proceedings of the 24th International Conference on Machine Learning*, pages 911–918, 2007.

H. Yu, J. Yang, and J. Han. Classifying large data sets using SVMs with hierarchical clusters. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 306–315, 2003.

G. X. Yuan, C. H. Ho, and C. J. Lin. An improved GLMNET for l1-regularized logistic regression. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 33–41, 2011.

T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the 21st International Conference on Machine Learning*, pages 919–926, 2004.

Journal of Machine Learning Research 15 (2014) 99-140 Submitted 3/13; Revised 11/13; Published 1/14

Detecting Click Fraud in Online Advertising: A Data Mining Approach

Richard Oentaryo ROENTARYO@SMU.EDU.SG
Ee-Peng Lim EPLIM@SMU.EDU.SG
Michael Finegold MFINEGOL@SMU.EDU.SG
David Lo DAVIDLO@SMU.EDU.SG
Feida Zhu FDZHU@SMU.EDU.SG
Living Analytics Research Centre
Singapore Management University
80 Stamford Road, Singapore

Clifton Phua CLIFTON.PHUA@SAS.COM
SAS Institute Pte. Ltd.
20 Anson Road, Singapore

Eng-Yeow Cheu EYCHEU@I2R.A-STAR.EDU.SG
Ghim-Eng Yap GEYAP@I2R.A-STAR.EDU.SG
Kelvin Sim SHSIM@I2R.A-STAR.EDU.SG
Minh Nhut Nguyen MNNGUYEN@I2R.A-STAR.EDU.SG
Data Analytics Department
Institute for Infocomm Research
1 Fusionopolis Way, Singapore

Kasun Perera BPERERA@MASDAR.AC.AE
Bijay Neupane BNEUPANE@MASDAR.AC.AE
Mustafa Faisal MFAISAL@MASDAR.AC.AE
Zeyar Aung ZAUNG@MASDAR.AC.AE
Wei Lee Woon WWOON@MASDAR.AC.AE
Masdar Institute of Science and Technology
Abu Dhabi, United Arab Emirates

Wei Chen CHENWEI@I2R.A-STAR.EDU.SG
Urban Systems Programme Office
Institute for Infocomm Research
1 Fusionopolis Way, Singapore

Dhaval Patel PATELFEC@IITR.ERNET.IN
Department of Computer Science and Engineering
Indian Institute of Technology Roorkee
Century Road, Roorkee, Uttarakhand, India

Daniel Berrar BERRAR.D.AA@M.TITECH.AC.JP
Interdisciplinary Graduate School of Science and Engineering
Tokyo Institute of Technology
4259 Nagatsuta, Midori-ku, Yokohama, Japan

Abstract

Click fraud–the deliberate clicking on advertisements with no real interest on the product or service offered–is one of the most daunting problems in online advertising. Building an effective fraud detection method is thus pivotal for online advertising businesses. We

organized a *Fraud Detection in Mobile Advertising* (FDMA) 2012 Competition, opening the opportunity for participants to work on real-world fraud data from BuzzCity Pte. Ltd., a global mobile advertising company based in Singapore. In particular, the task is to identify fraudulent publishers who generate illegitimate clicks, and distinguish them from normal publishers. The competition was held from September 1 to September 30, 2012, attracting 127 teams from more than 15 countries. The mobile advertising data are unique and complex, involving heterogeneous information, noisy patterns with missing values, and highly imbalanced class distribution. The competition results provide a comprehensive study on the usability of data mining-based fraud detection approaches in practical setting. Our principal findings are that features derived from fine-grained time-series analysis are crucial for accurate fraud detection, and that ensemble methods offer promising solutions to highly-imbalanced nonlinear classification tasks with mixed variable types and noisy/missing patterns. The competition data remain available for further studies at `http://palanteer.sis.smu.edu.sg/fdma2012/`.

Keywords: ensemble learning, feature engineering, fraud detection, imbalanced classification

1. Introduction

Advances in data management and web technologies have rendered online advertising as the ideal choice for small and large businesses to effectively target the appropriate marketing segments on the fly. The main coordinator in this setting is the advertising *commissioner* (also known as *ad network*), acting as a broker between advertisers and content publishers. An *advertiser* plans a budget, provides the commissioner with advertisements, and agrees on a commission for every customer action (e.g., clicking an ad, filling a form, bidding in an auction, etc). A *content publisher* contracts with the commissioner to display advertisements on their websites, and gets commissions based on the traffic it drives to the advertisers. This model, however, may incentivise dishonest publishers to generate illegitimate clicks on their sites–a major issue known as *click fraud*. Click fraud degrades the reliability of online advertising systems and, if not kept under control, can lead to a contraction of the advertising market in the long term. There have also been high-profile, costly litigations from unsatisfied advertisers, giving bad reputation for the commissioners. Thus, a reliable click fraud detection system is needed to help the commissioners proactively prevent click fraud and assure their advertisers that their dollars have been well spent.

To this end, we organized a *Fraud Detection in Mobile Advertising* (FDMA) 2012 Competition, centered around real-world mobile advertising data. The goal is to develop and crowdsource data mining and machine learning methods capable of building effective predictive models to detect fraudulent publishers. The competition offers a unique opportunity to work on click and publisher data sets provided by BuzzCity Pte. Ltd., a global mobile advertising network that has millions of consumers around the world (particularly in India, Indonesia and Africa) accessing internet contents and interacting on mobile phones and devices. Most publishers in the BuzzCity network adopt the *cost per click* (CPC) payment scheme, which is subject to abuses by malicious publishers through click fraud. In Q1 2012, over 45 billion ad banners were delivered across the BuzzCity network, having over 10,000 publisher sites and reaching an average of 300 million unique users per month. A fast and robust detection of the most predictive variables for fraudulent behavior is thus of great importance. Currently, BuzzCity uses an in-house developed detection mechanism

to identify fraudulent publishers semi-automatically. The use of data mining and machine learning methods will provide more detailed insights for improving the detection accuracy, while reducing the efforts for manual interventions.

Accordingly, the FDMA 2012 Competition aims at providing an empirical platform to gauge the state-of-the-art data mining and machine learning methods in a setting typical of industrial applications. We summarize the key contributions of this paper below:

- We present an important application of machine learning and data mining methods to tackle real-world fraud detection problems, which serves as valuable resources for industrial and research practitioners. Thus far, there is a lack of comprehensive study on data mining/machine learning approaches for fraud detection in advertising.
- Our study involves proprietary, industrial data, which are rarely available and pose a challenging problem for many data mining and machine learning algorithms. The solutions presented in this paper address some important issues in data mining and machine learning research, including highly imbalanced distribution of the output variable, heterogeneous data (mixture of numerical and categorical variables), and noisy patterns with missing/unknown values.
- We show that exploratory data analysis and feature engineering are crucial milestones for effective fraud detection. In particular, we present systematic analysis of both *spatial* and *temporal factors* at different levels of granularity, which leads to creation of good, predictive features for accurate fraud detection.
- We investigate the applicability of a wide range of *single* and *ensemble learning* algorithms in fraud detection task. We found that the ensemble algorithms produce significant improvement over the single algorithms. Also, coupling ensemble learning with feature ranking analysis leads to discovery of the most important features for distinguishing between fraudulent and normal behaviors.
- To the best of our knowledge, FDMA 2012 is also the first open international competition and crowdsourcing initiative on fraud detection in online advertising. The results not only provide useful research insights, but also illustrate how companies (such as BuzzCity) can use data mining and machine learning methods to obtain useful, actionable knowledge for improving their business operations.

In this paper, we report the selected, winning entries of the FDMA 2012 Competition, which provide important insights on click fraud behavior. In Section 2, we give an overview of the competition data, challenges, and evaluation procedures. Table 1 summarizes the profiles of the winning teams and contributors of this paper, and Sections 3 to 6 elaborate in turn their "journeys" and key findings. We next describe the work independently done by the competition organizer in Section 7. Finally, Section 8 provides concluding remarks.

2. Competition

In this section, we first describe the data set, the task objective, as well as the evaluation criterion adopted in the competition. We also briefly describe the online website and public leaderboard systems we built to support the competition.

Rank	Team name	Members	Contribution
1	`starrystarrynight`	• Clifton Phua • Eng-Yeow Cheu • Ghim-Eng Yap • Kelvin Sim • Minh-Nhut Nguyen	Section 3
2	`TeamMasdar`	• Kasun Perera • Bijay Neupane • Mustafa A. Faisal • Zeyar Aung • Wei Lee Woon	Section 4
3	`DB2`	• Wei Chen • Dhaval Patel	Section 5
4	`Tea`	• Daniel Berrar	Section 6

Table 1: Winning teams in the FDMA 2012 Competition.

2.1 Data

The raw data supplied by BuzzCity consist of two categories: *publisher database* and *click database*, both provided in comma-separated values (CSV) format. The publisher database records the publisher/partners profile, and consists of several fields as listed in Table 2. On the other hand, the click database captures the click traffic associated with various publishers. Table 3 lists the fields in the click database. Table 4 provides a sample of the two largest publishers of each status in the training set, a `Fraud` and an `OK` publisher, and Table 5 lists three click samples from each publisher. There is another `Observation` status, comprising small number of new publishers, or publishers who have high click traffic and not yet deemed as fraudulent. Note that some fields in the publisher and click databases have been anonymized for privacy protection.

Field	Description
`publisherid`	Unique identifier of a publisher
`bankaccount`	Bank account associated with a publisher (anonymized; may be missing/unknown)
`address`	Mailing address of a publisher (anonymized; may be missing/unknown)
`status`	Label of a publisher, which falls into three categories: • `OK`: Publishers whom BuzzCity deems as having healthy traffic (or those who slipped their detection mechanisms) • `Observation`: Publishers who may have just started their traffic or their traffic statistics deviates from system wide average. BuzzCity does not have any conclusive stand with these publishers yet • `Fraud`: Publishers who are deemed as fraudulent with clear proof. BuzzCity suspends their accounts and their earnings will not be paid

Table 2: Fields in the publisher database.

2.2 Challenge

The FDMA 2012 competition aims at building a data-driven methodology for effective detection of fraudulent publishers. In particular, each participant is tasked to highlight potential `Fraud` publishers and distinguish them from `OK` and `Observation` (or collectively

Field	Description
`id`	Unique identifier of a particular click
`numericip`	Public IP address of a clicker/visitor
`deviceua`	Phone model/agent used by a clicker/visitor
`publisherid`	Unique identifier of a publisher
`campaignid`	Unique identifier of a given advertisement campaign
`usercountry`	Country from which the clicker/visitor is
`clicktime`	Timestamp of a given click (in yyyy-mm-dd format)
`referredurl`	URL where ad banners are clicked (anonymized; may be missing/unknown)
`channel`	Publisher's channel type, which consists of: • `ad`: Adult sites • `co`: Community • `es`: Entertainment and lifestyle • `gd`: Glamour and dating • `in`: Information • `mc`: Mobile content • `pp`: Premium portal • `se`: Search, portal, services

Table 3: Fields in the click database.

`publisherid`	`bankaccount`	`address`	`status`
8iaxj		14vxbyt6sao00s84	Fraud
8jljr			OK

Table 4: Publisher sample in raw training data. There are missing values in `bankaccount` and `address`. In pay per click online advertising, `Fraud` involves a large number of intentional click charges with no real interest in the advertisements, using automated scripts or click farms. The perpetrators can be the publishers themselves or their competitors, or the competitors of advertisers.

Normal) publishers, based on their click traffic and account profiles. This will help shed light on several key areas, such as identifying the common underlying fraud schemes or concealment strategies, understanding patterns of dishonest publishers, and developing new ways for effective prevention/detection plans.

`id`	`numericip`	`deviceua`	`publisherid`	`campaignid`	`usercountry`	`clicktime`	`channel`	`referredurl`
13417867	3648406743	GT-I9100	8iaxj	8fj2j	ru	2012-02-09 00:00:00	ad	26okyx5i82hws84o
13417870	3756963656	Samsung_S5233	8jljr	8geyk	in	2012-02-09 00:00:00	es	15vynjr7rm00gw0g
13417872	693232332	SonyEricsson_K70	8jljr	8gkkx	ke	2012-02-09 00:00:00	es	
13417893	2884200452	Nokia_6300	8jljr	8gp95	vn	2012-02-09 00:00:01	es	
13418096	3648406743	GT-I9100	8iaxj	8fj2m	ru	2012-02-09 00:00:08	ad	24w9x4d25ts00400
13418395	781347853	GT-I9003	8iaxj	8fj2j	ru	2012-02-09 00:00:20	ad	4im401arl30gc0gk

Table 5: Click samples in raw training data. There are missing values in `referredurl` and `deviceua`. The raw features include IP address of a clicker (`numericip`), mobile device model used by the visitor (`deviceua`), campaign ID of a particular advertisement campaign (`campaignid`), country of the visitor (`usercountry`), publisher channel type (`channel`), or an URL where the ad banner is clicked (`referredurl`).

More specifically, we seek to answer this question: Given historical patterns (of both fraudulent and normal publishers) in some time period (e.g., a 3 day period), how to detect fraudulent publishers in a future period (e.g., a 3 day period in the week after)? That is, we are interested in a detection (predictive) model that can generalize well over time. To this end, BuzzCity provides three sets of publishers and clicks data taken from different time periods: a *training set* (for building predictive model), a *validation set* (for model selection), and a *test set* (for evaluating the models' generalization abilities and determining the competition winners). Each click data set captures the click traffic over a 3 day period, while each publisher data set records publishers receiving at least one click in that period. We summarize the count statistics of the publishers and clicks in Table 6.

It is worth noting that the publisher labels (i.e., `Fraud`, `Observation`, `OK`) were generated from BuzzCity's semi-automatic detection mechanism (cf. Section 1) that uses two types of auxiliary information together: *offline* and *online*; the former corresponds to information that BuzzCity deems impossible to automate or does not attempt to computerize (e.g., manually contact the publishers and verify their responses), while the latter is obtained based on the statistical analysis of the click behavior done by BuzzCity's proprietary automated programs. Due to the proprietary nature of this practice and for simplicity, the details of the label generation process were not given as part of the competition. Here BuzzCity's primary interest is whether the competition participants can independently infer and discover fraudulent patterns based on the click and publisher databases alone, without using the auxiliary information. Also note that the `Fraud` and `Observation` publishers constitute very small portions of the population relative to the `OK` publishers (cf. Table 6), rendering this problem challenging for many contemporary classification methods.

			No. of publishers			
Data set	Time period	No. of clicks	Fraud	Observation	OK	Total
Train	9-11 Feb 2012	3,173,834	72 (2.34%)	80 (2.60%)	2,929 (95.07%)	3,081
Validation	23-25 Feb 2012	2,689,005	85 (2.77%)	84 (2.74%)	2,895 (94.48%)	3,064
Test	8-10 Mar 2012	2,598,815	82 (2.73%)	71 (2.37%)	2,847 (94.90%)	3,000

Table 6: Statistics of the competition data.

2.3 Evaluation

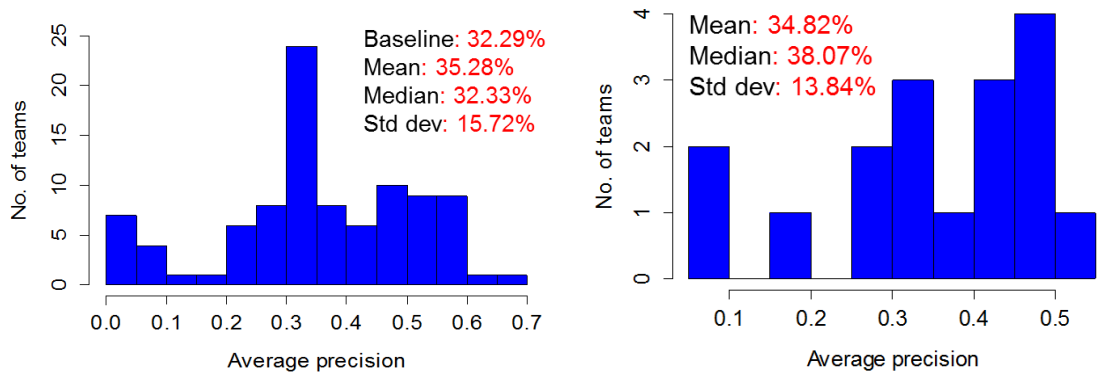
For performance evaluation, we chose to adopt the *average precision* criterion, which favors algorithms capable of ranking the few useful items ahead of the rest. Such a criterion is particularly suitable for detecting rare instances such as fraud cases (Zhu, 2004). We describe the criterion as follows: Let π be the number of relevant (i.e., actual `Fraud`) instances. We first ranked the instances according to the prediction/detection scores produced by each algorithm. Among the $t \times 100\%$ top-ranked instances, supposing $h(t) \leq t$ are truly relevant (also called *hits*), let $r(t) = \frac{h(t)}{\pi}$ and $p(t) = \frac{h(t)}{t}$ be the *recall* and *precision* respectively. Typically, $h(t)$ and $p(t)$ takes values only at a finite number of points $t_i = \frac{i}{n}, i = 1, 2, ..., n$. Using these variables, the average precision (AP) criterion can be computed as

$$AP = \sum_{i=1}^{n} p(t_i) \left(r(t_i) - r(t_{i-1}) \right).$$

Essentially, the AP criterion summarizes the precision-recall performances at different threshold levels, and corresponds to the area under the precision-recall curve (Zhu, 2004). In the case of fraud detection, simply evaluating precision and recall at a specific threshold level is inadequate, since these metrics vary with the strictness of a classification algorithm's threshold and the range of its prediction outputs. Further details on the AP criterion can be found in Zhu (2004).

2.4 Website and Leaderboard

Our FDMA 2012 website supports a public leaderboard system displaying the best AP score and the submission time of each team on the validation set. We ran the competition for 1 month, from 1 to 30 September 2012, with two submissions per day allowed for each team. BuzzCity offered a total prize of 7,000 Singapore dollars (SGD) for the competition winners (i.e., SGD 4,000, 2,000, and 1,000 for the first, second, and third winners respectively). During the competition, the actual publishers' status labels (i.e., ground-truth) in the validation set were hidden, and our system computes the average precision for each submission. The submitting teams received email notifications showing their current scores and submission time. This allows teams to track their progress and fine-tune their models. To ensure the models developed do not overfit the validation set, we used the test set for the final evaluations, the status labels of which were also hidden during the competition. The test set was only revealed 72 hours before the competition ended.



(a) (b)

Figure 1: Statistics of the average precision scores: (a) Public leaderboard (validation set), and (b) Private leaderboard (test set).

Figure 1(a) shows the overall statistics of the leaderboard scores as of September 30, 2012, while Figure 1(b) shows the statistics of the final test scores. In total, we had 127 teams registering for the competition, 88 of which explicitly specified their affiliation and country: 60 from academic institutions and 28 from industry, and 51 were local teams from Singapore and 37 were overseas teams. A total of 95 teams submitted to the leaderboard during the competition period. For baseline in the leaderboard, we used the logistic re-

Rank	Team	Average precision		Affiliation
		Validation set	Test set	
1	`starrystarrynight`	59.38%	51.55%	Institute of Infocomm Research
2	`TeamMasdar`	59.39%	46.42%	Masdar Institute of Science & Technology
3	`DB2`	62.21%	46.15%	National University of Singapore
4	`Tea`	51.55%	42.01%	Tokyo Institute of Technology
(*)	`LARC`	57.79%	55.64%	Singapore Management University

Table 7: Results of the top teams on the validation and test sets. Team ranks were determined using the test results.

gression method (Fan et al., 2008), which provides reasonable performance reference close to the mean or median AP score. The final standings on the validation and test sets are summarized in Table 7. We also showed the best result obtained by the competition organizer (dubbed as team `LARC`) in the last row. Comparing the validation and test results, the position of the top-3 ranks were reversed, which may be attributed to overfitting.

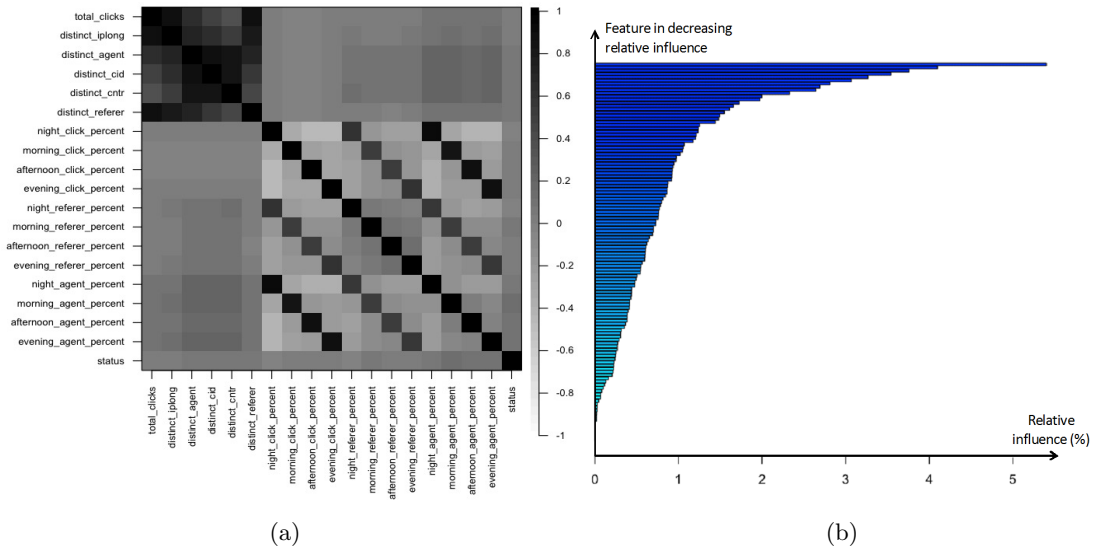
3. First Winner's Entry

This section describes the entry from the first winning team, which covers the data preprocessing or feature extraction techniques and classification method used, followed by the empirical results and insights obtained by the team.

3.1 Preprocessing and Feature Extraction

Figure 2(a) plots the correlations among some of our features including `status`, which we used to ensure feature diversity by excluding new features which are too similar to existing ones. In Figure 2(b), using specific model parameters described in the next section, we obtained the relative influence or importance of 118 predictive features in the final training set. A complete listing of the 118 features is available at `http://clifton.phua.googlepages.com/feature-list.txt`. In addition, there are two other features: `publisherid`, which is not used for model building, and `status`, which is the class or dependent feature. Adding all the features' relative influence will sum up to a score of one hundred. On one extreme, there are a few features with relative influence above three. On another extreme, there are a few features with negligible influence on the results such as `channel`-related features (see Section 3.3.3 on a discussion of leveraging the predictiveness of the raw `channel` feature). The average relative influence per feature is about 0.88%.

The 118 predictive features can be grouped into three types of features: 67 click behavior (57%), 40 repetitive click behavior (34%), and 11 high-risk click behavior (9%). The average rank of all the features (based on the model output, as described in Section 3.2) is 69, 35, and 69 respectively, meaning that duplicated clicks are likely to be invalid clicks. We use simple statistical features based on average, standard deviation, and percentages, and none of our features are created directly from `status` or specific values from raw anonymized features, such as `bankaccount`, `address`, `numericip`, `campaignid`, and `referredurl`.



(a) (b)

Figure 2: (a) Correlation plot of some click behavior features in the training set. (b) Relative influence of all features in the training set.

3.2 Method

Gradient boosting is a machine learning technique used for classification problems with a suitable loss function, which produces a final prediction model in the form of an ensemble of weak prediction decision trees (Friedman, 2000). We used the implementation of generalized boosted regression model (GBM) in R's `gbm` package (Ridgeway, 2007). The final parameters used on the final training data set for our best average precision on the test data set are:

- `distribution` (loss function): "bernoulli" - also tested "Adaboost" distribution
- `n.trees` (number of iterations): 5000 - tested 100 to 5000 decision trees
- `shrinkage` (learning rate): 0.001 - tested 0.001 to 0.01
- `interaction.depth` (tree depth): 5 - tested 2 to 5
- `n.minobsinnode` (minimum observations in terminal node): 5 - tested 2 to 5

In the early to mid stages of the competition, we used two layers of GBM to select the most important features. During the final stages, we focused on only one layer of GBM as we had identified the three best types of features. Initially, we also tried random forest (Breiman, 2001) (decision trees ensemble algorithm) in R's `randomForest` package as well as RIPPER (Cohen, 1995) (rule induction algorithm) in WEKA (Hall et al., 2009). As the random forest and RIPPER did not perform as well as GBM on the validation set, we did not conduct further explorations on them or other classification algorithms. If we did find alternative algorithms that perform on par with (or better than) GBM, we could train a set of base classifiers and combine them with stacking (Wolpert, 1992).

3.3 Result and Discussion

This section describes several key empirical results and insights gained by the first winner.

3.3.1 Spatial and Temporal Patterns

In Table 8, we list the top-10 features of each type to show that our features capture some temporal and spatial aspects of clicks for each publisher. Within the one minute interval, fraudulent clicks have significantly more duplicates than normal ones. For repetitive click behavior features, the shorter intervals produce better results after we tested one, five, fifteen, thirty, and sixty minutes intervals using Chao-Shen entropy (Chao and Shen, 2003). Chao-Shen entropy is a non-parametric estimation of Shannon's index of diversity. It combines the Horvitz-Thompson estimator (Horvitz and Thompson, 1952) and the concept of sample coverage proposed by Good (1953) to adjust for unseen observations in a sample. For example, there are multi-feature duplicates such as `avg_spiky_ReAgCnIpCi` (average number of the same `referredurl`, `deviceua`, `usercountry`, `numericip`, and `campaignid` being duplicated in one minute), as well as single feature duplicates such as `std_spiky_numericip` (standard deviation of `numericip` being duplicated in one minute).

For our top click behavior and duplication features, we created conditional features based on finer-grained time intervals to better capture temporal dynamics of click fraud behavior. We divided a day into four six-hour periods: night (12am to 5:59am), morning (6am to 11:59am), afternoon (12pm to 5:59pm), and evening (6pm to 11:59pm). For example, `night_referredurl_percent` is the number of distinct referredurls at night divided by the total number of distinct referredurls, and `night_avg_spiky_referredurl` is the average number of the same referredurl being duplicated within one minute at night. Also, we divided an hour into four fifteen-minute periods: first (0-14), second (15-29), third (30-44), and last (45-59). For example, `second_15_minute_percent` is the number of clicks between 15th to 29th minute divided by total number of clicks.

Fraudulent clicks tend to come from some countries (or finer-grained spatial regions) more than others, for example, businesses in India and Indonesia are hardest hit by fraud (Kroll Advisory Solutions, 2012). Most clicks on mobile advertisements also come from these two countries. We tested the top five, ten, fifteen, twenty, and twenty-five high-risk countries (out of two hundred over countries), and found that the top ten high-risk countries works best. For example, `usercountry_in_percent` and `usercountry_id_percent` are the percentages of invalid clicks originating from India and Indonesia respectively. The large numbers of invalid clicks coming from `usercountry_sg_percent` or Singapore could be due to BuzzCity's penetration tests being conducted from there.

3.3.2 Performance

Using the GBM configuration with the 118 features mentioned in Section 3.1, our team was ranked fourth with an average precision of **59.38%** on the validation set, as displayed on the public leaderboard. After the competition ended and teams submitted their results based on the test data, we were ranked the first with the average precision of **51.55%**. (The second winner finished the line with an average precision of 46.42%, and the third winner with 46.15%.) As such, comparing our result with that of the other top teams, we can conclude that our GBM model fits the data well.

Rank	Feature	Relative influence
6	std_per_hour_density	3.12
12	total_clicks	1.76
14	brand_Generic_percent	1.71
15	avg_distinct_referredurl	1.62
19	std_total_clicks	1.43
23	night_referredurl_percent	1.19
24	second_ 15_minute_percent	1.18
27	distinct_referredurl	1.15
29	std_distinct_referredurl	1.12
30	morning_click_percent	1.1

(a)

Rank	Feature	Relative influence
2	std_spiky_numericip	3.81
3	avg_spiky_ReAgCnIpCi	3.69
4	night_avg_spiky_referredurl	3.66
5	avg_spiky_deviceua	3.29
8	avg_spiky_referredurl	2.67
9	avg_spiky_ReAgCn	2.59
10	night_avg_spiky_ReAgCnIpCi	2.49
11	afternoon_avg_spiky_ReAgCnIpCi	1.98
13	afternoon_avg_spiky_deviceua	1.75
16	std_spiky_referredurl	1.6

(b)

Rank	Feature	Relative influence
1	usercountry_id_percent	5.49
7	usercountry_sg_percent	2.69
49	usercountry_other_percent	0.72
72	usercountry_us_percent	0.44
77	usercountry_th_percent	0.39
84	usercountry_uk_percent	0.34
91	usercountry_in_percent	0.26
103	usercountry_ng_percent	0.05
104	usercountry_tr_percent	0.04
106	usercountry_ru_percent	0.04

(c)

Table 8: Top-10 features by type in final training data set: (a) By click behavior, (b) By repetitive click behavior, and (c) By high-risk click behavior.

3.3.3 Other Potentials

In Table 9, we show that there is some potential for an alternative approach using the `channel` of each fraudulent publisher. Fraudulent *mobile* and *adult* content publishers tend to produce much more invalid clicks than the other fraudulent publishers, especially at night and morning periods. In contrast, fraudulent *entertainment*, *lifestyle*, and *premium portal* publishers produce a lot less invalid clicks, and tend to have relatively more invalid clicks during afternoon and evening periods. We attempted to split the data sets and build models

separately by channel, but did not have enough time to integrate/normalize the different sets of prediction scores in a meaningful way.

Channel	Publisher count	Fraud clicks (fraud %)	Night Fraud clicks (fraud %)	Morning Fraud clicks (fraud %)	Afternoon Fraud clicks (fraud %)	Evening Fraud clicks (fraud %)
Adult (ad)	10	**47226 (37%)**	**15435 (12%)**	6439 (5%)	**11299 (9%)**	**14053 (11%)**
Mobile content (mc)	**23**	41941 (33%)	13589 (11%)	**9284 (7%)**	9623 (8%)	9445 (7%)
Community (co)	12	16411 (13%)	7218 (6%)	3301 (3%)	2612 (2%)	3280 (3%)
Entertainment and lifestyle (es)	14	14433 (11%)	2649 (2%)	3265 (3%)	3573 (3%)	4946 (4%)
Search, portal, services (se)	4	3180 (3%)	682 (1%)	572 (0%)	689 (1%)	1568 (1%)
Premium portal (pp)	6	2926 (2%)	351 (0%)	608 (0%)	732 (1%)	904 (1%)
Information (in)	3	893 (1%)	49 (0%)	284 (0%)	428 (0%)	132 (0%)
Total	72	127010 (100%)	**39973 (31%)**	23753 (19%)	28956 (23%)	34328 (27%)

Table 9: High risk categories in final training data set.

3.3.4 Recommendations

We conclude this section by addressing several key questions in relation to a broader context:

- **What is the underlying click fraud scheme?** Simply put, a relatively large number of clicks or rapid duplicate clicks, or a high percentage of clicks from high-risk countries have been shown to be important fraud indicators.
- **What sort of concealment strategies commonly used by fraudulent parties?** The Tuzhilin Report (Tuzhilin, 2006) on the Google AdWords/AdSense system lists ten possible strategies or sources of invalid clicks. The hard-to-detect click fraud tends to come from organized crime in hard-to-prosecute countries (Chambers, 2012). For example, hard-to-detect and hard-to-prosecute click fraud uses existing user traffic including 0-size iframes, forced searching, and zombie computers.
- **How to interpret data for patterns of dishonest publishers and websites?** From a machine learning point-of-view, some decision tree and rule induction algorithms can provide high interpretability to fraud patterns. However, the key step prior to this is still to engineer the best features using domain knowledge and experimentation, and to allow investigators to discern and validate these fraud patterns from top ranked features, even through black-box classification algorithms.
- **How to build effective fraud prevention/detection plans?** Effective fraud detection plans need to have elements of resilience, adaptivity, and quality data (Phua et al., 2012). Resilience is "defense-in-depth" with multiple, sequential, and independent layers of defense. For example, BuzzCity already has anomaly-based detectors to place some publishers under observation, and they can consider adding classifier-based detectors to their click fraud detection system. In the context of click fraud, the classifier-based detectors need to be adaptive to changing fraud and normal click behavior. The classifier-based detectors also need to use quality data with timely updates when publishers are discovered to be fraudulent. Other than increasing advertisers' awareness of click/conversion ratios, having better customer service and fraud policies, and improving automated filters, one can pursue click fraud more aggressively, switch from cost-per-click to cost-per-action advertising model, or cultivate trust with advertisers by having independent audits (Jansen, 2007).

4. Second Winner's Entry

In this section, we describe the second winner's entry. Following the organization of Section 3, we elaborate the preprocessing/feature extraction technique and classification method used by the team, as well as their empirical results and insights.

4.1 Preprocessing and Feature Extraction

We first analyzed each attribute in the raw click and publisher database, and evaluated its effects on the behavior of a publisher. We observed that not all features are useful; attributes from the publisher database such as `address`, `bankaccount` can be excluded from the feature construction process. To facilitate different experiment settings, we considered three data sets. In the first data set, all publishers labeled as `Observation` were relabeled as `OK`. In the second data set, all publishers labeled as `Observation` were relabeled as `Fraud`. Finally, the third data set retains all the three (original) labels. Training and testing were performed accordingly on all the three data sets.

Feature extraction is another important facet for building the prediction model. Properly selected features should be able to capture properties or trends that are specific to fraudulent publishers and are robust against their evolving behavior. We took each raw attribute in the click database and model the publisher's click pattern by creating several statistical features based on that particular attribute. The feature extraction procedures applied to different attributes are detailed hereafter.

4.1.1 Attribute: `clicktime`

Fraudulent publishers often disguise their activities using various tricks such as generating very sparse click sequences, changes in IP addresses, issuing clicks from different computers in different countries and so on. Others stick to the conservative approach of generating the maximum number of clicks in a given interval. It is important for any fraud detection system to recognize both kinds of concealment strategies. Accordingly, we derived several statistical features from the `clicktime` attribute in the click database, with the number of clicks for each publisher observed over different time intervals: 1 minute, 5 minutes, 1 hours, 3 hours and 6 hours. The goal is to capture both the short and long term behavior of the publishers, based on the observation that publishers often try to act rationally and have constant clicks in very sparse time intervals.

Specifically, for each time interval, we counted the number of clicks each publisher receives and aggregated these counts using several features: maximum clicks, average click, click skewness, and click variance. Click variance measures the deviation of number of clicks from the average clicks (norm) of a publisher, while click skewness is a measure of the asymmetry of the click distribution. Figure 3 shows all the features we derived from the attribute `clicktime` from the raw click database.

4.1.2 Attribute: `numericip`

Internet protocol (IP) address is another attribute that can be used to characterize the behavior of a publisher, since it is a reflection of the number of computers/mobile devices used or different times at which the user clicks on a particular advertisement. Since many

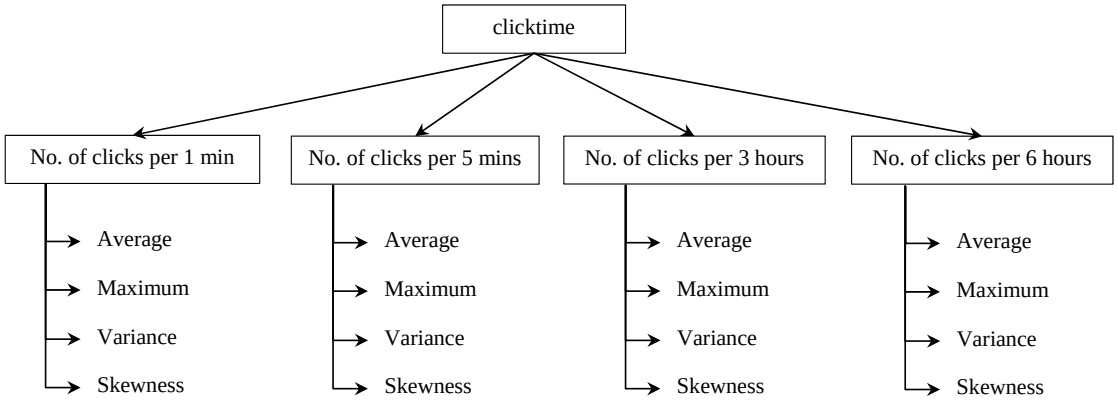


Figure 3: Feature creation from the `clicktime` attribute.

Feature	Description
`MaxSameIPClicks`	Maximum number of clicks from all unique IP addresses associated with a publisher
`NoOfIPs`	Number of clicks from all unique IP addresses associated with a publisher
`ClickOverIPRatio`	Ratio of the number of clicks over the number of unique IP addresses for a publisher
`EntropySameIPClicks`	Entropy of the number of clicks from all IP addresses associated with a publisher
`VarSameIPClicks`	Variance of the number of clicks from all IP addresses associated with a publisher

Table 10: Features derived from the `numericip` attribute.

IP addresses are dynamically allocated when users connects via an internet service provider (ISP), it is not unusual for the same user to have different IP addresses. For a given 3-day period, we observed changes in the IP addresses and number of clicks from a given IP address for a given publisher id. We used parametric measures over IP address attribute (`numericip`) to define the behavior of a publisher. Table 10 lists the feature set created from the `numericip` attribute.

Some fraudulent publishers may try to increase their reward by clicking repeatedly on an advertisement but all of these clicks might come from the same IP. From the data, we observed that many clicks originating from the same IP or an unusually large click to IP ratio tend to be associated with fraudulent behavior, and may place the associated publisher under suspicion. We also observed that lower variance in the number of clicks from each IP is indicative of a legitimate publisher, whereas higher variances might indicate a fraudulent publisher. Similarly, the entropy for the distribution of the number of clicks originating from each IP can be another useful indicator of fraudulent activity.

4.1.3 Attribute: `deviceua`

The `deviceua` attribute is the phone model that the visitors use to browse the web and click on advertisements. As mentioned, a fraudulent visitor might use one phone, but with many dynamically allocated IP addresses. Thus, we use the following measures to derive features from `deviceua` attribute in the set of attributes: `MaxSameAgentClicks`,

`MaxSameAgentClicks`, `VarSameAgentClicks`, and `SkewnessSameAgentClicks`. These features also calculated in a similar way to those for the `numericip` attribute.

4.1.4 Other Attributes

We also used the same method to generate features for `country` and `campaignid`. On the other hand, each publisher is assigned to only one channel, thus we avoid taking `channel` to derive more attributes. Instead, we defined the prior probability of being fraud for a given channel based on the training set. That is, we computed number of visitors for each channel, and then the number of fraudulent publishers in that set to obtain the prior probability. Finally, for the `referredurl` attribute we derived `ReferrerOverClickRatio` by computing the number of referred clicks over the total number of clicks for a given publisher.

At the end of feature extraction process, we had 41 different features created from different individual and set of attributes from the data set. The full list of those 41 features is provided in `www.dnagroup.org/PDF/FDMA12_TeamMasdar_AppendixA.pdf`.

4.2 Method

In this section, we describe our machine learning framework for addressing fraud detection problem. We first present our base classification models, followed by data resampling strategies for handling imbalanced label distribution. Finally, we discuss on ensemble learning methods that combine several base classifiers for improved detection performance.

4.2.1 Base Classifier

Our approach to detecting fraud consists of employing contemporary classification models over data derived from click database. We tuned the parameters of these models such that they work robustly with the train and validation data sets, and can generalize well to unseen test set. We tried a range of different model parameters that yielded the highest precision and area under the receiver operating characteristics (AUC) curve, with low standard deviation to ensure performance consistency.

We explored a variety of classification methods including decision tree, neural network, and support vector machine. For each method, we also employed different learning algorithms. Our preliminary experiments on the train and validation sets revealed that the decision tree technique is particularly promising and gave good prediction results. As such, we shall focus on decision tree-based models in the subsequent sections.

4.2.2 Resampling

As mentioned, only a small fraction of publishers are fraudulent, and the skewed nature of the data would drive the prediction model to be more biased towards the majority class. In light of this issue, we used various resampling strategies such as up/downsampling and the synthetic minority oversampling technique (SMOTE) (Chawla et al., 2002). Upsampling was done by replicating samples from the minority class until their number is equal to the that of the majority class. Conversely, downsampling randomly discards the majority class until the class distribution is balanced. In our experiments, we tried both up/downsampling

Base tree	Ensemble (meta) learner
C4.5 tree	Bagging, Metacost, Logitboost, random subspace
REP tree	Bagging, Metacost, Logitboost
Random forest	Bagging, Metacost, Logitboost

Table 11: Decision tree algorithms and the corresponding meta-learning algorithms.

and SMOTE, followed by shuffling of the data instances. Results obtained with and without sampling methods shall be discussed in Section 4.3.

4.2.3 Ensemble Learning

Decision tree-based algorithms are weak learners known for their stability issues. To improve a weak classifier, one may construct many weak classifiers instead of a single one, and to combine them into a powerful decision rule. Recently, a number of combining techniques have been developed, the most popular being bagging (Breiman, 1996), boosting (Freund and Schapire, 1996) and the random subspace method (Ho, 1998). In bagging, one samples the training set, generating random independent bootstrap replicates, constructs the classifier on each of these, and aggregates them by a simple majority vote in the final decision rule. In boosting, classifiers are constructed on weighted versions of the training set, which depend on previous classification results. In the random subspace method, classifiers are constructed in random subspaces of the data feature space. These classifiers are typically combined by simple majority voting.

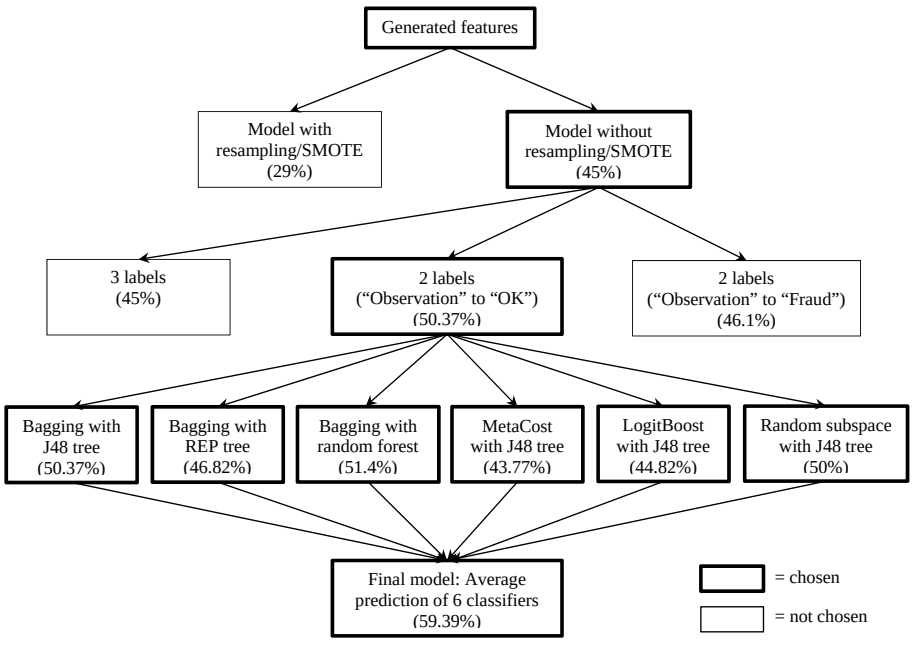


Figure 4: The final classification model comprising an ensemble of six learners.

For bagging, we considered the standard algorithm (Breiman, 1996) and MetaCost (Domingos, 1999), a special type of bagging that produces a single cost-sensitive classifier of the base learner, giving the benefits of fast classification and interpretable output. For boosting, we considered the LogitBoost method (Friedman et al., 2000), which treats AdaBoost (Freund and Schapire, 1995) as a generalized additive model and applies the cost functional of logistic regression. Finally, we used the standard random subspace method (Ho, 1998), with decision trees as the base learners.

We built classification models using different combinations of base learners and meta-learning algorithm, as shown in Table 11, and evaluated them using the train and validations sets. The base learners are C4.5 decision tree (Quinlan, 1993), reduced error pruning (REP) tree (Su and Zhang, 2006), and random forest (Breiman, 2001). Our final classifier consists of an ensemble of six models, which gave the best overall performance in terms of precision, recall, and area under the ROC curve. The goal of evaluating all three measures is to build a model that can detect high percentage of `Fraud` cases, while maintaining high degree of precision. The "journey" towards the final classification model and the scores of the constituent learners are presented in Figure 4.

4.3 Results and Discussion

In this section, we summarize the key observations and results obtained using our proposed machine learning approach as described in Section 4.2.

4.3.1 Effect of Resampling

To see the effect of resampling on the model performance, we conducted experiments with both up/downsampling and SMOTE (Chawla et al., 2002). The resampling and SMOTE performed very well on the training set but performed badly on the validation set. That is, results using the original data were found to be over 15% better than those obtained using resampling and SMOTE.

4.3.2 Two- vs. Three-Class Task

Different models were trained using data sets containing only 2 classes (i.e., `OK` and `Fraud`), and all 3 classes (i.e, `OK`, `Observation` and `Fraud`). For the 2-class setting, two approaches were taken: In the first approach, all `Observation` cases were converted to `OK`, and in the second approach all `Observation` cases were treated as `Fraud` cases. In the 3-class setting, we simply used the labels provided with the data. Of all three approaches, the 2-class data set gave the best performance. The AP score obtained using the J48 tree for the (`Observation` $\rightarrow$ `OK` conversion case was 50.37%, and for the `Observation` $\rightarrow$ `Fraud` case we got 46.1%. When we used all 3 classes, the precision score was 45%. Thus, we deem that converting all `Observation` cases to `OK` was the best approach.

4.3.3 Performance

We evaluated the prediction performance of different algorithms for all data sets. Few algorithms which gave best result alone are mentioned above. There were many algorithms with very low true positive and false negative rates, thus giving very high precision scores.

These algorithms were able to obtain high precision because of their low false positive value. We were only interested on algorithms which have high true positive rates and precision. The precision scores of the different algorithms when applied on the 2-class validation set were C4.5 tree: 50.37%, REP tree: 46.82%, and LogitBoost: 44.82%, as per Figure 4.

With our ensemble approach, we were able to pass the baseline score but none of the algorithms alone was able to obtain precision higher than 50.37%. We analyzed the results and found that every algorithm has a drawback, which was either a high false positive rate or a lower rate of true positives. This indicated that choosing any one of the algorithms represented a trade-off between high sensitivity on the one hand, and higher precision on the other. Subsequently, we combined the results from the different algorithms trained using the 2-class data set. Six different algorithms were chosen which obtained higher values for precision, recall and AUC when evaluated alone. This method proved to be the best as we obtained an average precision of **59.39%** on the validation set. It also performed well on the final test set, achieving a score of **46.42%**.

5. Third Winner's Entry

This section describes the third winner's entry, from the preprocessing and feature extraction methods to the classification algorithms and empirical results, as per the previous sections.

5.1 Preprocessing and Feature Extraction

Click fraud can be generated through various ways (Dave et al., 2012), such as (1) botnets (where malware on the user's computer clicks on ads in the background), (2) tricking or confusing users into clicking ads (e.g., on parked domains), and (3) directly paying users to click on ads. To deal with various fraud patterns, we first need to extract publisher's feature from various statistics such as mean, standard deviation, count from different views and at different time granularities. We can analyse these features and choose the most discriminative ones to build an effective classifier.

5.1.1 Click Statistics by Publisher

We calculate the basic click statistics of each publisher (i.e., `publisherid`), unique count of attribute such as `numericip`, `country`, `deviceua`, `referredurl`, `campaignid` and total visit. The features used in this work and their descriptions are shown in Table 12:

Feature	**Description**
unique_count(numericip)	unique count of the IP
unique_count(country)	unique count of the country
unique_count(deviceua)	unique count of the deviceua
unique_count(referredurl)	unique count of the referredurl
unique_count(campaignid)	unique count of the campaignid
total_visit	count of the click log's row

Table 12: Features derived from the click statistics of each publisher.

5.1.2 Click Statistics by numericip

The fraudulent visitors may visit the advertisements from the same IP address. In order to capture this, we calculate the average access, standard deviation, counting by grouping each IP for each publisher in different time granularity (by second, by min, by day). For example, in the Table 13, suppose it is the full click log for the publisher "$8kxij$", we can get the average access by IP in minute granularity is 5, standard deviation is 0, and the counting for the IP $2,919,155,822$ visit is 5. We can also get the same statistics in different time granularity such as by day, hour, second. The feature created is shown in Table 14.

id	numericip	deviceua	campaignid	usercountry	clicktime	channel	referredurl
8kxij	2,919,155,822	Nokia X6	8gava	us	2012-02-09 07:23:13.0	mc	?
8kxij	2,919,155,822	Nokia X6	8gava	us	2012-02-09 07:23:13.0	mc	?
8kxij	2,919,155,822	Nokia X6	8gghw	us	2012-02-09 07:23:13.0	mc	?
8kxij	2,919,155,822	Nokia X6	8gghw	us	2012-02-09 07:23:19.0	mc	?
8kxij	2,919,155,822	Nokia X6	8gava	us	2012-02-09 07:23:19.0	mc	?

Table 13: Example of fraud pattern: Clicking from the same IP address.

Feature	**Description**
avg_IP_sec	average visit by IP per second
std_IP_sec	standard deviation of average visit by IP per second
count_IP_sec	sum of visit count (larger that 2) by IP per second
avg_IP_min	average visit by IP in minute level
std_IP_min	standard deviation of average visit by IP per minute
count_IP_min	sum of visit count (larger that 2) by IP per minute
avg_IP_hour	average visit by IP per hour
std_IP_hour	standard deviation of average visit by IP per hour
count_IP_hour	sum of visit count (larger that 2) by IP per hour
avg_IP_day	average visit by IP per day
std_IP_day	standard deviation of average visit by IP per day
count_IP_day	sum of visit count (larger that 2) by IP per day

Table 14: Click statistics by IP address.

We also conjecture that fraudulent visitors will visit the advertisements not from the same IP address but from the same subnetwork (see Table 15). For this, we tried to obtain the same statistics by each subnetwork instead of IP. The subnetwork of different granularity can be obtained by dividing the IP by 1,000 or 1,000,000 and rounding the result. Based on this, we got the same statistics as shown in Table 14 for different subnetwork.

id	numericip	deviceua	campaignid	usercountry	clicktime	channel	referredurl
8jk0d	1,917,853,114	MSIE_6.0	8gp6q	cn	2012-02-09 11:52:27.0	se	?
8jk0d	1,917,853,057	MSIE_6.0	8gp6q	cn	2012-02-09 11:56:13.0	se	?
8jk0d	1,917,853,952	MSIE_6.0	8gp6q	cn	2012-02-09 11:58:49.0	se	?
8jk0d	1,917,853,022	MSIE_6.0	8gp6q	cn	2012-02-09 12:06:55.0	se	?

Table 15: Example of fraud pattern from the same subnetwork.

5.1.3 Click Statistics by deviceua

Sometimes the malicious publishers use the same `deviceua` but different IP to access a website at different time periods. For example, in Table 16, the fraudulent visitor using same `deviceua` MSIE 6.0 visits the same publisher using different IP at different times, while there is no other `deviceua` visiting this publisher during these time period. To capture this behavior, we sorted the click by `clicktime` and then `deviceua`. Afterwards, for each click log row, we compared with the next click log row. If the `deviceua` is the same, we kept the current row, otherwise it is removed. We can then calculate the portion of the filtering rows over the total rows for each publisher; we call this feature *deviceua*1. For example, in the Table 16, after filtering, we have 8 rows left, so the final result will be 8/11. In addition, we tried to sort the click data by deviceua only and calculate the statistics again as discussed above. This feature is named as *deviceua*2.

id	numericip	deviceua	campaignid	usercountry	clicktime	channel	referredurl
8jk0d	1,917,852,952	MSIE_6.0	8gp6q	cn	2012-02-11 02:55:50.0	se	?
8jk0d	1,917,853,022	MSIE_6.0	8gp6q	cn	2012-02-11 02:56:36.0	se	?
8jk0d	1,917,853,060	MSIE_6.0	8gp6q	cn	2012-02-11 03:53:12.0	se	?
8jk0d	1,917,852,993	MSIE_6.0	8gp6q	cn	2012-02-11 04:49:42.0	se	?
8jk0d	701,380,683	Nokia2600c	8k7xb	ng	2012-02-11 04:51:58.0	se	?
8jk0d	1,917,852,993	MSIE_6.0	8gp6q	cn	2012-02-11 05:33:51.0	se	?
8jk0d	1,917,853,114	MSIE_6.0	8gp6q	cn	2012-02-11 06:30:02.0	se	?
8jk0d	1,917,853,146	MSIE_6.0	8gp6q	cn	2012-02-11 07:09:23.0	se	?
8jk0d	1,917,853,146	MSIE_6.0	8gp6q	cn	2012-02-11 07:31:21.0	se	?
8jk0d	1,917,852,993	MSIE_6.0	8gp6q	cn	2012-02-11 07:53:16.0	se	?
8jk0d	1,917,852,952	MSIE_6.0	8gp6q	cn	2012-02-11 07:55:14.0	se	?

Table 16: Example of fraud pattern from one `deviceua` but different IP at different times.

5.1.4 Click Statistics by campaignid

The click data also suggest that malicious publishers may access the same advertisement campaign repeatedly using the same IP address and phone agent at a brief time period. This is shown in Table 17, where `campaignid` "8*gkwy*" was accessed many times by the same `numericip` and `deviceua`. These clicks are thus likely to be illegitimate. Accordingly, we calculated the average access, standard deviation, counting by grouping `campaignid` for each publisher in different time granularity (by second, by minute, by day). We then obtained results similar to Table 14, grouped by `campaignid` instead of by `numericip`.

id	numericip	deviceua	campaignid	usercountry	clicktime	channel	referredurl
8kv5w	3,251,257,947	Iphone	8gkwx	dk	2012-02-09 02:36:05.0	co	?
8kv5w	3,251,257,947	Iphone	8gkwx	dk	2012-02-09 02:36:16.0	co	?
8kv5w	3,251,257,947	Iphone	8gkwx	dk	2012-02-09 02:36:40.0	co	?
8kv5w	3,251,257,947	Iphone	8gkwx	dk	2012-02-09 02:38:01.0	co	?
8kv5w	3,251,257,947	Iphone	8gkwx	dk	2012-02-09 02:38:09.0	co	?
8kv5w	3,251,257,947	Iphone	8gkwx	dk	2012-02-09 02:38:26.0	co	?

Table 17: Example of fraud pattern on the same campaign ID.

5.1.5 Click Statistics by `numericip+deviceua`

The `numericip` corresponds to a public IP address that may be assigned to different clickers at different time periods. Hence, an IP address may not uniquely identify a clicker. A better estimate is to use `numericip+deviceua` for identification. Using this identifier, we calculated the average access, standard deviation, counting by grouping `numericip+deviceua` for each publisher in different time resolutions. Again, we obtained the statistics similar to Table 14, grouped by `numericip+deviceua` instead of by `numericip`.

5.2 Method

We employed a linear blending of many predictive models, and our approach consists of three main steps. In the first step, we created our own validation set from the training set (since the validation set provided by the competition organizer has no label). Specifically, the internal validation set was generated by randomly selecting a subset of publishers in the training set such that the original class (status) distribution is maintained (i.e., stratified sampling). We chose this approach instead of cross-validation procedure for simplicity and computational efficiency. In the second step, we optimized the meta-parameters of the models and picked the configuration that yielded the best result on the internal validation set. Finally, using the best meta-parameters found, the third step consists of retraining the model on the combined train and internal validation sets. The model was then evaluated on the test set and the prediction scores were recorded for submission.

5.2.1 Classification Algorithms

The classification methods we considered for this competition are listed in Table 18. Most of these techniques are tree/rule-based classifiers (except for Bayesian network and resilient propagation (RPROP)), which we selected due to their relatively good performances. All methods were trained and evaluated using the three step procedure described above.

5.2.2 Blending

To combine the predictions of all the classification algorithms in Table 18, we used a linear blending method. A simple way to compute the final prediction is to average all predictions in the ensemble, but better results can be achieved by computing weighted sum of the predictions. In this work, we performed blending via weighted sum approach, where the weight coefficients were learned using linear regression. Furthermore, we normalized all inputs (i.e., the predictions) to [0, 1]. We optimized the weight coefficients based on the average precision for 10-fold cross-validation.

5.3 Results and Discussion

Below we summarize our main experimental results, focusing on our mode performances as well as findings on several important features indicative of fraudulent behavior.

Type	Method	Description
Single	FT tree (Gama, 2004)	A type of decision trees with logistic regression functions at the inner nodes and/or leaves
	REP tree (Su and Zhang, 2006)	A decision tree is built using information gain or variance and then pruned using reduced-error pruning (REP) method
	Decision table (Kohavi, 1995)	Rule-based classifier that uses simple decision table majority voting, providing a precise yet compact representation
	Bayesian network (Neapolitan, 2003)	Directed graphical model for encoding statistical dependencies among a set of variables
	RPROP (Riedmiller and Braun, 1993)	Learning for feed-forward neural networks that locally adapts the weight updates based on the error function's behavior
Ensemble	LAD tree (Holmes et al., 2002)	A multi-class alternating decision tree that is built using the LogitBoost strategy
	NB tree (Kohavi, 1996)	Decision tree with Naive Bayes classifiers at the leaf nodes
	Random forest (Breiman, 2001)	Combination of tree predictors such that each tree depends on the values of a random vector sampled independently
	Rotation forest (Rodrguez et al., 2006)	Classifier ensemble based on feature extraction. Features are split into subsets and principal component analysis is applied to each
	Tree ensemble (Berthold et al., 2009)	Variant of random forest whereby each tree model is learned on different set of records and/or attributes

Table 18: Classification methods employed for the FDMA 2012 Competition.

5.3.1 Performance

The performances of the individual models and their blending are shown in Table 19. For the single models, RPROP performed the best, suggesting that neural network approach is suitable for the detection task. Among the ensemble approaches, only the tree ensemble outperformed RPROP, but the improvement was marginal. Lastly, the linear blending approach gave better result than all its constituent models. Hence, we chose it as our final model for the competition submissions. We obtained an average precision of **62.21%** and **46.15%** on the validation and test sets, respectively.

Type	Method	Average Precision
Single	FT tree	36.3%
	REP tree	35.8%
	Bayes network	33.7%
	RPROP	48.3%
Ensemble	LAD tree	37.0%
	NB tree	37.9%
	Random forest	47.7%
	Random subspace	38.9%
	Rotation forest	42.9%
	Tree ensemble	49.3%
Ensemble of ensemble	Blending	52.3%

Table 19: Performance for different algorithms on the internal validation set.

5.3.2 Important Features

To prune inconsequential features and improve prediction performance, we performed iterative feature elimination (Guyon and Elisseeff, 2003) as follows. Initially, classification is performed using the complete N features. In the next $N-1$ iterations, each of the input features is disabled once. Then the algorithm discards the feature that influences the prediction result the least (in this case giving the smallest degradation in average precision). The subsequent $n-2$ iterations follow where each of the remaining features is ruled out once. The total number of iterations is therefore $N*(N+1)/2-1$. Finally, we had all computed levels of the feature elimination together with the average precision. We specified an error threshold and select the level with fewest features that has a prediction error below the threshold. Table 20 lists the final set of features after the elimination, which represent the important variables potentially correlated with fraudulent cases.

Feature	Description
unique_count(referredurl)	Unique count of referredurl
unique_count(campaignid)	Unique count of campaignid
unique_count(country)	Unique count of country
total_visit	Count of the click log's row
count_ip_hour	Sum of visit count (> 2) by numericip per hour
count_ip_ag_sec	Sum of visit count (> 2) by numericip+deviceua per second
count_ip_ag_day	Sum of visit count (> 2) by numericip+deviceua per day
count_sip2_sec	Sum of visit count (> 2) by subnetwork (divided by 1,000,000) per second
count_sip2_min	Sum of visit count (> 2) by subnetwork (divided by 1,000,000) per minute
count_sip2_hour	Sum of visit count (> 2) by subnetwork (divided by 1,000,000) per hour
count_sip_day	Sum of visit count (> 2) by subnetwork (divided by 1,000) per day
avg_sip2_day	Average visit by subnetwork (divided by 1,000,000) per day
avg_ip_ag_min	Average visit by numericip+deviceua per minute
avg_ip_ag_day	Average visit by numericip+deviceua per day
avg_campaignid_min	Average visit by campaignid per minute
deviceua1	Statistics for click data sorted by time and deviceua, as discussed in Section 5.1.3
deviceua2	Statistics for click data sorted by deviceua, as discussed in Section 5.1.3

Table 20: Final feature set after backward elimination.

6. Runner-up's Entry

In this section, we elaborate the approach and results obtained by the runner up (i.e., fourth winner), with similar organization as that of the previous sections.

6.1 Preprocessing and Feature Extraction

For each publisher, we considered the following basic attributes: (1) total number of clicks, (2) number of clicks from the same computer (inferred from attribute `numericip`), (3) distinct IP addresses (inferred from `numericip`), (4) distinct parts of the IP addresses, (5) publisher's channel type (inferred from `channel`), (6) phone models used by clickers (inferred from `deviceua`); (7) advertisement campaign (inferred from `campaignid`); and (8) number of clicks from different countries (*geo tracking*; inferred from `usercountry`).

Click fraud, notably manual click fraud, is known to correlate with the geographical location of the clicker. The heatmaps in Figure 5 visualize the geo tracking of clicks from the training set with respect to the status of the publisher. For all three groups (`Fraud`, `Observation`, and `OK`), the clicks originate from only a few countries (cf. columns). For

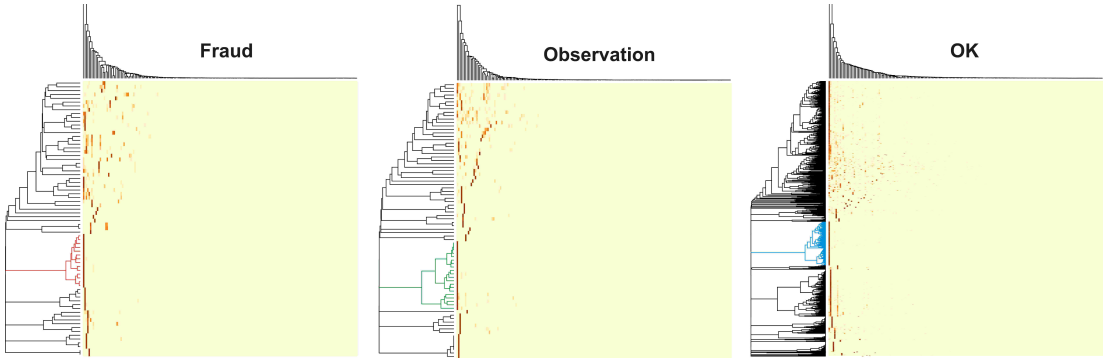


Figure 5: Geo tracking of clicks. Columns show the 197 countries of origin; rows show the publishers from the training set. For each publisher, the click percentage from each country is color-coded. Darker colors reflect higher percentages; yellow is 0% and dark red is 100%. Rows and columns are clustered based on complete linkage. For status `OK`, only those publishers with at least 50 clicks are shown.

about half of the publishers in each group (cf. rows), the clicks are distributed across these countries, while for the rest the majority of clicks come from only a small number of countries. In fact, we found such clusters in all three groups.

Geo tracking alone, however, is not reliable for click fraud detection. There may be various explanations for the observed clusters, such as (obviously) the type and the target of the advertisements. Furthermore, malicious scripts could generate fraudulent clicks, and these scripts could run on computers in different geographical locations. In that case, we might fail to detect any clusters.

6.1.1 Click Profiles

For each publisher and each unique IP address, we investigated the *click profile*, that is, the time delay between consecutive clicks. For the majority of fraudulent publishers in the training set, we observed that the number of unique IP addresses was below 3000. Only for two fraudulent publishers, we observed that clicks were coming from more than 3000 unique IP addresses. To derive the click profile, we discarded all publishers for which we observed clicks coming from more than 3000 unique IP addresses. This approach was of course far from being ideal, but it reduced the computational time considerably.

6.1.2 Long Click Profile

We assumed that many consecutive clicks from the same IP address in short time intervals were suspicious. So for each publisher, we counted how many clicks from the same IP address occurred each day in less than 5s, between 5s and 10s, between 10s and 20s, between 20s and 30s, and so on up to the interval > 300s. Furthermore, we required that at least 10 clicks must have come from each IP address. Table 21 shows an example of a set of consecutive clicks from the same IP address for the publisher 8ih09.

id	deviceua	campaignid	usercountry	date	clicktime	category	referredurl	same URL	gap
14090783	Opera_Mini	8flxe	fr	2012-02-09	08:53:21.0	in	24940f5c4q688oc8	0	0
14096272	Opera_Mini	8flxd	fr	2012-02-09	09:02:01.0	in	24940f5c4q688oc8	1	520
14096576	Opera_Mini	8flyo	fr	2012-02-09	09:02:30.0	in	24940f5c4q688oc8	1	29
14135449	Opera_Mini	8flyo	fr	2012-02-09	09:59:33.0	in	24940f5c4q688oc8	1	3423
14149730	Opera_Mini	8flyo	fr	2012-02-09	10:18:31.0	in	24940f5c4q688oc8	1	1138
14153291	Opera_Mini	8flyp	fr	2012-02-09	10:23:32.0	in	14qhcdsqvou88kos	0	301
14153584	Opera_Mini	8flyu	fr	2012-02-09	10:23:57.0	in	14qhcdsqvou88kos	1	25
14154864	Opera_Mini	8flyp	fr	2012-02-09	10:25:42.0	in	24940f5c4q688oc8	0	105
14197361	Apple_iPhone	8flyo	fr	2012-02-09	11:23:23.0	in	3gza50jfnzcw44wc	0	3461
14197602	Apple_iPhone	8jdc9	fr	2012-02-09	11:23:42.0	in	14qhcdsqvou88kos	0	19
14198413	Apple_iPhone	8flxc	fr	2012-02-09	11:24:50.0	in	14qhcdsqvou88kos	1	68
14198584	Apple_iPhone	8flyp	fr	2012-02-09	11:25:05.0	in	14qhcdsqvou88kos	1	15
14199113	Apple_iPhone	8flys	fr	2012-02-09	11:25:51.0	in	14qhcdsqvou88kos	1	46
14201181	Apple_iPhone	8flxe	fr	2012-02-09	11:28:31.0	in	14qhcdsqvou88kos	1	160
14206726	Apple_iPhone	8flxf	fr	2012-02-09	11:35:50.0	in	23ge85exom8084s0	0	439
14217945	Apple_iPhone	8flyu	fr	2012-02-09	11:50:32.0	in	23ge85exom8084s0	1	882
15754245	Opera_Mini	8gpd5	fr	2012-02-10	10:18:52.0	in	3cu2xmfag82sosk4	0	0
15764598	Opera_Mini	8gpd5	fr	2012-02-10	10:33:56.0	in	4jyefurnmxkwoo4w	0	904
15768527	HTC_Vision	8gdka	fr	2012-02-10	10:39:47.0	in	1rbl5y69ej34gg8w	0	351
15768829	HTC_Vision	8gpd5	fr	2012-02-10	10:40:17.0	in	3cu2xmfag82sosk4	0	30
15777019	Opera_Mini	8gpd5	fr	2012-02-10	10:52:51.0	in	3cu2xmfag82sosk4	1	754
783581	Opera_Mini	8gpd5	fr	2012-02-11	10:34:38.0	in	1rbl5y69ej34gg8w	0	0
901789	SPH-P100	8gpd5	fr	2012-02-11	13:01:12.0	in	1rbl5y69ej34gg8w	1	8794
902642	SPH-P100	8gdka	fr	2012-02-11	13:02:09.0	in	3cu2xmfag82sosk4	0	57
903031	SPH-P100	8gpd5	fr	2012-02-11	13:02:38.0	in	3cu2xmfag82sosk4	1	29

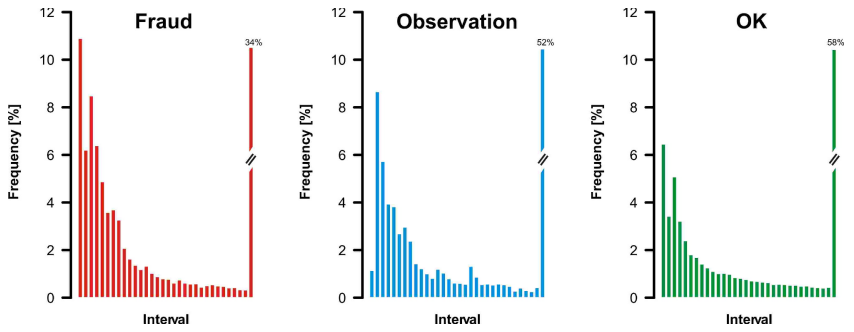
Table 21: Example of 25 consecutive clicks from the same IP for publisher 8ih09.

For the clicks in Table 21, we see the following interval frequencies: $(0, 0, 2, 4,\ 0, 1, 1, 1,\ 0, 0, 0, 1,\ 0, 0, 0, 0, 0,\ 1, 0, 0, 0, 0,\ 0, 0, 0, 0, 0,\ 0, 0, 0, 0, 0,\ 0, 0, 0, 11)$, where the i^{th} element represents the count for the i^{th} interval. For example, we observe two clicks with a time gap between 10s and 20s (id 14197602 with 19s and id 14198584 with 15s). We also tracked the relative counts, that is, the interval counts divided by the respective set size. For each publisher, we proceeded analogously for all IP addresses and derived the cumulative interval frequency count, which is the *long click profile*. For the publisher 8ih09, for instance, this profile is $(0, 0, 17, 25,\ 26, 30, 20, 21,\ 19, 14, 7, 5,\ 9, 9, 7, 2,\ 7, 8, 4, 3,\ 7, 3, 3, 2,\ 1, 1, 2, 3,\ 1, 1, 1, 41)$, where the i^{th} element represents the cumulative count for the i^{th} interval. We saw that clicks from the same IP address tend to occur in relatively short sequences for 8ih09, which is in fact labeled as `Fraud`. By contrast, let us consider the profile of a publisher labeled as `OK`, for example, 8i7wi. Its long click profile is $(0, 0, 4, 3,\ 0, 0, 2, 1,\ 0, 1, 1, 0,\ 1, 1, 0, 1,\ 0, 0, 0, 0,\ 0, 0, 0, 0, 0, 2, 0, 1,\ 1, 0, 0, 76)$; here, short click sequences are less frequent. Another publisher, 8jkh1, is labeled as `Observation`, and its long click profile is $(6, 13, 11, 19,\ 22, 8, 13, 13,\ 21, 17, 13, 14,\ 11, 7, 12, 9,\ 5, 8, 7, 7,\ 5, 8, 1, 5,\ 6, 5, 2, 1,\ 2, 2, 5, 404)$.

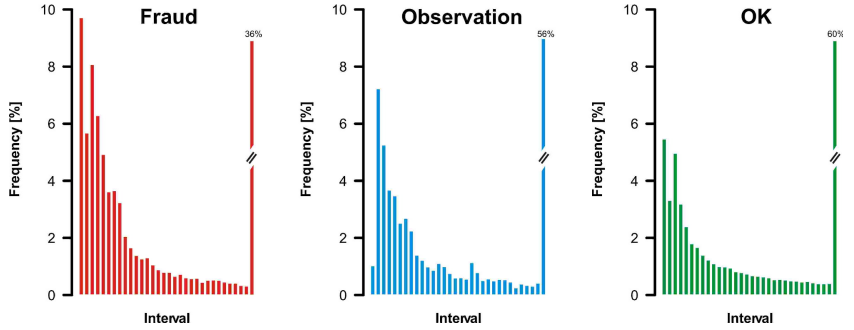
Figure 6(a) shows the click frequencies per interval, derived from all long click profiles per group. We see that, overall, consecutive clicks that follow one another rather quickly occur more often for fraudulent publishers than for those with status `Observation` or `OK`.

6.1.3 Short Click Profile

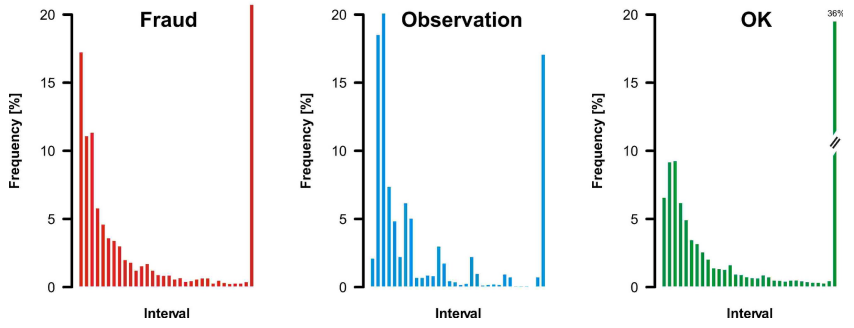
The short click profile was derived in the same way as the long click profile, except that at least 5 (and not 10) consecutive clicks must have come from the same IP address. Figure 6(b) shows the click frequencies per interval, derived from all short click profiles per group. Again, we observed that quick consecutive clicks occur more often in fraudulent publishers than in those with status `Observation` or `OK`.



(a)



(b)



(c)

Figure 6: Click frequency per interval based on (a) long click profiles, (b) short click profiles, and (c) click profiles from the same URL. The intervals are $[0s, 5s],]5s, 10s],]10s, 20s], ...]300s, +\infty)$.

6.1.4 Clicks Coming from the Same URL

The long and short click profiles ignored the URL where an advertisement had been clicked on. It is possible, however, that a fraudulent (human) clicker does not navigate too often from one web site to another. To derive a pattern of clicks coming from the same URL, we

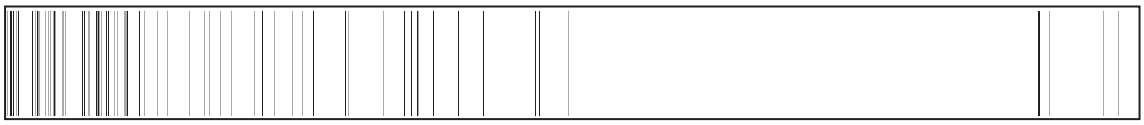


Figure 7: Publishers (from the training set) ranked from left to right based on decreasing values of redflag. Solid black bars denote publishers with status `Fraud`; gray bars denote publishers with status `Observation`; white bars denote publishers with status `OK`. Fraudulent publishers and those under observation are significantly concentrated towards the left hand side ($P < 0.001$, Kruskal-Wallis test).

asked for each publisher: how many clicks came from the same IP address *and* the same URL in less than 5s, between 5s and 10s, between 10s and 20s, and so on up to the interval > 300s. At least 5 clicks must have come from each IP address. A problem with this approach, however, was that the URL information was missing for many clickers.

Consider again Table 21. The column *referredurl* contains encrypted information about the URL. The column *same URL* contains a flag, indicating whether the clicker has left ($= 0$) or stayed ($= 1$) on the same URL. We considered only those time gaps that refer to the same URL; thus, we ignored the gap of 19s for id 14197602, for example, because the clicker has navigated from 3gza50jfnzcw44wc to 14qhcdsqvou88kos.

Figure 6(c) shows the click frequencies per interval, derived from all click profiles from identical URLs. Similarly, we observed that quick consecutive clicks occur more often in fraudulent publishers than in those with status Observation or OK.

6.1.5 Redflag

For each publisher, we checked if there were at least 5 clicks from the same IP address and the same URL and with a time gap of less than 20s. If so, we incremented a flag (*redflag*) for that publisher. In Figure 7, publishers from the training set are ranked from left to right based on decreasing values of redflag. Fraudulent publishers (black) and those under observation (gray) are concentrated towards the left hand side. Thus, the larger redflag, the more suspicious is the publisher. Redflag is in fact a significant indicator of fraudulent behavior ($P < 0.001$, Kruskal-Wallis test).

6.2 Method

According to the rules of the competition, each team was allowed to submit the predictions of two models for the final evaluation. Below we describe our two models.

For the first model, we used only the basic attributes and the long click profile. The algorithm was random forests (Breiman, 2001), which first generates a number of unpruned decision trees from bootstrap samples of the train set. Each tree uses a random subset of features. Subsequently, the algorithm combines the trees into one "forest" whose predictions stem from aggregating the predictions of the individual trees.

Because of the drastic class imbalance in the train set, either cost-sensitive learning or up/down-sampling is necessary. For random forests, both approaches were shown to be on par in terms of performance (Chen et al., 2004). Up/down-sampling, however, is computationally less expensive because each tree uses only a small subset of the train set. Given the time constraints, we adopted only one approach: up/down-sampling. A multitude of sample sizes were tested, and the models were selected based on the out-of-bag (OOB) error rate. Our preliminary results suggested that the differences between the predictive performance of the models were not so large. From all models, we finally selected seven (Table 22) and combined them into one *ensemble of random forests*. This is model #1, and it was submitted for the final evaluation.

In addition to the reduced data set, we included the following data for our second model: short click profile, click profile from identical URLs, and redflag. The algorithm was random forest with up/down-sampling. We tested again various parameters (number of trees, terminal node size, sampling ratios) and selected the final model on the basis of the OOB error rate. The final model consisted of 50 trees with a terminal node size of 3. The percentages for the bootstrap samples were: 97% for class `Fraud`, 88% for `Observation`, and 61% for `OK`. This is model #2, and it was submitted for the final evaluation.

#	ntree	nodesize	Fraud	Observation	OK	OOB error
1	250	5	90%	63%	41%	4.64%
2	250	5	90%	63%	41%	4.67%
3	250	3	83%	75%	51%	4.64%
4	250	3	69%	38%	34%	4.48%
5	250	3	69%	38%	34%	4.74%
6	250	3	90%	44%	51%	4.54%
7	250	4	83%	63%	68%	4.45%

Table 22: Individual random forests with up- and down-sampling (ntree: number of trees in each forest; nodesize: number of terminal nodes in each tree).

6.3 Results and Discussion

How well can the three click profiles discriminate the publishers? To address this question, we trained three random forests, each using only one of these click profiles and no further data. Each model consisted of 250 trees, each with 3 terminal nodes. The sampling was 90% for `Fraud`, 75% for `Observation`, and 61% for `OK`. Table 23 shows the classification results of these three models (not submitted for the final evaluation).

6.3.1 Performance

Table 24 shows the classification results of the two models that were submitted for evaluation on the final test set. Model #2 (single random forest, 50 trees) achieved a better performance on both the training and the validation set. However, the performance on the final test remarkably deteriorated, compared with the performance of model #1 (ensemble of random forests, 1750 trees) that used only the reduced data set.

Click profile	OOB error	Average precision (validation)
Long	4.61%	47.67%
Short	4.64%	46.77%
Clicks from same URL	4.51%	47.89%

Table 23: Classification results of random forests using only click profiles.

Model #	OOB error	Average precision (validation)	Average precision (test)
1	4.61%	49.99%	42.01%
2	3.66%	51.55%	36.94%

Table 24: Classification results of the final two models.

6.3.2 Remarks

Two problems made this competition particularly challenging. First, there is the problem of concept drift. The train, validation, and test data sets came from different time windows. A publisher may appear across different sets with a similar pattern, but the provided, actual status label may be different. For example, a publisher may have been labeled as `OK` in early February and then as `Observation` in late February, perhaps because this publisher showed a suspicious pattern.

Second, there is no "ground truth" about which publishers are indeed really fraudulent, which are truly `OK`, and which should be under scrutiny. The real status labels were generated by some fraud detection algorithm (here called "ground model", for short), but how reliable are its predictions? Consider the following example. For one publisher in the train set, we observed 5706 total clicks. All clicks came from the same IP address, and 3307 (58%) occurred in less than 5s. Also, is it not suspicious that 1086 consecutive clicks (19%) have an interval of even 0s? It is tempting to speculate that a script generated these clicks. Surprisingly, the status of that publisher is `OK` in the train set. In the validation set, the click profiles of that publisher may also raise attention: more than 70% of all 1149 clicks – from one and only IP address – occurred in less than 5s, with 248 clicks having a time gap of 0s. Several similar examples of questionable status labels could be found.

7. Organizer's Entry

In this section, we describe the work by the FDMA 2012 organizer's research team, which was carried out independently from the other competition participants. The description will follow the same organization as that of the previous sections.

7.1 Preprocessing and Feature Extraction

We first analyzed the basic statistics of the publishers, derived by grouping the entries in the click database by publisher. For each publisher, we computed the probability distribution (i.e., normalized frequency) of the number of clicks, number of visitors (identified by `numericip`), number of `referredurls`, and the ratio of the number of clicks over the number of visitors. Figure 8 shows the four distributions in the train set respectively, grouped by the publisher's status. Interestingly, we can see from Figure 8(a) that the `Fraud` publishers have lower click probability than the `OK` publishers. This can be attributed to the fact

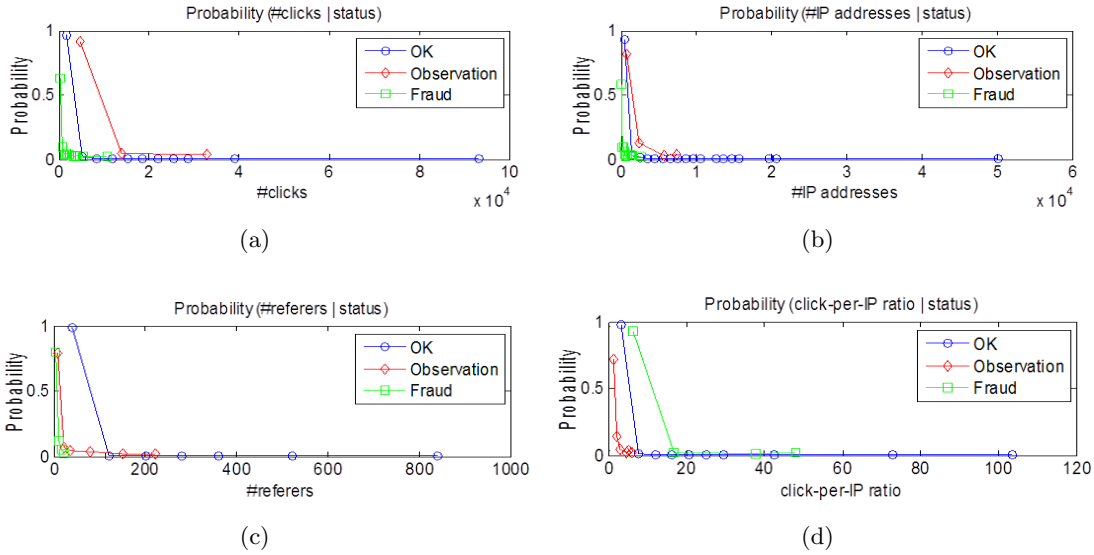


Figure 8: Distribution of the train data set: (a) Number of clicks, (b) Number of unique visitors (`numericip`), (c) Number of unique referrers (`referredurl`), and (d) Click per visitor ratio (number of clicks divided by number of unique visitors).

that BuzzCity blocks the traffic of a publisher as soon as its system deems the publisher as fraudulent. Figure 8(b) shows a similar observation for the distribution of the visitors.

For referrer (`referredurl`) distribution, we found `Fraud` publishers to be quite different from `OK` ones, and the former have low probability similar to the `Observation` publishers. Further investigation revealed that many `Fraud` publishers have missing/unknown `referredurl` fields. Hence, features derived from `referredurl` can be good indicators for fraudulent acts. Lastly, the distribution of the click per visitor ratio in Figure 8(c) shows that `Fraud` publishers have higher ratio than the other groups, suggesting that the former focus on more efficient use of resources (IP address, in this case) to inflate the click traffic. This motivates us to investigate other ratio-based features (e.g., click per referredurl ratio, click per deviceua ratio, click per country ratio, etc). Note that, if the denominator of the ratio (e.g., number of visitors) is zero, the ratio value will be set to zero.

We extracted several basic ratio features from the click database, as listed in Table 25(a). Since each publisher is associated with only one of the 10 channel categories (cf. Table 3), we also derived 10 Boolean features where only one feature can be set to 1 (true). From the publisher database, on the other hand, we computed two Boolean features: whether a publisher has a bank account (nonempty bank account), and whether (s)he has an address (nonempty address). Altogether, we have $5 + 10 + 2 = 17$ basic features. With these basic features alone, however, we found the detection unsatisfactory. To improve the results, we conducted fine-grained analysis on the spatiotemporal aspects of the publishers' click traffic, leading to two new types of features: *spatial* and *time series*. Table 25(b)-(c) respectively list the spatial and time series features used in this work.

Feature type	#features
Click per visitor ratio	1
Click per ad ratio	1
Click per deviceua ratio	1
Click per country ratio	1
Click per referredurl ratio	1
Category (binary)	10
Nonempty account (binary)	1
Nonempty address (binary)	1

(a)

Feature type	#features
Click fraction from top 20 countries	20
Click fraction from missing country	1
Click fraction from other countries	1
Click fraction from missing referredurl	1
Click fraction from missing deviceua	1

(b)

Feature type	#features
Click series	$6 \times 3 = 18$
Visitor series	$6 \times 3 = 18$
Ad series	$6 \times 3 = 18$
Deviceua series	$6 \times 3 = 18$
Country series	$6 \times 3 = 18$
Referredurl series	$6 \times 3 = 18$
Click per visitor ratio series	$6 \times 3 = 18$
Click per ad ratio series	$6 \times 3 = 18$
Click per deviceua ratio series	$6 \times 3 = 18$
Click per country ratio series	$6 \times 3 = 18$
Click per referredurl ratio series	$6 \times 3 = 18$
Gap interval series	$6 \times 3 = 18$

(c)

Table 25: List of features extracted from the BuzzCity's databases: (a) Basic features, (b) Spatial features, and (c) Time series features.

For the *spatial* features, we computed for each publisher the fraction of clicks coming from different `usercountry`, `referredurl` and `deviceua`. For the `usercountry` case, we considered the following top 20 countries in terms of their total number of clicks: {`bd` (Bangladesh), `br` (Brazil), `et` (Ethiopia), `gh` (Ghana), `id` (Indonesia), `in` (India), `ir` (Iran), `ke` (Kenya), `mx` (Mexico), `my` (Malaysia), `ng` (Nigeria), `pk` (Pakistan), `ru` (Russia), `sa` (Saudi Arabia), `th` (Thailand), `tr` (Turkey), `uk` (United Kingdom), `us` (United States), `vn` (Vietnam), and `za` (South Africa). We further computed the click fraction from missing/unknown country, and collate the click fraction from the remaining countries. Finally, we calculated the click fractions from missing `referredurl` and `deviceua`, which according to our preliminary studies correlate with the probability of fraud.

For the *time series* features, we broke down the 3-day span of each (train, validation, or test) data set into windows of 1 minute long, and tracked several values of interest (e.g., number of clicks in each minute, number of visitors per minute, duration between etc). This resulted in a time series vector of length 4,320 (i.e., 3 days = 4.320 minutes) for each value type. We also experimented with longer time interval (e.g., 1 hour and 1 day), but the results were worse than the 1 minute interval. Next, we computed several statistical features aggregating the time series values over the 3-day period. That is, we generated 6 statistical features for each time series: *nonzero count*, *mean*, *maximum*, *sum*, *sum of square*, and *standard deviation*. Note that, as most publishers have sparse time series, we filtered out all zero values prior to computing the statistical features. For example, given the series

value $v(t)$ at time window $t = \{1, 2, \ldots, 4320\}$, the mean of the series is $\overline{v} = \frac{1}{N} \sum_{v(t) \neq 0} v(t)$, where $N = |\{v(t)|v(t) \neq 0\}|$ is the number of nonzero entries in the series. Our preliminary studies showed that such approach led to better results than including zero values.

To capture trending patterns, we also derived the same set of statistical features for the *positive* and *negative gradients* of the series. Given a count/ratio value $v(t)$ at time t, the positive gradient $d^+(t)$ and negative gradient $d^-(t)$ are respectively computed as

$$d^+(t) = v(t+1) - v(t), \text{if } v(t) < v(t+1),$$
$$d^-(t) = v(t) - v(t+1), \text{if } v(t) > v(t+1)$$

where $t = \{1, 2, \ldots, 4319\}$. As before, we ruled out all zero values when constructing the gradient vectors $d^+(t)$ and $d^-(t)$, and in turn when computing the 6 statistical features summarizing the gradients. Thus, for each time series type, we have $6 \times 3 = 18$ features.

We also included a special time series called gap interval series. The gap interval refers the gap between the timestamps of two consecutive clicks. So if a publisher receives C clicks, then we have a gap interval series of length $C - 1$ (in contrast to the other time series that consists of 1-minute windows). We summarizes this series using the same $6 \times 3 = 18$ feature set. In sum, each publisher has 12 types of time series (as per Table 25) and hence a total of $18 \times 12 = 216$ time series features.

Adding all the basic, spatial, and time series features, we have a total of $17 + 216 + 24 = 257$ features. Not all these features were useful though, and feature elimination steps shall be carried out to improve the classification results. This is discussed in Section 7.2.2. Finally, after generating all the features, we normalized the feature values to be within $[0, 1]$.

7.2 Method

This section describes our proposed approach to tackling fraud detection task. We first describe the classification models we used, followed by refinement of the models via a feature elimination phase.

7.2.1 Classification Algorithms

We considered various single and ensemble-typed classification algorithms for the fraud detection task. As our single classifiers, we employed the following popular algorithms:

- *Logistic regression*: A popular classification method which extends linear regression analysis to model the relationship between a set of predictive variables and a binary outcome variable. It produces an outcome probability between 0 and 1. In this work, we employed the L2-regularized logistic regression implemented in the LIBLINEAR framework (Fan et al., 2008). To cope with the imbalanced class distribution, we adjusted the class weights to be inversely proportional to class frequencies.

- *Support vector machine* (SVM): A state-of-the-art classification method that aims at maximizing the margin of separation between data points from different classes. Intuitively, larger margin implies lower generalization error. We employed the SVM implementation in the LIBSVM framework (Chang and Lin, 2011), with different kernel functions including *linear*, *polynomial*, and *radial basis* kernels. As with logistic regression, we defined class weights to be inversely proportional to class frequencies.

- *k-nearest neighbors* (k-NN): A type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data (Cover, 1967). Classification is computed from a simple majority vote of the nearest neighbors of each point; a query point is assigned the class that has the most representatives within the nearest neighbors of the point.

On the other hand, we employed several decision tree-based ensemble classifiers, which have been widely used in data mining competitions:

- *Random forest*: Each tree in the forest is built from a sample drawn with replacement (i.e., bootstrap sample) from the train set (Breiman, 2001). Moreover, when splitting a node during the tree construction, the split chosen is not the best split among all features, but rather the best among a random subset of the features. In contrast to the original work (Breiman, 2001), which lets each classifier vote for a single class, we combined the tree classifiers by averaging their probabilistic predictions.

- *Gradient tree boosting* (GTB): Generalization of boosting to arbitrary differentiable loss functions (Friedman, 2000). GTB is an accurate off-the-shelf classification method that builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage, regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function.

- *Extremely randomized trees* (Extra tree): In this tree-based ensemble approach, randomness goes one step further in the way splits are computed. Similar to random forest, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule (Geurts et al., 2006). To arrive at the final decision, we again combined the tree classifiers by averaging their probabilistic predictions.

7.2.2 Feature Elimination

As mentioned, the complete 257 features are not all useful, and further improvement can be attained by removing inconsequential or noisy features. To this end, we devised a simple wrapper-based feature selection approach which we refer to as *backward feature elimination*. Algorithm 1 outlines the approach, which is inspired by the method described in Guyon and Elisseeff (2003). The algorithm requires as input the set of features and their predefined ranking or relative importance. The feature importance can be computed in several ways. In tree-based ensemble methods (e.g., random forest and GTB), features at the top of the tree are used to contribute to the final prediction decision of a larger fraction of the input samples. The expected fraction of the samples they contribute to can thus be used as an estimate of the relative importance of the features. In logistic regression or linear SVM, the feature importance can be estimated from the absolute values of its weight coefficients. For other methods that do not naturally give feature ranking (e.g., nonlinear SVM or k-NN), we estimated the feature importance using the F-value of the analysis of variance (ANOVA) test (Box, 1953). A higher the F-value implies a higher feature importance.

Algorithm 1 Backward feature elimination.

Require: Features set $\mathbf{F} = \{f_i\}$ and their ranks r_i, minimum number of features m, train data $\mathbf{T}$ and validation data $\mathbf{V}$

1: $\mathbf{R} \leftarrow \{\}$ and $\mathbf{F}_{best} \leftarrow \mathbf{F}$ // Initialization
2: $M \leftarrow$ train($\mathbf{F}$, $\mathbf{T}$) // Train M using features $\mathbf{F}$ from $\mathbf{T}$
3: $AP \leftarrow$ evaluate(M, $\mathbf{F}$, $\mathbf{V}$) // Evaluate M using features $\mathbf{F}$ from $\mathbf{V}$
4: $AP_{best} \leftarrow AP$
5: $\mathbf{F}_s \leftarrow$ sort($\mathbf{F}$) // Sort the features set in ascending order of r_i
6: **for** $i = 1$ **to** $|\mathbf{F}_s| - m$ **do**
7: $\quad \mathbf{R} \leftarrow \mathbf{R} \cup \{f_i\}$
8: $\quad M \leftarrow$ train($\mathbf{F}_s \setminus \mathbf{R}$, $\mathbf{T}$) // Train M using features $\mathbf{F}_s \setminus \mathbf{R}$ from $\mathbf{T}$
9: $\quad AP \leftarrow$ evaluate(M, $\mathbf{F}_s \setminus \mathbf{R}$, $\mathbf{V}$) // Evaluate M using features $\mathbf{F}_s \setminus \mathbf{R}$ from $\mathbf{V}$
10: $\quad$ **if** $AP_{best} \leq AP$ **then**
11: $\quad\quad AP_{best} \leftarrow AP$
12: $\quad\quad \mathbf{F}_{best} \leftarrow \mathbf{F}_s \setminus \mathbf{R}$
13: $\quad$ **end if**
14: **end for**
15: **return** $\mathbf{F}_{best}$

Our feature elimination method starts from the full feature set and eliminates features one by one from the lowest rank. An updated classifier is then generated from the remaining features. Our approach bears some similarities to the recursive feature elimination (Guyon et al., 2002), but instead of re-ranking the features after each elimination step, we still refer to the original (full) feature ranking. We also kept track of all performances obtained by each elimination, rather than stopping the elimination when a performance degradation is detected. The process is repeated until the minimum number of features m is reached. For SVM, logistic regression, and k-NN, we set $m = 1$. For tree-based ensemble methods, we set m as the maximum number of features $maxFeat$ (i.e., $m = maxFeat$), since the largest tree in the ensemble most likely has $maxFeat$ features.

7.3 Results and Discussion

We present hereafter the experimental results obtained by the approach described in Section 7.2. We first discuss the detection performance obtained using the full 257 feature set, followed by identification of the most important features based on our best models. Finally, we present improved detection results using our simplified models after feature elimination.

7.3.1 Performance

Table 26 shows the prediction results of all classifiers using the complete features listed in Figure 25. The best configuration for each classifier was determined using *grid search* on the corresponding (meta) parameter space. For logistic regression and SVM, we varied the penalty parameter C within the range $\{2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, \ldots, 2^{15}\}$ and then selected the parameter that gave the best AP score on the validation set. We also experimented with different kernels for the SVM, including linear, polynomial (with degree 2), and radial

Type	Method	Best parameter	Average precision	
			Validation	Test
Single	Logistic regression	$C = 512$	41.20%	29.65%
	SVM (Linear)	$C = 32$	29.74%	23.73%
	SVM (Polynomial)	$C = 8192$, degree $= 2$	29.33%	20.02%
	SVM (Radial basis)	$C = 128$	30.37%	23.63%
	k-NN	$k = 5$, weight = "distance"	28.78%	33.46%
Ensemble	Random forest	$ntrees = 100, maxFeat = 6$	57.53%	51.44%
	GTB	$ntrees = 900, maxFeat = 28, depth = 5$	48.78%	49.25%
	Extra trees	$ntrees = 200, maxFeat = 28$	55.36%	54.04%

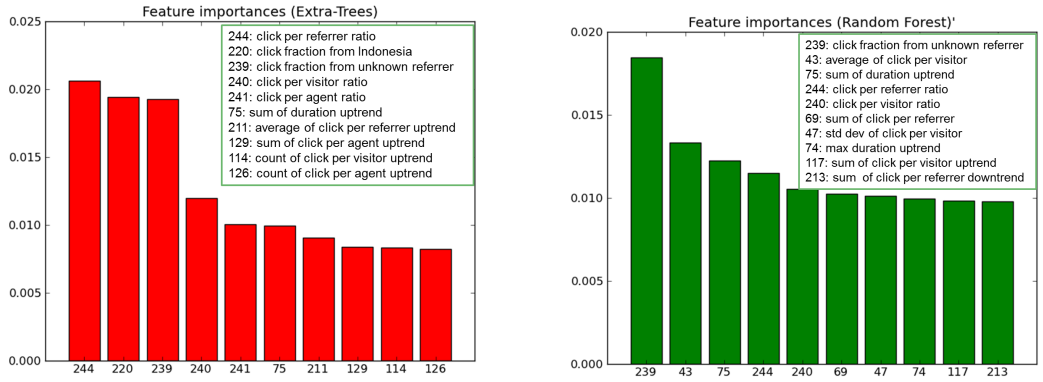
Table 26: Performances of various classifiers for the validation and test sets. For SVM and logistic regression, C is the penalty parameter controlling the tradeoff between training errors and margin maximization. For the ensemble methods, $ntrees$ is the number of tree models in the ensemble, $maxFeat$ is the maximum number of features allowed for each tree, and $depth$ is the maximum depth of a tree.

basis kernels. As for k-NN, we took the parameter k from the range $\{1, 3, 5, ..., 15\}$. We also considered "uniform" and "distance" weight functions for the k-NN predictions, giving equal weight and weight proportional to the inverse of distance to the neighborhood points, respectively. Out of these single classifiers, k-NN with 'distance' weight yielded the best score on the test set, suggesting that the decision boundary for the `Fraud` class is complex. For SVM, employing nonlinear kernel in place of linear kernel did not improve the result.

We further experimented with the tree ensemble methods (i.e., random forest, GTB, and Extra trees). For each method, we selected the best combination of $ntree$, $maxFeat$, and $depth$ from the range $\{100, 200, 300, \ldots, 1000\}$, $\{2, 4, 6, \ldots, 30\}$, and $\{1, 2, 3, 4, 5\}$, respectively. Among the three methods, we found that the extremely randomized trees yielded the best AP score int the test set, followed by the random forest which performed best on the validation set. In comparison to the single classifiers, we can see that the ensemble methods produced substantially higher AP scores. A plausible explanation is that ensemble methods try to exploit the local different behavior of the base learners to enhance the accuracy of the overall system. Also, using mixture of base models (instead of choosing just one) can help reduce the risk of (accidentally) selecting a poorly performing classifier, thereby reducing the overall system variance. Yet another key ingredient of effective ensemble system is the diversity of its constituent models. In the case of extremely randomized trees, the randomization of the split thresholds (in addition to the bootstrapping step in random forest) helps promote the diversity, leading not only to faster ensemble construction but also reduced variance of the overall system.

7.3.2 Feature Ranking

In Figure 9, we show the top 10 features found by our two best classifiers, Extra trees and random forest, whereby feature importance was estimated from the expected fraction of the samples the tree components contribute to (cf. Section 7.2.2). These features revealed several interesting observations. For instance, the top three features of the extremely randomized trees suggest that `Fraud` publishers tend to have missing/unknown `referredurl`,



(a) (b)

Figure 9: Feature ranking obtained by: (a) Extra trees, and (b) Random forest.

		Full features		Reduced features		
Type	Method	Validation AP	Test AP	#features	Validation AP	Test AP
Single	Logistic regression	41.20%	29.65%	46 / 257	46.02%	31.18%
	SVM (Linear)	30.45%	21.89%	38 / 257	36.75%	26.91%
	SVM (Polynomial)	22.69%	16.72%	256 / 257	28.42%	20.23%
	SVM (Radial basis)	34.32%	23.38%	255 / 257	39.10%	23.66%
	k-NN	28.78%	33.46%	257 / 257	28.78%	33.46%
Ensemble	Random forest	57.53%	51.44%	59 / 257	58.84%	52.17%
	GTB	48.78%	49.25%	235 / 257	58.33%	49.90%
	Extra trees	55.36%	54.04%	118 / 257	57.79%	55.64%

Table 27: Performances of single and ensemble classifiers after feature elimination.

which would be captured by the high click per referredurl ratio and high click fraction from unknown `referredurl`. Moreover, high click traffic from high-risk countries such as Indonesia can indicate fraudulent behavior (similar to the findings from the first competition winner). We can also observe that the ratio features and their related time series features, particularly the sum, count and uptrend (positive gradient) features, may be indicative of fraudulent behaviors. Similar observations were found by random forest, although the click fraction from high-risk countries were not deemed as important. This, in turn, may explain its inferior performance with respect to Extra trees.

7.3.3 Model Simplification

Using the feature elimination procedure outlined in Algorithm 1, we can remove inconsequential features and further improve the detection performance. Table 27 shows the consolidated results of different classifiers after feature elimination. In general, feature elimination improved both performances on the validation and test sets, while simplifying the classification models. We observed a large portion of features being removed by the logistic regression, linear SVM, random forest, and Extra trees, leading to a fair amount of improvements. By contrast, only small improvements were observed in GTB and nonlinear SVMs, which can be attributed to the marginal feature removals.

8. Conclusion

The results of the FDMA 2012 competition exceeded our expectations in several ways. First, we had high level of participation, although this is our first time organizing such a competition. Second, the participants turned in good results quickly, and the performances continually improved toward the end of the competition, showing the interesting potentials of various feature engineering and data mining methods. We conclude this paper by summarizing the solutions proposed by the winning teams, and then providing several important lessons we can learn from the competition results.

8.1 Methods Employed

We briefly comment on the methods commonly used by the winning teams as well as the remaining participants, as follows:

- **Preprocessing**: From the competition, we can see that most participants focused on time-series features generated through analyzing the click traffic of a publisher at multiple time resolutions (windows) and taking the statistics across time. Only a few participants used spatial features by grouping the click traffic based on `country`, `referredurl`, `channel`, etc. Simple normalization was also often used to improve the performance of the classifiers, such as normalization to $[0, 1]$. Feature transformation methods (e.g., principal component analysis) were rarely used and reported not to bring performance improvements.

- **Feature selection**: Feature selection approaches broadly fall into two types: *filter* and *wrapper* methods (Guyon and Elisseeff, 2003). Filter methods include feature selection algorithms that are independent of any predictors, filtering out features that have little chance to be useful in data analysis. Filters are usually less computationally intensive than wrappers, but they produce a feature set which is not tuned to a specific type of predictive model. On the other hand, algorithms of the wrapper type are wrapped around predictors, providing them subsets of features and receiving their feedback (usually accuracy). These wrapper approaches are aimed at improving results of the specific predictors they work with. Some participants used feature selection methods and reported that, in general, the wrapper methods performed better than the filter methods.

- **Classification algorithm**: In this competition, ensembles of decision trees were the most widely used approach, providing fairly fast learning and well suited to highly-skewed class distribution, noisy nonlinear patterns, and mixed variable types. More specifically, this success can be attributed to several factors. First, there is a lack of data to properly represent the true distribution, in which case the learning algorithm can find many different hypotheses that all give the same accuracy on the train data. By constructing an ensemble out of these accurate classifiers, the algorithm can "average" their votes and reduce the risk of choosing the wrong classifier. Second, many learning algorithms work by some form of local search that may get stuck in local optima. An ensemble built by running the local search from many different initial conditions may provide better approximation to the true, unknown function than

any of the individual classifiers. Lastly, for many data sets, the true function cannot be captured by a single hypothesis. By forming weighted sums of hypotheses in an ensemble, we can expand the space of representable functions.

All in all, ensemble methods combined with wrapper methods proved to be an effective approach for fraud detection. In practice, however, single models would still be preferred, owing to interpretability and tractability reasons. Future endeavors in this enterprise include devising a novel class of single classification algorithms, capable of matching the performances obtained by the top ranking participants.

8.2 What We Have Learned

The results from the FDMA 2012 Competition offer important insights on fraudulent behavior in online advertising. Below we summarize the key findings of the winning teams:

- **First winner's entry**: From a temporal standpoint, detecting large and/or duplicate clicks and distinguishing between morning and night traffics are important indicators for fraudulent acts. In spatial context, it was shown that high click fractions from top 10 high-risk countries provide strong signals for click fraud. For model selection, it is also important to have a model that balances between accuracy and overfitting.

- **Second winner's entry**: Fraudulent partners often try to act rationally by mimicking legitimate ones. Large variance on `deviceua` suggests that fraudulent partners use many agents to act rationally. An ensemble model that averages the predictions of different algorithms helps to gracefully deal with highly skewed class distribution, and can lead to better performance than that of the individual methods.

- **Third winner's entry**: Analyzing time series at different granularity levels (e.g., sec, min, hour, day) is important, and simple statistics (e.g., average, count) worked well in detecting click fraud. Backward feature elimination can be used to derive the most important fraud indicators. Linear blending of different models were found to give the best performance and improve the performance of the individual models.

- **Runner-up's entry**: Many consecutive clicks from the same IP address in a short-time interval are considered suspicious. The ensemble of random forests was found to be an effective fraud detection method. The problem of concept drift was observed in the competition data, and the "ground-truth" labels in the data may be inaccurate and biased toward BuzzCity's internal detection procedure.

- **Organizer's entry**: Fine-grained analysis of time series at short interval (1 minute) is crucial for deriving informative features for fraudulent publishers. The best model showed that `referredurl`-based features, for example, click per referredurl ratio and fraction of missing/unknown `referredurl`, provide informative indicators for fraud. Also, the high fraction of clicks from high-risk countries may be used as a signal for click fraud. Lastly, the combination of tree-based ensemble classifiers and backward feature elimination leads to a promising approach to tackle highly-imbalanced data.

8.3 Research Outlook

Despite the encouraging results and practical usability of our solutions to fraud detection task, there remains a considerable need for further work on this important topic. An obvious room for improvement is how the current methods can be used to tackle more sophisticated types of click fraud. For instance, fraudsters can work together as a group, allowing them to not only gain more with less (shared) resources, but also reduce the risk of getting detected. The solutions presented in this paper are currently unable to catch such coalition attacks. Several works have been dedicated to identify coalition fraud in online advertising (Metwally et al., 2007; Kim et al., 2011). However, these methods have so far been focused on investigating network topology structure (e.g., a bipartite graph of publishers and site visitors), without probing into the detailed spatial and temporal characteristics of either the individuals or group of individuals involved. Future research is needed to augment extrinsic network/group features with fine-grained analysis of spatio-temporal features.

A related research question concerns the adaptability of the current methods in the face of rapidly-evolving fraudulent behavior and strategies. For instance, several solutions presented in this paper use `deviceua` (user agent) as a feature, which may be easily exploited by malicious parties once it is known. Although ensemble approaches and the use of multiple complementary features can address this issue to some extent, there is a need for more general and robust learning methodologies. A plausible approach is to employ *online learning* (Shalev-Shwartz, 2012), which helps deal with concept drift pertaining to the behavioral changes of fraudulent parties, without having to retrain the prediction model from scratch. Another interesting direction to address the issue is to develop a *transfer learning* capability (Pan and Yang, 2010), using information from (multiple) auxiliary domains. Such approach is useful to deal with future data having different feature space and/or different distribution, while minimizing the model re-calibration efforts.

Acknowledgments

We are grateful to BuzzCity Pte. Ltd. for the competition data, prizes, and fruitful discussions. Special thanks go to Clifford Chew and Elvin Tan of BuzzCity, who contributed their inputs on the data and related domain knowledge. We also thank the organizers of the Asian Conference on Machine Learning (ACML) 2012 for hosting the FDMA 2012 Workshop. This work is supported by the National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the Interactive Digital Media (IDM) Programme Office.

References

M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel. KNIME–The Konstanz information miner: Version 2.0 and beyond. *SIGKDD Explorations Newsletter*, 11(1):26–31, 2009.

G. E. P. Box. Non-normality and tests on variances. *Biometrika*, 30(3/4):318–335, 1953.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

C. Chambers. Is click fraud a ticking time bomb under Google? *Forbes Magazine*, 2012. URL `http://www.forbes.com/sites/investor/2012/06/18/is-click-fraud-a-ticking-time-bomb-under-google/`.

C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems Technology*, 2(3):27:1–27:27, 2011.

A. Chao and T. Shen. Nonparametric estimation of shannon's index of diversity when there are unseen species in sample. *Environmental and Ecological Statistics*, 10:429–443, 2003.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

C. Chen, A. Liaw, and L. Breiman. Using random forests to learn imbalanced data. Technical Report No. 666, Department of Statistics, University of California, Berkeley, 2004.

W. Cohen. Fast effective rule induction. In *Proceedings of the International Conference on Machine Learning*, pages 115–123, Tahoe City, California, 1995.

T. Cover. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

V. Dave, S. Guha, and Y. Zhang. Measuring and fingerprinting click-spam in ad networks. In *ACM SIGCOMM Computer Communication Review*, volume 42, pages 175–186, Helsinki, Finland, 2012.

P. Domingos. MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 155–164, 1999.

R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the European Conference on Computational Learning Theory*, pages 23–37, London, UK, 1995.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the International Conference on Machine Learning*, pages 148–156, Bari, Italy, 1996.

J. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

J. Gama. Functional trees. *Machine Learning*, 55(3):219–250, 2004.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63 (1):3–42, 2006.

I. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264, 1953.

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. *Maching Learning*, 46(1–3):389–422, 2002.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA data mining software: An update. *SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

G. Holmes, B. Pfahringer, R. Kirkby, E. Frank, and M. Hall. Multiclass alternating decision trees. In *Proceedings of the European Conference on Machine Learning*, pages 161–172, Helsinki, Finland, 2002.

D. Horvitz and D. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47:663–685, 1952.

B. Jansen. Click fraud. *IEEE Computer*, 40(7):85–86, 2007.

C. Kim, H. Miao, and K. Shim. CATCH: A detecting algorithm for coalition attacks of hit inflation in internet advertising. *Information Systems*, 36(8):1105 – 1123, 2011.

R. Kohavi. The power of decision tables. In *Proceedings of the European Conference on Machine Learning*, pages 174–189, Heraclion, Crete, Greece, 1995.

R. Kohavi. Scaling up the accuracy of naïve Bayes classifiers: A decision-tree hybrid. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 202–207, Portland, OR, 1996.

Kroll Advisory Solutions. Global fraud report, 2012. URL `http://www.krolladvisory.com/insights-reports/global-fraud-reports/`.

A. Metwally, D. Agrawal, and A. El Abbadi. DETECTIVES: Detecting coalition hit inflation attacks in advertising networks streams. In *Proceedings of the International Conference on World Wide Web*, pages 241–250, 2007.

R. E. Neapolitan. *Learning Bayesian networks*. Prentice-Hall, Inc., 2003.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

C. Phua, K. Smith-Miles, V. Lee, and R. Gayler. Resilient identity crime detection. *IEEE Transactions on Knowledge and Data Engineering*, 24(3):533–546, 2012.

J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1993.

G. Ridgeway. Generalized boosted models: A guide to the gbm package, 2007. URL `http://cran.open-source-solution.org/web/packages/gbm/vignettes/gbm.pdf`.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 586–591, San Francisco, CA, 1993.

J. J. Rodrguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

S. Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012. ISSN 1935-8237.

J. Su and H. Zhang. A fast decision tree learning algorithm. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 500–505, Boston, MA, 2006.

A. Tuzhilin. The lane's gifts v. Google report, 2006. URL `http://googleblog.blogspot.sg/pdf/Tuzhilin_Report.pdf`.

D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

M. Zhu. Recall, precision and average precision. Technical Report (Working Paper 2004-09), University of Waterloo, 2004.

Journal of Machine Learning Research 15 (2014) 141-145 Submitted 4/13; Revised 10/13; Published 1/14

EnsembleSVM: A Library for Ensemble Learning Using Support Vector Machines

Marc Claesen MARC.CLAESEN@ESAT.KULEUVEN.BE
KU Leuven, ESAT – STADIUS/iMinds Future Health
Kasteelpark Arenberg 10, box 2446
3001 Leuven, Belgium

Frank De Smet FRANK.DESMET@CM.BE
KU Leuven, Department of Public Health and Primary Care, Environment and Health
Kapucijnenvoer 35 blok d, box 7001
3000 Leuven, Belgium

Johan A.K. Suykens JOHAN.SUYKENS@ESAT.KULEUVEN.BE
Bart De Moor BART.DEMOOR@ESAT.KULEUVEN.BE
KU Leuven, ESAT – STADIUS/iMinds Future Health
Kasteelpark Arenberg 10, box 2446
3001 Leuven, Belgium

Editor: Geoff Holmes

Abstract

EnsembleSVM is a free software package containing efficient routines to perform ensemble learning with support vector machine (SVM) base models. It currently offers ensemble methods based on binary SVM models. Our implementation avoids duplicate storage and evaluation of support vectors which are shared between constituent models. Experimental results show that using ensemble approaches can drastically reduce training complexity while maintaining high predictive accuracy. The EnsembleSVM software package is freely available online at http://esat.kuleuven.be/stadius/ensemblesvm.

Keywords: classification, ensemble learning, support vector machine, bagging

1. Introduction

Data sets are becoming increasingly large. Machine learning practitioners are confronted with problems where the main computational constraint is the amount of time available. Problems become particularly challenging when the training sets no longer fit into memory. Accurately solving the dual problem for SVM training with nonlinear kernels requires a run time which is at least quadratic in the size of the training set n, thus training complexity is $\Omega(n^2)$ (Bottou and Lin, 2007; List and Simon, 2009).

EnsembleSVM employs a divide-and-conquer strategy by aggregating many SVM models, trained on small subsamples of the training set. Through subdivision, total training time decreases significantly, even though more models need to be trained. For example, training p classifiers on subsamples of size n/p, results in an approximate complexity of $\Omega(n^2/p)$. This reduction in complexity helps in dealing with large data sets and nonlinear kernels.

Ensembles of SVM models have been used in various applications (Wang et al., 2009; Linghu and Sun, 2010; Mordelet and Vert, 2011). Collobert et al. (2002) use ensembles for large scale learning and employ a neural network to aggregate base models. Valentini and Dietterich (2003) provide an implementation which allows base models to use different kernels. For efficiency reasons, we require base models to share a single kernel function.

While other implementations mainly focus on improving predictive performance, our framework primarily aims to (i) make nonlinear large-scale learning feasible through complexity reductions and (ii) enable fast prototyping of novel ensemble algorithms.

2. Software Description

The `EnsembleSVM` software is freely available online under a LGPL license. `EnsembleSVM` provides ensembles of instance-weighted SVMs, as defined in Equation (1). The default approach we offer is bagging, which is commonly used to improve the performance of unstable classifiers (Breiman, 1996). In bagging, base models are trained on bootstrap subsamples of the training set and their predictions are aggregated through majority voting.

Base model flexibility is maximized by using instance-weighted binary support vector machine classifiers, as defined in Equation (1). This formulation lets users define misclassification penalties per training instance $C_i,\ i = 1, \ldots, n$ and encompasses popular approaches such as C-SVC and class-weighted SVM (Cortes and Vapnik, 1995; Osuna et al., 1997).

$$\begin{aligned} \min_{\mathbf{w},\xi,\rho} \ & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} C_i\xi_i, & (1)\\ \text{subject to } & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + \rho) \geq 1 - \xi_i, & i = 1, \ldots, n,\\ & \xi_i \geq 0, & i = 1, \ldots, n. \end{aligned}$$

When aggregating SVM models, the base models often share support vectors (SVs). The `EnsembleSVM` software intelligently caches distinct SVs to ensure that they are only stored and used for kernel evaluations once. As a result, `EnsembleSVM` models are smaller and faster in prediction than ensemble implementations based on wrappers.

2.1 Implementation

`EnsembleSVM` has been implemented in `C++` and makes heavy use of the standard library. The main implementation focus is training speed. We use facilities provided by the `C++11` standard and thus require a moderately recent compiler, such as `gcc` ≥ 4.7 or `clang` ≥ 3.2. A portable Makefile system based on GNU autotools is used to build `EnsembleSVM`.

`EnsembleSVM` interfaces with `LIBSVM` to train base models (Chang and Lin, 2011). Our code must be linked to `LIBSVM` but does not depend on a specific version. This allows users to choose the desired version of the `LIBSVM` software in the back-end.

The `EnsembleSVM` programming framework is designed to facilitate prototyping of ensemble algorithms using SVM base models. We particularly provide extensive support to define novel aggregation schemes, should the available options be insufficient. Key components are extensively documented and on a broad overview is provided on our wiki.[1]

1. The `EnsembleSVM` development wiki is available at `https://github.com/claesenm/EnsembleSVM/wiki`.

The `EnsembleSVM` library was built with extensibility and user contributions in mind. Major API functions are well documented to lower the threshold for external development. The executable tools provided with `EnsembleSVM` are essentially wrappers for the library itself. The tools can be considered as use cases of the main API functions to help developers.

2.2 Tools

The main tools in this package are `esvm-train` and `esvm-predict`, used to train and predict with ensemble models. Both of these are pthread-parallelized. Additionally, the `merge-models` tool can be used to merge standard `LIBSVM` models into ensembles. Finally, `esvm-edit` provides facilities to modify the aggregation scheme used by an ensemble.

`EnsembleSVM` includes a variety of extra tools to facilitate basic operations such as stratified bootstrap sampling, cross-validation, replacing categorical features by dummy variables, splitting data sets and sparsifying standard data sets. We recommend retaining the original ratio of positives and negatives in the training set when subsampling.

3. Benchmark Results

To illustrate the potential of our software, `EnsembleSVM` 2.0 has been benchmarked with respect to `LIBSVM` 3.17. To keep the experiments simple, we use majority voting to aggregate predictions, even though more sophisticated methods are offered. For reference, we also list the best obtained accuracy with a linear model, trained using `LIBLINEAR` (Fan et al., 2008). Linear methods are common in large-scale learning due to their speed, but may result in significantly decreased accuracy. This is why scalable nonlinear methods are desirable.

We used two binary classification problems, namely the `covtype` and `ijcnn1` data sets.[2] Both data sets are balanced. Features were always scaled to $[0, 1]$. We have used C-SVC as SVM and base models ($\forall\ i : C_i = C$). Reported numbers are averages of 5 test runs to ensure reproducibility. We used the RBF kernel, defined by the kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2}$. Optimal parameter selection was done through cross-validation.

The `covtype` data set is a common classification benchmark featuring 54 dimensions (Blackard and Dean, 1999). We randomly sampled balanced training and test sets of $100,000$ and $40,000$ instances respectively and classified class 2 versus all others. The `ijcnn1` data set was used in a machine learning challenge during IJCNN 2001 (Prokhorov, 2001). It contains $35,000$ training instances in 22 dimensions.

Results in Table 1 show several interesting trends. Training `EnsembleSVM` models is orders of magnitude faster, because training SVMs on small subsets significantly reduces complexity. Subsampling induces smaller kernels per base model resulting in lower overall memory use. Due to our parallelized implementation, ensemble models were faster in prediction than `LIBSVM` models in both experiments despite having twice as many SVs.

The ensembles in these experiments are competitive with a traditional SVM even though we used simple majority voting. For `covtype`, ensemble accuracy is 3% lower than a single SVM and for `ijcnn1` the ensemble is marginally better (0.2%). Linear SVM falls far short in terms of accuracy for both experiments, but is trained much faster (< 2 seconds).

2. Both data sets are available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`.

data set	test set accuracy			no. of SVs		time (s)	
	LIBSVM	LIBLINEAR	ESVM	LIBSVM	ESVM	LIBSVM	ESVM
covtype	0.92	0.76	0.89	26516	50590	728	35
ijcnn1	0.98	0.92	0.98	3564	7026	9.5	0.3

Table 1: Summary of benchmark results per data set: test set accuracy, number of support vectors and training time. Accuracies are listed for a single LIBSVM model, LIBLINEAR model and an ensemble model.

We obtained good results with very basic aggregation. Collobert et al. (2002) illustrated that more sophisticated aggregation methods can improve the predictive performance of ensembles. Others have reported performance improvements over standard SVM for ensembles using majority voting (Valentini and Dietterich, 2003; Wang et al., 2009).

4. Conclusions

EnsembleSVM provides users with efficient tools to experiment with ensembles of SVMs. Experimental results show that training ensemble models is significantly faster than training standard LIBSVM models while maintaining competitive predictive accuracy.

Linear methods are frequently applied in large-scale learning, mainly due to their low training complexity. Linear methods are known to have competitive accuracy for high dimensional problems. As our benchmarks showed, the difference in accuracy may be large for low dimensional problems. As such, fast nonlinear methods remain desirable in large-scale learning, particularly for low dimensional tasks with many training instances. Our benchmarks illustrate the potential of the ensemble approaches offered by EnsembleSVM.

Ensemble performance may be improved by using more complex aggregation schemes. EnsembleSVM currently offers various aggregation schemes, both linear and nonlinear. Additionally, it facilitates fast prototyping of novel methods.

EnsembleSVM strives to provide high-quality, user-friendly tools and an intuitive programming framework for ensemble learning with SVM base models. The software will be kept up to date by incorporating promising new methods and ideas when they are presented in the literature. User requests and suggestions are welcome and appreciated.

Acknowledgments

Frank De Smet is a member of the medical management department of the National Alliance of Christian Mutualities. Acknowledged funding sources: Marc Claesen (IWT grant number 111065); Research Council KU Leuven: GOA MaNet, CoE SymBioSys; EU: ERC AdG A-DATADRIVE-B.

References

Jock A. Blackard and Denis J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24(3):131–151, December 1999.

Léon Bottou and Chih-Jen Lin. Support vector machine solvers. In Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors, *Large Scale Kernel Machines*, pages 301–320, Cambridge, MA, USA, 2007. MIT Press.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Ronan Collobert, Samy Bengio, and Yoshua Bengio. A parallel mixture of SVMs for very large scale problems. *Neural Computation*, 14(5):1105–1114, 2002.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, September 1995.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, June 2008.

Bin Linghu and Bing-Yu Sun. Constructing effective SVM ensembles for image classification. In *Knowledge Acquisition and Modeling (KAM), 2010 3rd International Symposium on*, pages 80–83, 2010.

Nikolas List and Hans Ulrich Simon. SVM-optimization and steepest-descent line search. In *Proceedings of the 22nd Annual Conference on Computational Learning Theory*, 2009.

Fantine Mordelet and Jean-Philippe P. Vert. ProDiGe: Prioritization Of Disease Genes with multitask machine learning from positive and unlabeled examples. *BMC bioinformatics*, 12(1):389+, 2011.

Edgar Osuna, Robert Freund, and Federico Girosi. Support Vector Machines: Training and Applications. Technical Report AIM-1602, 1997.

Danil Prokhorov. IJCNN 2001 neural network competition. *Slide Presentation in IJCNN'01*, 2001.

Giorgio Valentini and Thomas G. Dietterich. Low bias bagged support vector machines. In *International Conference on Machine Learning, ICML-2003*, pages 752–759. Morgan Kaufmann, 2003.

Shi-jin Wang, Avin Mathew, Yan Chen, Li-feng Xi, Lin Ma, and Jay Lee. Empirical analysis of support vector machine ensemble classifiers. *Expert Systems with Applications*, 36(3, Part 2):6466 – 6476, 2009.

Journal of Machine Learning Research 15 (2014) 147-191 Submitted 4/13; Revised 9/13; Published 1/14

A Junction Tree Framework for Undirected Graphical Model Selection

Divyanshu Vats DVATS@RICE.EDU
Department of Electrical and Computer Engineering
Rice University
Houston, TX 77005, USA

Robert D. Nowak NOWAK@ECE.WISC.EDU
Department of Electrical and Computer Engineering
University of Wisconsin–Madison
Madison, WI 53706, USA

Editor: Sebastian Nowozin

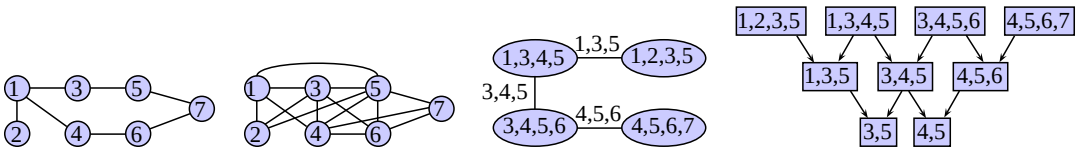
Abstract

An undirected graphical model is a joint probability distribution defined on an undirected graph G^*, where the vertices in the graph index a collection of random variables and the edges encode conditional independence relationships among random variables. The undirected graphical model selection (UGMS) problem is to estimate the graph G^* given observations drawn from the undirected graphical model. This paper proposes a framework for decomposing the UGMS problem into multiple subproblems over clusters and subsets of the separators in a junction tree. The junction tree is constructed using a graph that contains a superset of the edges in G^*. We highlight three main properties of using junction trees for UGMS. First, different regularization parameters or different UGMS algorithms can be used to learn different parts of the graph. This is possible since the subproblems we identify can be solved independently of each other. Second, under certain conditions, a junction tree based UGMS algorithm can produce consistent results with fewer observations than the usual requirements of existing algorithms. Third, both our theoretical and experimental results show that the junction tree framework does a significantly better job at finding the weakest edges in a graph than existing methods. This property is a consequence of both the first and second properties. Finally, we note that our framework is independent of the choice of the UGMS algorithm and can be used as a wrapper around standard UGMS algorithms for more accurate graph estimation.

Keywords: Graphical models, Markov random fields, junction trees, model selection, graphical model selection, high-dimensional statistics, graph decomposition

1. Introduction

An undirected graphical model is a joint probability distribution P_X of a random vector X defined on an undirected graph G^*. The graph G^* consists of a set of vertices $V = \{1, \ldots, p\}$ and a set of edges $E(G^*) \subseteq V \times V$. The vertices index the p random variables in X and the edges $E(G^*)$ characterize conditional independence relationships among the random variables in X (Lauritzen, 1996). We study undirected graphical models (also known as Markov random fields) so that the graph G^* is undirected, that is, if an edge $(i, j) \in E(G^*)$,



(a) Graph G^* (b) Graph H (c) Junction tree (d) Region graph

Figure 1: Our framework for estimating the graph in (a) using (b) computes the junction tree in (c) and uses a region graph representation in (d) of the junction tree to decompose the UGMS problem into multiple subproblems.

then $(j, i) \in E(G^*)$. The undirected graphical model selection (UGMS) problem is to estimate G^* given n observations $\mathfrak{X}^n = \left(X^{(1)}, \ldots, X^{(n)}\right)$ drawn from P_X. This problem is of interest in many areas including biological data analysis, financial analysis, and social network analysis; see Koller and Friedman (2009) for some more examples.

> **This paper studies the following problem:** *Given the observations $\mathfrak{X}^n$ drawn from P_X and a graph H that contains all the true edges $E(G^*)$, and possibly some extra edges, estimate the graph G^*.*

A natural question to ask is how can the graph H be selected in the first place? One way of doing so is to use screening algorithms, such as in Fan and Lv (2008) or in Vats (to appear), to eliminate edges that are clearly non-existent in G^*. Another method can be to use partial prior information about X to remove unnecessary edges. For example, this could be based on (i) prior knowledge about statistical properties of genes when analyzing gene expressions, (ii) prior knowledge about companies when analyzing stock returns, or (iii) demographic information when modeling social networks. Yet another method can be to use clever model selection algorithms that estimate more edges than desired. Assuming an initial graph H has been computed, our main contribution in this paper is to show how a junction tree representation of H can be used as a wrapper around UGMS algorithms for more accurate graph estimation.

1.1 Overview of the Junction Tree Framework

A junction tree is a tree-structured representation of an arbitrary graph (Robertson and Seymour, 1986). The vertices in a junction tree are clusters of vertices from the original graph. An edge in a junction tree connects two clusters. Junction trees are used in many applications to reduce the computational complexity of solving graph related problems (Arnborg and Proskurowski, 1989). Figure 1(c) shows an example of a junction tree for the graph in Figure 1(b). Notice that each edge in the junction tree is labeled by the set of vertices common to both clusters connected by the edge. These set of vertices are referred to as a *separator*.

Let H be a graph that contains all the edges in G^*. We show that the UGMS problem can be decomposed into multiple subproblems over *clusters* and *subsets of the separators* in a junction tree representation of H. In particular, using the junction tree, we construct

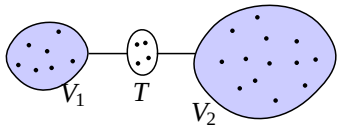


Figure 2: Structure of the graph used to analyze the junction tree framework for UGMS.

a region graph, which is a directed graph over clusters of vertices. An example of a region graph for the junction tree in Figure 1(c) is shown in Figure 1(d). The first two rows in the region graph are the clusters and separators of the junction tree, respectively. The rest of the rows contain subsets of the separators.[1] The multiple subproblems we identify correspond to estimating a subset of edges over each cluster in the region graph. For example, the subproblem over the cluster $\{1, 2, 3, 5\}$ in Figure 1(d) estimates the edges $(2, 3)$ and $(2, 5)$.

We solve the subproblems over the region graph in an iterative manner. First, all subproblems in the first row of the region graph are solved in parallel. Second, the region graph is updated taking into account the edges removed in the first step. We keep solving subproblems over rows in the region graph and update the region graph until all the edges in the graph H have been estimated.

As illustrated above, our framework depends on a junction tree representation of the graph H that contains a superset of the true edges. Given any graph, there may exist several junction tree representations. An optimal junction tree is a junction tree representation such that the maximum size of the cluster is as small as possible. Since we apply UGMS algorithms to the clusters of the junction tree, and the complexity of UGMS depends on the number of vertices in the graph, it is useful to apply our framework using optimal junction trees. Unfortunately, it is computationally intractable to find optimal junction trees (Arnborg et al., 1987). However, there exists several computationally efficient greedy heuristics that compute close to optimal junction trees (Kjaerulff, 1990; Berry et al., 2003). We use such heuristics to find junction trees when implementing our algorithms in practice.

1.2 Advantages of Using Junction Trees

We highlight three main advantages of the junction tree framework for UGMS.
Choosing Regularization Parameters and UGMS Algorithms: UGMS algorithms typically depend on a regularization parameter that controls the number of estimated edges. This regularization parameter is usually chosen using model selection algorithms such as cross-validation or stability selection. Since each subproblem we identify in the region graph is solved independently, different regularization parameters can be used to learn different parts of the graph. This has advantages when the true graph G^* has different characteristics in different parts of the graph. Further, since the subproblems are independent, different UGMS algorithms can be used to learn different parts of the graph. Our numerical simulations clearly show the advantages of this property.
Reduced Sample Complexity: One of the key results of our work is to show that in many cases, the junction tree framework is capable of consistently estimating a graph under weaker conditions than required by previously proposed methods. For example, we show that if

1. See Algorithm 1 for details on how to exactly construct the region graph.

G^* consists of two main components that are separated by a relatively small number of vertices (see Figure 2 for a general example), then, under certain conditions, the number of observations needed for consistent estimation scales like $\log(p_{\min})$, where $p_{\min}$ is the number of vertices in the smaller of the two components. In contrast, existing methods are known to be consistent if the observations scale like $\log p$, where p is the total number of vertices. If the smaller component were, for example, exponentially smaller than the larger component, then the junction tree framework is consistent with about $\log \log p$ observations. For generic problems, without structure that can be exploited by the junction tree framework, we recover the standard conditions for consistency.

Learning Weak Edges: A direct consequence of choosing different regularization parameters and the reduced sample complexity is that certain weak edges, not estimated using standard algorithms, may be estimated when using the junction tree framework. We show this theoretically and using numerical simulations on both synthetic and real world data.

1.3 Related Work

Several algorithms have been proposed in the literature for learning undirected graphical models. Some examples include References Spirtes and Glymour (1991), Kalisch and Bühlmann (2007), Banerjee et al. (2008), Friedman et al. (2008), Meinshausen and Bühlmann (2006), Anandkumar et al. (2012a) and Cai et al. (2011) for learning Gaussian graphical models, references Liu et al. (2009), Xue and Zou (2012), Liu et al. (2012a), Lafferty et al. (2012) and Liu et al. (2012b) for learning non-Gaussian graphical models, and references Bresler et al. (2008), Bromberg et al. (2009), Ravikumar et al. (2010), Netrapalli et al. (2010), Anandkumar et al. (2012b), Jalali et al. (2011), Johnson et al. (2012) and Yang et al. (2012) for learning discrete graphical models. Although all of the above algorithms can be modified to take into account prior knowledge about a graph H that contains all the true edges (see Appendix B for some examples), our junction tree framework is fundamentally different than the standard modification of these algorithms. The main difference is that the junction tree framework allows for using the *global Markov property* of undirected graphical models (see Definition 1) when learning graphs. This allows for improved graph estimation, as illustrated by both our theoretical and numerical results. We note that all of the above algorithms can be used in conjunction with the junction tree framework.

Junction trees have been used for performing *exact* probabilistic inference in graphical models (Lauritzen and Spiegelhalter, 1988). In particular, given a graphical model, and its junction tree representation, the computational complexity of exact inference is exponential in the size of the cluster in the junction tree with the most of number of vertices. This has motivated a line of research for learning *thin junction trees* so that the maximum size of the cluster in the estimated junction tree is small so that inference is computationally tractable (Chow and Liu, 1968; Bach and Jordan, 2001; Karger and Srebro, 2001; Chechetka and Guestrin, 2007; Kumar and Bach, 2013). We also make note of algorithms for learning decomposable graphical models where the graph structure is assumed to triangulated (Malvestuto, 1991; Giudici and Green, 1999). In general, the goal in the above algorithms is to learn a joint probability distribution that approximates a more complex probability distribution so that computations, such as inference, can be done in a tractable manner. On the other hand, this paper considers the problem of learning the *structure of*

the graph that best represents the conditional dependencies among the random variables under consideration.

There are two notable algorithms in the literature that use junction trees for learning graphical models. The first is an algorithm presented in Xie and Geng (2008) that uses junction trees to find the direction of edges for learning *directed* graphical models. Unfortunately, this algorithm cannot be used for UGMS. The second is an algorithm presented in Ma et al. (2008) for learning chain graphs, that are graphs with both directed and undirected edges. The algorithm in Ma et al. (2008) uses a junction tree representation to learn an undirected graph before orienting some of the edges to learn a chain graph. Our proposed algorithm, and subsequent analysis, differs from the work in Ma et al. (2008) in the following ways:

(i) Our algorithm identifies an ordering on the edges, which subsequently results in a lower sample complexity and the possibility of learning weak edges in a graph. The ordering on the edges is possible because of our novel region graph interpretation for learning graphical models. For example, when learning the graph in Figure 1(a) using Figure 1(b), the algorithm in Ma et al. (2008) learns the edge $(3, 5)$ by applying a UGMS algorithm to the vertices $\{1, 2, 3, 4, 5, 6\}$. In contrast, our proposed algorithm first estimates all edges in the second layer of the region graph in Figure 1(d), re-estimates the region graph, and then only applies a UGMS algorithm to $\{3, 4, 5\}$ to determine if the edge $(3, 4)$ belongs to the graph. In this way, our algorithm, in general, requires applying a UGMS algorithm to a smaller number of vertices when learning edges over separators in a junction tree representation.

(ii) Our algorithm for using junction trees for UGMS is independent of the choice of the UGMS algorithm, while the algorithm presented in Ma et al. (2008) uses conditional independence tests for UGMS.

(iii) Our algorithm, as discussed in (i), has the additional advantage of learning certain weak edges that may not be estimated when using standard UGMS algorithms. We theoretically quantify this property of our algorithm, while no such theory was presented in Ma et al. (2008).

Recent work has shown that solutions to the graphical lasso (gLasso) (Friedman et al., 2008) problem for UGMS over Gaussian graphical models can be computed, under certain conditions, by decomposing the problem over connected components of the graph computed by thresholding the empirical covariance matrix (Witten et al., 2011; Mazumder and Hastie, 2012). The methods in Witten et al. (2011) and Mazumder and Hastie (2012) are useful for computing solutions to gLasso for particular choices of the regularization parameter and not for accurately estimating graphs. Thus, when using gLasso for UGMS, we can use the methods in Witten et al. (2011) and Mazumder and Hastie (2012) to solve gLasso when performing model selection for choosing suitable regularization parameters. Finally, we note that recent work in Loh and Wainwright (2012) uses properties of junction trees to learn discrete graphical models. The algorithm in Loh and Wainwright (2012) is designed for learning discrete graphical models and our methods can be used to improve its performance.

1.4 Paper Organization

The rest of the paper is organized as follows:

- Section 2 reviews graphical models and formulates the undirected graphical model selection (UGMS) problem.
- Section 3 shows how junction trees can be represented as region graphs and outlines an algorithm for constructing a region graph from a junction tree.
- Section 4 shows how the region graphs can be used to apply a UGMS algorithm to the clusters and separators of a junction tree.
- Section 5 presents our main framework for using junction trees for UGMS. In particular, we show how the methods in Sections 3-4 can be used iteratively to estimate a graph.
- Section 6 reviews the PC-Algorithm, which we use to study the theoretical properties of the junction tree framework.
- Section 7 presents theoretical results on the sample complexity of learning graphical models using the junction tree framework. We also highlight advantages of using the junction tree framework as summarized in Section 1.2.
- Section 8 presents numerical simulations to highlight the advantages of using junction trees for UGMS in practice.
- Section 9 summarizes the paper and outlines some future work.

2. Preliminaries

In this section, we review some necessary background on graphs and graphical models that we use in this paper. Section 2.1 reviews some graph theoretic concepts. Section 2.2 reviews undirected graphical models. Section 2.3 formally defines the undirected graphical model selection (UGMS) problem. Section 2.4 reviews junction trees, which we use use as a tool for decomposing UGMS into multiple subproblems.

2.1 Graph Theoretic Concepts

A graph is a tuple $G = (V, E(G))$, where V is a set of vertices and $E(G) \subseteq V \times V$ are edges connecting vertices in V. For any graph H, we use the notation $E(H)$ to denote its edges. We only consider undirected graphs where if $(v_1, v_2) \in E(G)$, then $(v_2, v_1) \in E(G)$ for $v_1, v_2 \in V$. Some graph theoretic notations that we use in this paper are summarized as follows:

- Neighbor $ne_G(i)$: Set of nodes connected to i.
- Path $\{i, s_1, \ldots, s_d, j\}$: A sequence of nodes such that $(i, s_1), (s_d, j), (s_k, s_{k+1}) \in E$ for $k = 1, \ldots, d-1$.

- Separator S: A set of nodes such that all paths from i to j contain at least one node in S. The separator S is *minimal* if no proper subset of S separates i and j.
- Induced Subgraph $G[A] = (A, E(G[A]))$: A graph over the nodes A such that $E(G[A])$ contains the edges only involving the nodes in A.
- Complete graph K_A: A graph that contains all possible edges over the nodes A.

For two graphs $G_1 = (V_1, E(G_1))$ and $G_2 = (V_2, E(G_2))$, we define the following standard operations:

- Graph Union: $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$.
- Graph Difference: $G_1 \backslash G_2 = (V_1, E_1 \backslash E_2)$.

2.2 Undirected Graphical Models

Definition 1 (Undirected Graphical Model, Lauritzen, 1996) *An undirected graphical model is a probability distribution P_X defined on a graph $G^* = (V, E(G^*))$, where $V = \{1, \ldots, p\}$ indexes the random vector $X = (X_1, \ldots, X_p)$ and the edges $E(G^*)$ encode the following Markov property: for a set of nodes A, B, and S, if S separates A and B, then $X_A \perp\!\!\!\perp X_B | X_S$.*

The Markov property outlined above is referred to as the *global Markov property.* Undirected graphical models are also referred to as Markov random fields or Markov networks in the literature. When the joint probability distribution P_X is non-degenerate, that is, $P_X > 0$, the Markov property in Definition 1 are equivalent to the pairwise and local Markov properties:

- Pairwise Markov property: For all $(i, j) \notin E$, $X_i \perp\!\!\!\perp X_j | X_{V \setminus \{i,j\}}$.
- Local Markov property: For all $i \in V$, $X_i \perp\!\!\!\perp X_{V \setminus \{ne_G(i) \cup \{i\}\}} | X_{ne_G(i)}$.

In this paper, we always assume $P_X > 0$ and say P_X is *Markov* on G to reflect the Markov properties. Examples of conditional independence relations conveyed by a probability distribution defined on the graph in Figure 3(d) are $X_1 \perp\!\!\!\perp X_6 | \{X_2, X_4\}$ and $X_4 \perp\!\!\!\perp X_6 | \{X_2, X_5, X_8\}$.

2.3 Undirected Graphical Model Section (UGMS)

Definition 2 (UGMS) *The undirected graphical model selection (UGMS) problem is to estimate a graph G^* such that the joint probability distribution P_X is Markov on G^*, but not Markov on any subgraph of G^*.*

The last statement in Definition 2 is important, since, if P_X is Markov on G^*, then it is also Markov on any graph that contains G^*. For example, all probability distributions are Markov on the complete graph. Thus, the UGMS problem is to find the minimal graph that captures the Markov properties associated with a joint probability distribution. In the literature, this is also known as finding the minimal I-map.

Let Ψ be an abstract UGMS algorithm that takes as inputs a set of n i.i.d. observations $\mathfrak{X}^n = \{X^{(1)}, \ldots, X^{(n)}\}$ drawn from P_X and a regularization parameter λ_n. The output of Ψ is a graph $\widehat{G}_n$, where λ_n controls the number of edges estimated in $\widehat{G}_n$. Note the dependence of the regularization parameter on n. We assume Ψ is consistent, which is formalized in the following assumption.

Assumption 1 *There exists a λ_n for which $P(\widehat{G}_n = G^*) \to 1$ as $n \to \infty$, where $\widehat{G}_n = \Psi(\mathfrak{X}^n, \lambda_n)$.*

We give examples of Ψ in Appendix B. Assumption 1 also takes into account the high-dimensional case where p depends on n in such a way that $p, n \to \infty$.

2.4 Junction Trees

Junction trees (Robertson and Seymour, 1986) are used extensively for efficiently solving various graph related problems, see Arnborg and Proskurowski (1989) for some examples. Reference Lauritzen and Spiegelhalter (1988) shows how junction trees can be used for exact inference (computing marginal distribution given a joint distribution) over graphical models. We use junction trees as a tool for decomposing the UGMS problem into multiple subproblems.

Definition 3 (Junction tree) *For an undirected graph $G = (V, E(G))$, a junction tree $\mathcal{J} = (\mathcal{C}, E(\mathcal{J}))$ is a tree-structured graph over clusters of nodes in V such that*

(i) Each node in V is associated with at least one cluster in $\mathcal{C}$.

(ii) For every edge $(i, j) \in E(G)$, there exists a cluster $C_k \in \mathcal{C}$ such that $i, j \in C_k$.

(iii) $\mathcal{J}$ satisfies the running intersection property: For all clusters C_u, C_v, and C_w such that C_w separates C_u and C_v in the tree defined by $E(\mathcal{J})$, $C_u \cap C_v \subset C_w$.

The first property in Definition 3 says that all nodes must be mapped to at least one cluster of the junction tree. The second property states that each edge of the original graph must be contained within a cluster. The third property, known as the running intersection property, is the most important since it restricts the clusters and the trees that can be be formed. For example, consider the graph in Figure 3(a). By simply clustering the nodes over edges, as done in Figure 3(b), we can *not* get a valid junction tree (Wainwright, 2002). By making appropriate clusters of size three, we get a valid junction tree in Fig. 3(c). In other words, the running intersection property says that for two clusters with a common node, all the clusters on the path between the two clusters must contain that common node.

Proposition 4 (Robertson and Seymour, 1986) *Let $\mathcal{J} = (\mathcal{C}, E(\mathcal{J}))$ be a junction tree of the graph G. Let $S_{uv} = C_u \cap C_v$. For each $(C_u, C_v) \in \mathcal{E}$, we have the following properties:*

1. *$S_{uv} \neq \emptyset$.*
2. *S_{uv} separates $C_u \backslash S_{uv}$ and $C_v \backslash S_{uv}$.*

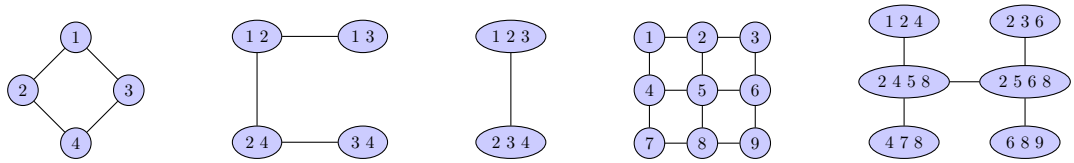


Figure 3: (a) An undirected graph, (b) Invalid junction tree since $\{1, 2\}$ separates $\{1, 3\}$ and $\{3, 4\}$,but $3 \notin \{1, 2\}$. (c) Valid junction tree for the graph in (a). (d) A grid graph. (e) Junction tree representation of (d).

The set of nodes S_{uv} on the edges are called the *separators* of the junction tree. Proposition 4 says that all clusters connected by an edge in the junction tree have at least one common node and the common nodes separate nodes in each cluster. For example, consider the junction tree in Figure 3(e) of the graph in Figure 3(d). We can infer that 1 and 5 are separated by 2 and 4. Similarly, we can also infer that 4 and 6 are separated by 2, 5, and 8. It is clear that if a graphical model is defined on the graph, then the separators can be used to easily define conditional independence relationships. For example, using Figure 3(e), we can conclude that $X_1 \perp\!\!\!\perp X_5$ given X_2 and X_4. As we will see in later Sections, Proposition 4 allow the decomposition of UGMS into multiple subproblems over clusters and subsets of the separators in a junction tree.

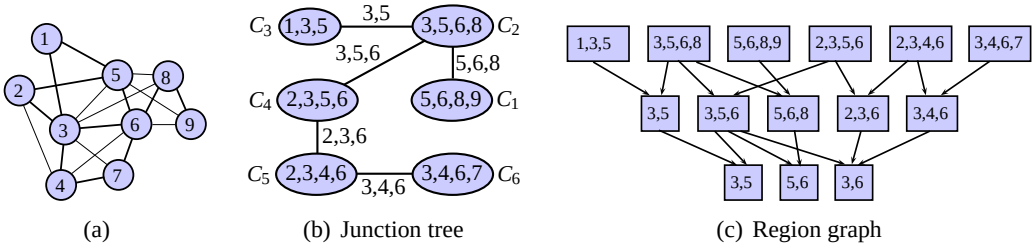
3. Overview of Region Graphs

In this section, we show how junction trees can be represented as region graphs. As we will see in Section 5, region graphs allow us to easily decompose the UGMS problem into multiple subproblems. There are many different types of region graphs and we refer the readers to Yedidia et al. (2005) for a comprehensive discussion about region graphs and how they are useful for characterizing graphical models. The region graph we present in this section differs slightly from the standard definition of region graphs. This is mainly because our goal is to estimate edges, while the standard region graphs defined in the literature are used for computations over graphical models.

A *region* is a collection of nodes, which in this paper can be the clusters of the junction tree, separators of the junction tree, or subsets of the separators. A region graph $\mathcal{G} = (\mathcal{R}, \vec{E}(\mathcal{G}))$ is a directed graph where the vertices are regions and the edges represent directed edges from one region to another. We use the notation $\vec{E}(\cdot)$ to emphasize that region graphs contain directed edges. A description of region graphs is given as follows:

- The set $\vec{E}(\mathcal{G})$ contains directed edges so that if $(R, S) \in \vec{E}(\mathcal{G})$, then there exists a directed edge from region R to region S.
- Whenever $R \longrightarrow S$, then $S \subseteq R$.

Algorithm 1 outlines an algorithm to construct region graphs given a junction tree representation of a graph H. We associate a label l with every region in $\mathcal{R}$ and group



(a) (b) Junction tree (c) Region graph

Figure 4: (a) An example of H. (b) A junction tree representation of H. (c) A region graph representation of (b) computed using Algorithm 1.

Algorithm 1: Constructing region graphs

Input: A junction tree $\mathcal{J} = (\mathcal{C}, E(\mathcal{J}))$ of a graph H.

Output: A region graph $\mathcal{G} = (\mathcal{R}, \vec{E}(\mathcal{G}))$.

1 $\mathcal{R}^1 = \mathcal{C}$, where $\mathcal{C}$ are the clusters of the junction tree $\mathcal{J}$.

2 Let $\mathcal{R}^2$ be all the separators of $\mathcal{J}$, that is, $\mathcal{R}^2 = \{S_{uv} = C_u \cap C_v : (C_u, C_v) \in E(\mathcal{J})\}$.

3 To construct $\mathcal{R}^3$, find all possible pairwise intersections of regions in $\mathcal{R}^2$. Add all intersecting regions with cardinality greater than one to $\mathcal{R}^3$.

4 Repeat previous step to construct $\mathcal{R}^4, \ldots, \mathcal{R}^L$ until there are no more intersecting regions of cardinality greater than one.

5 For $R \in \mathcal{R}^\ell$ and $S \in \mathcal{R}^{\ell+1}$, add the edge (R, S) to $\vec{E}(\mathcal{G})$ if $S \subseteq R$.

6 Let $\mathcal{R} = \{\mathcal{R}^1, \ldots, \mathcal{R}^L\}$.

regions with the same label to partition $\mathcal{R}$ into L groups $\mathcal{R}^1, \ldots, \mathcal{R}^L$. In Algorithm 1, we initialize $\mathcal{R}^1$ and $\mathcal{R}^2$ to be the clusters and separators of a junction tree $\mathcal{J}$, respectively, and then iteratively find $\mathcal{R}^3, \ldots, \mathcal{R}^L$ by computing all possible intersections of regions with the same label. The edges in $\vec{E}(\mathcal{G})$ are only drawn from a region in $\mathcal{R}^l$ to a region in $\mathcal{R}^{l+1}$. Figure 4(c) shows an example of a region graph computed using the junction tree in Figure 4(b).

Remark 5 Note that the construction of the region graph depends on the junction tree. Using methods in Vats and Moura (2012), we can always construct junction trees such that the region graph only has two sets of regions, namely the clusters of the junction tree and the separators of the junction tree. However, in this case, the size of the regions or clusters may be too large. This may not be desirable since the computational complexity of applying UGMS algorithms to region graphs, as shown in Section 5, depends on the size of the regions.

Remark 6 (Region graph vs. Junction tree) For every junction tree, Algorithm 1 outputs a unique region graph. The junction tree only characterizes the relationship between the clusters in a junction tree. A region graph extends the junction tree representation to characterize the relationships between the clusters as well as the separators. For example, in Figure 4(c), the region $\{5, 6\}$ is in the third row and is a subset of two separators of the

junction tree. Thus, the only difference between the region graph and the junction tree is the additional set of regions introduced in $\mathcal{R}^3, \ldots, \mathcal{R}^L$.

Remark 7 From the construction in Algorithm 1, $\mathcal{R}$ may have two or more regions that are the same but have different labels. For example, in Figure 4(c), the region $\{3,5\}$ is in both $\mathcal{R}^2$ and $\mathcal{R}^3$. We can avoid this situation by removing $\{3,5\}$ from $\mathcal{R}^2$ and adding an edge from the region $\{1,3,5\}$ in $\mathcal{R}^1$ to the region $\{3,5\}$ in $\mathcal{R}^3$. For notational simplicity and for the purpose of illustration, we allow for duplicate regions. This does not change the theory or the algorithms that we develop.

4. Applying UGMS to Region Graphs

Before presenting our framework for decomposing UGMS into multiple subproblems, we first show how UGMS algorithms can be applied to estimate a subset of edges in a region of a region graph. In particular, for a region graph $\mathcal{G} = (\mathcal{R}, \vec{E}(\mathcal{G}))$, *we want to identify a set of edges in the induced subgraph $H[R]$ that can be estimated by applying a UGMS algorithm to either R or a set of vertices that contains R.* With this goal in mind, define the children $ch(R)$ of a region R as follows:

$$\text{Children: } ch(R) = \left\{ S : (R,S) \in \vec{\mathcal{E}} \right\}. \tag{1}$$

We say R connects to S if $(R,S) \in \vec{E}(\mathcal{G})$. Thus, the children in (1) consist of all regions that R connects to. For example, in Figure 4(c),

$$ch(\{2,3,4,6\}) = \{\{2,3,6\},\{3,4,6\}\}.$$

If there exists a direct path from S to R, we say S is an *ancestor* of R. The set of all ancestors of R is denoted by $an(R)$. For example, in Figure 4(c),

$$\begin{aligned} an(\{5,6,8,9\}) &= \emptyset, \\ an(\{3,5,6\}) &= \{\{3,5,6,8\},\{2,3,5,6\}\}, \text{and} \\ an(\{3,6\}) &= \{\{3,5,6\},\{2,3,6\},\{3,4,6\},\{2,3,5,6\},\{2,3,4,6\},\{3,4,6,7\},\{3,5,6,8\}\}. \end{aligned}$$

The notation $\overline{R}$ takes the union of all regions in $an(R)$ and R so that

$$\overline{R} = \bigcup_{S \in \{an(R), R\}} S. \tag{2}$$

Thus, $\overline{R}$ *contains the union of all clusters in the junction tree that contain R.* An illustration of some of the notations defined on region graphs is shown in Figure 5. Using $ch(R)$, define the subgraph H'_R as[2]

$$H'_R = H[R] \backslash \left\{ \cup_{S \in ch(R)} K_S \right\}, \tag{3}$$

where $H[R]$ is the induced subgraph that contains all edges in H over the region R and K_S is the complete graph over S. In words, H'_R *is computed by removing all edges from $H[R]$ that are contained in another separator.* For example, in Figure 4(c), when $R = \{5,6,8\}$, $E(H'_R) = \{(5,8),(6,8)\}$. The subgraph H'_R is important since it identifies the edges that can be estimated when applying a UGMS algorithm to the set of vertices $\overline{R}$.

2. For graphs G_1 and G_2, $E(G_1 \backslash G_2) = E(G_1) \backslash E(G_2)$ and $E(G_1 \cup G_2) = E(G_1) \cup E(G_2)$.

Algorithm 2: UGMS over regions in a region graph

1: **Input:** Region graph $\mathcal{G} = (\mathcal{R}, \vec{E}(\mathcal{G}))$, a region R, observations $\mathfrak{X}^n$, and a UGMS algorithm Ψ.
2: Compute H'_R using (3) and $\overline{R}$ using (2).
3: Apply Ψ to $\mathfrak{X}^n_{\overline{R}}$ to estimate edges in H'_R. See Appendix B for examples.
4: **Return** the estimated edges $\widehat{E}_R$.

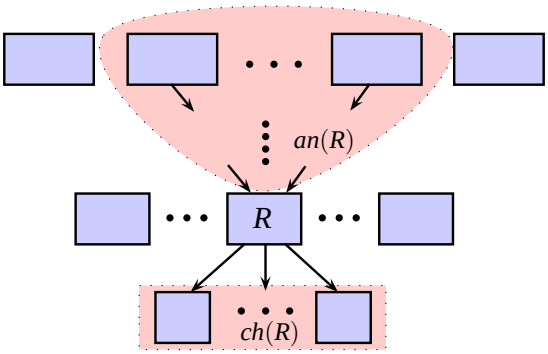


Figure 5: Notations defined on region graphs. The children $ch(R)$ are the set of regions that R connects to. The ancestors $an(R)$ are all the regions that have a directed path to the region R. The set $\overline{R}$ takes the union of all regions in $an(R)$ and R.

Proposition 8 *Suppose $E(G^*) \subseteq E(H)$. All edges in H'_R can be estimated by solving a UGMS problem over the vertices $\overline{R}$.*

Proof See Appendix C. ■

Proposition 8 says that all edges in H'_R can be estimated by applying a UGMS algorithm to the set of vertices $\overline{R}$. The intuition behind the result is that only those edges in the region R can be estimated whose Markov properties can be deduced using the vertices in $\overline{R}$. Moreover, the edges *not estimated* in $H[R]$ share an edge with another region that does not contain all the vertices in R. Algorithm 2 summarizes the steps involved in estimating the edges in H'_R using the UGMS algorithm Ψ defined in Section 2.3. Some examples on how to use Algorithm 2 to estimate some edges of the graph in Figure 4(a) using the region graph in Figure 4(c) are described as follows.

1. Let $R = \{1, 3, 5\}$. This region only connects to $\{3, 5\}$. This means that all edges, except the edge $(3, 5)$ in $H[R]$, can be estimated by applying Ψ to R.

2. Let $R = \{3, 5, 6\}$. The children of this region are $\{3, 5\}$, $\{5, 6\}$, and $\{3, 6\}$. This means that $H'_R = \emptyset$, that is, no edge over $H[R]$ can be estimated by applying Ψ to $\{3, 5, 6\}$.

Notation	Description
$G^* = (V, E(G^*))$	Unknown graph that we want to estimate.
H	Known graph such that $E(G^*) \subseteq E(H)$.
$\mathcal{G} = (\mathcal{R}, \vec{E}(\mathcal{G}))$	Region graph of H constructed using Algorithm 1.
$\mathcal{R} = (\mathcal{R}^1, \ldots, \mathcal{R}^L)$	Partitioning of the regions in $\mathcal{R}$ into L labels.
$\overline{R}$	The set of vertices used when applying Ψ to estimate edges over R. See (2) for definition.
H'_R	Edges in $H[R]$ that can be estimated using Algorithm 2. See (3) for definition.

Table 1: A summary of some notations.

3. Let $R = \{3, 4, 6\}$. This region only connects to $\{3, 6\}$. Thus, all edges except $(3, 6)$ can be estimated. The regions $\{2, 3, 4, 6\}$ and $\{3, 4, 6, 7\}$ connect to R, so Ψ needs to be applied to $\overline{R} = \{2, 3, 4, 6, 7\}$.

5. UGMS Using Junction Trees: A General Framework

In this section, we present the junction tree framework for UGMS using the results from Sections 3-4. Section 5.1 presents the junction tree framework. Section 5.2 discusses the computational complexity of the framework. Section 5.3 highlights the advantages of using junction trees for UGMS using some examples. We refer to Table 1 for a summary of all the notations that we use in this section.

5.1 Description of Framework

Recall that Algorithm 2 shows that to estimate a subset of edges in $H[R]$, where R is a region in the region graph $\mathcal{G}$, the UGMS algorithm Ψ in Assumption 1 needs to be applied to the set $\overline{R}$ defined in (2). Given this result, a straightforward approach to decomposing the UGMS problem is to apply Algorithm 2 to each region R and combine all the estimated edges. This will work since for any $R, S \in \mathcal{R}$ such that $R \neq S$, $E(H'_R) \cap E(H'_S) = \emptyset$. This means that each application of Algorithm 2 estimates a different set of edges in the graph. However, for some edges, this may require applying a UGMS algorithm to a large set of nodes. For example, in Figure 4(c), when applying Algorithm 2 to $R = \{3, 6\}$, the UGMS algorithm needs to be applied to $\overline{R} = \{2, 3, 4, 5, 6, 7, 8\}$, which is almost the full set of vertices. To reduce the problem size of the subproblems, we apply Algorithms 1 and 2 in an iterative manner as outlined in Algorithm 3.

Figure 6 shows a high level description of Algorithm 3. We first find a junction tree and then a region graph of the graph H using Algorithm 1. We then find the row in the region graph over which edges can be estimated and apply Algorithm 2 to each region in that row. We note that when estimating edges over a region, we use model selection algorithms to choose an appropriate regularization parameter to select the number of edges to estimate. Next, all estimated edges are added to $\widehat{G}$ and all edges that are estimated are removed from H. Thus, H now represents all the edges that are left to be estimated and $\widehat{G} \cup H$ contains

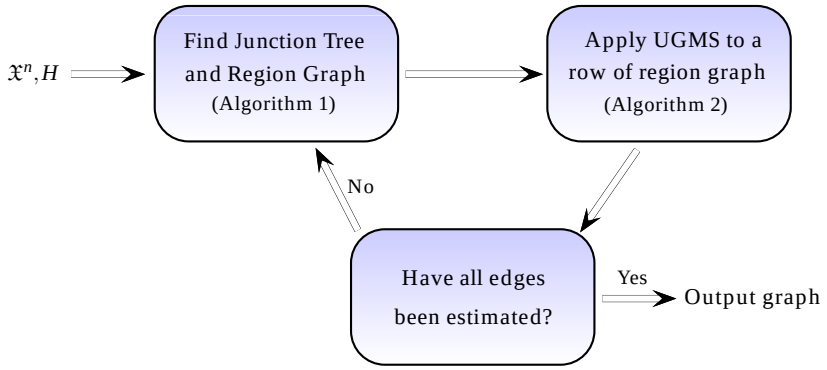


Figure 6: A high level overview of the junction tree framework for UGMS in Algorithm 3.

Algorithm 3: Junction Tree Framework for UGMS

See Table 1 for notations.

Step 1. Initialize $\widehat{G}$ so that $E(\widehat{G}) = \emptyset$ and find the region graph $\mathcal{G}$ of H.

Step 2. Find the smallest ℓ such that there exists a region $R \in \mathcal{R}^\ell$ such that $E(H'_R) \neq \emptyset$.

Step 3. Apply Algorithm 2 to each region in $\mathcal{R}^\ell$.

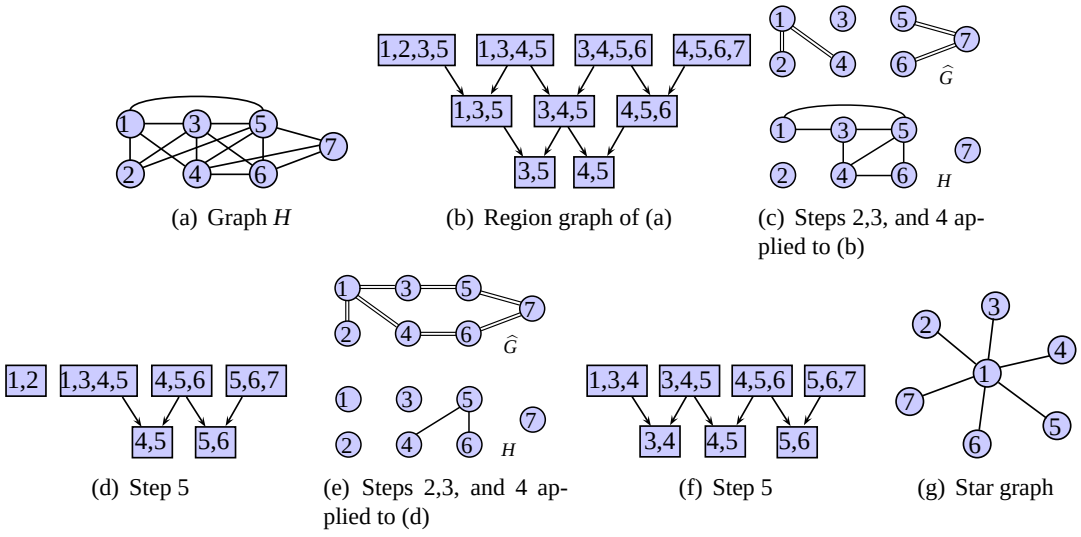
Step 4. Add all estimated edges to $\widehat{G}$ and *remove edges* from H that have been estimated. Now $H \cup \widehat{G}$ contains all the edges in G^*.

Step 5. Compute a new junction tree and region graph $\mathcal{G}$ using the graph $\widehat{G} \cup H$.

Step 6. If $E(H) = \emptyset$, stop the algorithm, else go to Step 2.

all the edges in G^*. We repeat the above steps on a new region graph computed using $\widehat{G} \cup H$ and stop the algorithm when H is an empty graph.

An example illustrating the junction tree framework is shown in Figure 7. The region graph in Figure 7(b) is constructed using the graph H in Figure 7(a). The true graph G^* we want to estimate is shown in Figure 1(a). The top and bottom in Figure 7(c) show the graphs $\widehat{G}$ and H, respectively, after estimating all the edges in $\mathcal{R}^1$ of Figure 7(b). The edges in $\widehat{G}$ are represented by double lines to distinguish them from the edges in H. Figure 7(d) shows the region graph of $\widehat{G} \cup H$. Figure 7(e) shows the updated $\widehat{G}$ and H where only the edges $(4, 5)$ and $(5, 6)$ are left to be estimated. This is done by applying Algorithm 2 to the regions in $\mathcal{R}^2$ of Figure 7(f). Notice that we did not include the region $\{1, 2\}$ in the last region graph since we know all edges in this region have already been estimated. In general, if $E(H[R]) = \emptyset$ for any region R, we can remove this region and thereby reduce the computational complexity of constructing region graphs.



(a) Graph H (b) Region graph of (a) (c) Steps 2,3, and 4 applied to (b)

(d) Step 5 (e) Steps 2,3, and 4 applied to (d) (f) Step 5 (g) Star graph

Figure 7: Example to illustrate the junction tree framework in Algorithm 3.

5.2 Computational Complexity

In this section, we discuss the computational complexity of the junction tree framework. It is difficult to write down a closed form expression since the computational complexity depends on the structure of the junction tree. Moreover, merging clusters in the junction tree can easily control the computations. With this in mind, the main aim in this section is to show that the complexity of the framework is roughly the same as that of applying a standard UGMS algorithm. Consider the following observations.

1. *Computing H:* Assuming no prior knowledge about H is given, this graph needs to be computed from the observations. This can be done using standard screening algorithms, such as those in Fan and Lv (2008) and Vats (to appear), or by applying a UGMS algorithm with a regularization parameter that selects a larger number of edges (than that computed by using a standard UGMS algorithm). Thus, the complexity of computing H is roughly the same as that of applying a UGMS algorithm to all the vertices in the graph.

2. *Applying UGMS to regions:* Recall from Algorithm 2 that we apply a UGMS algorithm to observations over $\overline{R}$ to estimate edges over the vertices R, where R is a region in a region graph representation of H. Since $|\overline{R}| \leq p$, it is clear that the complexity of Algorithm 2 is less than that of applying a UGMS algorithm to estimate all edges in the graph.

3. *Computing junction trees:* For a given graph, there exists several junction tree representations. The computational complexity of applying UGMS algorithms to a junction tree depends on the size of the clusters, the size of the separators, and the degree of the junction tree. In theory, it is useful to select a junction tree so that the overall computational complexity of the framework is as small as possible. However, this is hard

since there can be an exponential number of possible junction tree representations. Alternatively, we can select a junction tree so that the maximum size of the clusters is as small as possible. Such junction trees are often referred to as *optimal junction trees* in the literature. Although finding optimal junction trees is also hard (Arnborg et al., 1987), there exists several computationally tractable heuristics for finding close to optimal junction trees (Kjaerulff, 1990; Berry et al., 2003). The complexity of such algorithms range from $O(p^2)$ to $O(p^3)$, depending on the degree of approximation. We note that this time complexity is less than that of standard UGMS algorithms.

It is clear that the complexity of all the intermediate steps in the framework is less than that of applying a standard UGMS algorithm. The overall complexity of the framework depends on the number of clusters in the junction tree and the size of the separators in the junction tree. The size of the separators in a junction tree can be controlled by merging clusters that share a large separator. This step can be done in linear time. Removing large separators also reduces the total number of clusters in a junction tree. In the worst case, if all the separators in H are too large, the junction tree will only have one cluster that contains all the vertices. In this case, using the junction tree framework will be no different than using a standard UGMS algorithm.

5.3 Advantages of using Junction Trees and Region Graphs

An alternative approach to estimating G^* using H is to modify some current UGMS algorithms (see Appendix B for some concrete examples). For example, neighborhood selection based algorithms first estimate the neighborhood of each vertex and then combine all the estimated neighborhoods to construct an estimate $\widehat{G}$ of G^* (Meinshausen and Bühlmann, 2006; Bresler et al., 2008; Netrapalli et al., 2010; Ravikumar et al., 2010). Two ways in which these algorithms can be modified when given H are described as follows:

1. A straightforward approach is to decompose the UGMS problem into p different subproblems of estimating the neighborhood of each vertex. The graph H can be used to restrict the estimated neighbors of each vertex to be subsets of the neighbors in H. For example, in Figure 7(a), the neighborhood of 1 is estimated from the set $\{2, 3, 4, 5\}$ and the neighborhood of 3 is estimated from the set $\{1, 4, 5, 6\}$. This approach can be compared to independently applying Algorithm 2 to each region in the region graph. For example, when using the region graph, the edge $(1, 4)$ can be estimated by applying a UGMS algorithm to $\{1, 3, 4, 5\}$. In comparison, when not using region graphs, the edge $(1, 4)$ is estimated by applying a UGMS algorithm to $\{1, 2, 3, 4, 5\}$. In general, using region graphs results in smaller subproblems. A good example to illustrate this is the star graph in Figure 7(g). A junction tree representation of the star graph can be computed so that all clusters will have size two. Subsequently, the junction tree framework will only require applying a UGMS algorithm to a pair of nodes. On the other hand, neighborhood selection needs to be applied to all the nodes to estimate the neighbors of the central node 1 which is connected to all other nodes.

2. An alternative approach is to estimate the neighbors of each vertex in an iterative manner. However, it is not clear what ordering should be chosen for the vertices. The

region graph approach outlined in Section 5.1 leads to a natural choice for choosing which edges to estimate in the graph so as to reduce the problem size of subsequent subproblems. Moreover, iteratively applying neighborhood selection may still lead to large subproblems. For example, suppose the star graph in Figure 7(g) is in fact the true graph. In this case, using neighborhood selection always leads to applying UGMS to all the nodes in the graph.

From the above discussion, it is clear that using junction trees for UGMS leads to smaller subproblems and a natural choice of an ordering for estimating edges in the graph. We will see in Section 7 that the smaller subproblems lead to weaker conditions on the number of observations required for consistent graph estimation. Moreover, our numerical simulations in Section 8 empirically show the advantages of using junction tree over neighborhood selection based algorithms.

6. PC-Algorithm for UGMS

So far, we have presented the junction tree framework using an abstract undirected graphical model selection (UMGS) algorithm. This shows that our framework can be used in conjunction with any UGMS algorithm. In this section, we review the PC-Algorithm, since we use it to analyze the junction tree framework in Section 7. The PC-Algorithm was originally proposed in the literature for learning directed graphical models (Spirtes and Glymour, 1991). The first stage of the PC-Algorithm, which we refer to as PC, estimates an undirected graph using conditional independence tests. The second stage orients the edges in the undirected graph to estimate a directed graph. We use the first stage of the PC-Algorithm for UGMS. Algorithm 4 outlines PC. Variants of the PC-Algorithm for learning undirected graphical models have recently been analyzed in Anandkumar et al. (2012b,a). The main property used in PC is the global Markov property of undirected graphical models which states that if a set of vertices S separates i and j, then $X_i \perp\!\!\!\perp X_j | X_S$. As seen in Line 5 of Algorithm 4, PC deletes an edge (i, j) if it identifies a conditional independence relationship. Some properties of PC are summarized as follows:

1. *Parameter* κ: PC iteratively searches for separators for an edge (i, j) by searching for separators of size $0, 1, \ldots, \kappa$. This is reflected in Line 2 of Algorithm 4. Theoretically, the algorithm can automatically stop after searching for all possible separators for each edge in the graph. However, this may not be computationally tractable, which is why κ needs to be specified.

2. *Conditional Independence Test:* Line 5 of Algorithm 4 uses a conditional independence test to determine if an edge (i, j) is in the true graph. This makes PC extremely flexible since nonparametric independence tests may be used, see Hoeffding (1948), Rasch et al. (2012) and Zhang et al. (2012) for some examples. In this paper, for simplicity, we only consider Gaussian graphical models. In this case, conditional independence can be tested using the conditional correlation coefficient defined as

$$\text{Conditional correlation coefficient: } \rho_{ij|S} = \frac{\Sigma_{ij} - \Sigma_{i,S}\Sigma_{S,S}^{-1}\Sigma_{S,j}}{\sqrt{\Sigma_{i,i|S}\Sigma_{j,j|S}}},$$

Algorithm 4: PC-Algorithm for UGMS: $\mathsf{PC}(\kappa, \mathfrak{X}^n, H, L)$

Inputs:

κ: An integer that controls the computational complexity of PC.

$\mathfrak{X}^n$: n i.i.d. observations.

H: A graph that contains all the true edges G^*.

L: A graph that contains the edges that need to be estimated.

Output: A graph $\widehat{G}$ that contains edges in L that are estimated to be in G^*.

1 $\widehat{G} \leftarrow L$
2 **for** *each* $k \in \{0, 1, \ldots, \kappa\}$ **do**
3 **for** *each* $(i,j) \in E(\widehat{G})$ **do**
4 $\mathcal{S}_{ij} \leftarrow$ Neighbors of i or j in H depending on which one has lower cardinality.
5 **if** $\exists\ S \subset \mathcal{S}_{ij},\ |S| = k,\ s.t.\ X_i \perp\!\!\!\perp X_j | X_S$ *(computed using* $\mathfrak{X}^n$*)* **then**
6 Delete edge (i,j) from $\widehat{G}$ and H.
7 Return $\widehat{G}$.

where $P_X \sim \mathcal{N}(0, \Sigma)$, $\Sigma_{A,B}$ is the covariance matrix of X_A and X_B, and $\Sigma_{A,B|S}$ is the conditional covariance defined by

$$\Sigma_{A,B|S} = \Sigma_{A,B} - \Sigma_{A,S}\Sigma_{S,S}^{-1}\Sigma_{B,S}\,.$$

Whenever $X_i \perp\!\!\!\perp X_j | X_S$, then $\rho_{ij|S} = 0$. This motivates the following test for independence:

$$\text{Conditional Independence Test: } |\widehat{\rho}_{ij|S}| < \lambda_n \Longrightarrow X_i \perp\!\!\!\perp X_j | X_S\,, \tag{4}$$

where $\widehat{\rho}_{ij|S}$ is computed using the empirical covariance matrix from the observations $\mathfrak{X}^n$. The regularization parameter λ_n controls the number of edges estimated in $\widehat{G}$.

3. *The graphs H and L:* Recall that H contains all the edges in G^*. The graph L contains edges that need to be estimated since, as seen in Algorithm 2, we apply UGMS to only certain parts of the graph instead of the whole graph. As an example, to estimate edges in a region R of a region graph representation of H, we apply Algorithm 4 as follows:

$$\widehat{G}_R = \mathsf{PC}\left(\eta, \mathfrak{X}^n, H, H'_R\right)\,, \tag{5}$$

where H'_R is defined in (3). Notice that we do not use $\overline{R}$ in (5). This is because Line 4 of Algorithm 4 automatically finds the set of vertices to apply the PC algorithm to. Alternatively, we can apply Algorithm 4 using $\overline{R}$ as follows:

$$\widehat{G}_R = \mathsf{PC}\left(\eta, \mathfrak{X}^n_{\overline{R}}, K_{\overline{R}}, H'_R\right)\,, \tag{6}$$

where $K_{\overline{R}}$ is the complete graph over $\overline{R}$.

4. *The set $\mathcal{S}_{ij}$:* An important step in Algorithm 4 is specifying the set $\mathcal{S}_{ij}$ in Line 4 to restrict the search space for finding separators for an edge (i,j). This step significantly reduces the computational complexity of PC and differentiates PC from the first stage of the SGS-Algorithm (Spirtes et al., 1990), which specifies $\mathcal{S}_{ij} = V \backslash \{i,j\}$.

7. Theoretical Analysis of Junction Tree based PC

We use the PC-algorithm to analyze the junction tree based UGMS algorithm. Our main result, stated in Theorem 9, shows that when using the PC-Algorithm with the junction tree framework, we can potentially estimate the graph using fewer number of observations than what is required by the standard PC-Algorithm. As we shall see in Theorem 9, the particular gain in performance depends on the structure of the graph.

Section 7.1 discusses the assumptions we place on the graphical model. Section 7.2 presents the main theoretical result highlighting the advantages of using junction trees. Throughout this section, we use standard asymptotic notation so that $f(n) = \Omega(g(n))$ implies that there exists an N and a constant c such that for all $n \geq N$, $f(n) \geq cg(n)$. For $f(n) = O(g(n))$, replace $\geq$ by $\leq$.

7.1 Assumptions

(A1) *Gaussian graphical model:* We assume $X = (X_1, \ldots, X_p) \sim P_X$, where P_X is a multivariate normal distribution with mean zero and covariance Σ. Further, P_X is Markov on G^* and not Markov on any subgraph of G^*. It is well known that this is assumption translates into the fact that $\Sigma^{-1}_{ij} = 0$ if and only if $(i,j) \notin G^*$ (Speed and Kiiveri, 1986).

(A2) *Faithfulness:* If $X_i \perp\!\!\!\perp X_j | X_S$, then i and j are separated by[3] S. This assumption is important for the PC algorithm to output the correct graph. Further, note that the Markov assumption is different since it goes the other way: if i and j are separated by S, then $X_i \perp\!\!\!\perp X_j | X_S$. Thus, when both (A1) and (A2) hold, we have that $X_i \perp\!\!\!\perp X_j | X_S \iff (i,j) \notin G^*$.

(A3) *Separator Size η:* For all $(i,j) \notin G^*$, there exists a subset of nodes $S \subset V \backslash \{i,j\}$, where $|S| \leq \eta$, such that S is a separator for i and j in G^*. This assumption allows us to use $\kappa = \eta$ when using PC.

(A4) *Conditional Correlation Coefficient $\rho_{ij|S}$ and Σ:* Under (A3), we assume that $\rho_{ij|S}$ satisfies

$$\sup\{|\rho_{ij|S}| : i,j \in V, S \subset V, |S| \leq \eta\}\} \leq M < 1\,,$$

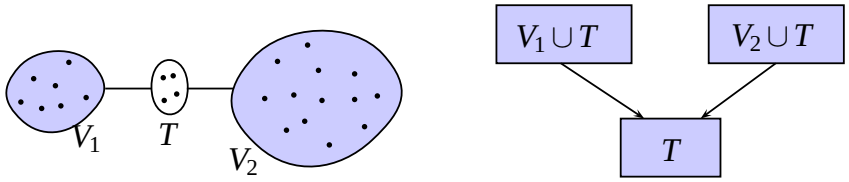
where M is a constant. Further, we assume that $\max_{i,S,|S|\leq\eta} \Sigma_{i,i|S} \leq L < \infty$.

(A5) *High-Dimensionality* We assume that the number of vertices in the graph p scales with n so that $p \to \infty$ as $n \to \infty$. Furthermore, both $\rho_{ij|S}$ and η are assumed to be functions of n and p unless mentioned otherwise.

(A6) *Structure of G^*:* Under (A3), we assume that there *exists* a set of vertices V_1, V_2, and T such that T separates V_1 and V_2 in G^* and $|T| < \eta$. Figure 8(a) shows the general structure of this assumption.

Assumptions (A1)-(A5) are standard conditions for proving high-dimensional consistency of the PC-Algorithm for Gaussian graphical models. The structural constraints on

3. If S is the empty set, then there is no path between i and j.



(a) Structure of the graph in (A5) (b) Region graph of (a)

Figure 8: General Structure of the graph we use in showing the advantages of the junction tree framework.

the graph in Assumption (A6) are required for showing the advantages of the junction tree framework. We note that although (A6) appears to be a strong assumption, there are several graph families that satisfy this assumption. For example, the graph in Figure 1(a) satisfies (A6) with $V_1 = \{1, 2\}$, $V_2 = \{1, 3, 4, 5, 6, 7\}$, and $T = \{1\}$. In general, if there exists a separator in the graph of size less than η, then (A6) is clearly satisfied. Further, we remark that we only assume the *existence* of the sets V_1, V_2, and T and do *not* assume that these sets are known *a priori*. We refer to Remark 17 for more discussions about (A6) and some extensions of this assumption.

7.2 Theoretical Result and Analysis

Recall PC in Algorithm 4. Since we assume (A1), the conditional independence test in (4) can be used in Line 5 of Algorithm 4. To analyze the junction tree framework, consider the following steps to construct $\widehat{G}$ using PC when given n i.i.d. observations $\mathfrak{X}^n$:

Step 1. Compute H: Apply PC using a regularization parameter λ_n^0 such that

$$H = \mathsf{PC}(|T|, \mathfrak{X}^n, K_V, K_V)\,,$$

where K_V is the complete graph over the nodes V. In the above equation, we apply PC to remove all edges for which there exists a separator of size less than or equal to $|T|$.

Step 2. Estimate a subset of edges over $V_1 \cup T$ and $V_2 \cup T$ using regularization parameters λ_n^1 and λ_n^2, respectively, such that

$$\widehat{G}_{V_k} = \mathsf{PC}\left(\eta, \mathfrak{X}^n, H[V_k \cup T] \cup K_T, H'_{V_k \cup T}\right), \text{for } k = 1, 2,$$

where $H'_{V_k \cup T} = H[V_k \cup T] \backslash K_T$ as defined in (3).

Step 3. Estimate edges over T using a regularization parameter λ_n^T:

$$\widehat{G}_T = \mathsf{PC}\left(\eta, \mathfrak{X}^n, H[T \cup ne_{\widehat{G}_{V_1} \cup \widehat{G}_{V_2}}(T)], H[T]\right)\,.$$

Step 4. Final estimate is $\widehat{G} = \widehat{G}_{V_1} \cup \widehat{G}_{V_2} \cup \widehat{G}_T$.

Step 1 is the screening algorithm used to eliminate some edges from the complete graph. For the region graph in Figure 8(b), Step 2 corresponds to applying PC to the regions $V_1 \cup T$ and $V_2 \cup T$. Step 3 corresponds to applying PC to the region T and all neighbors of T estimated so far. Step 4 merges all the estimated edges. Although the neighbors of T are sufficient to estimate all the edges in T, in general, depending on the graph, a smaller set of vertices is required to estimate edges in T. The main result is stated using the following terms defined on the graphical model:

$$
\begin{aligned}
&p_1 = |V_1| + |T|\,,\ p_2 = |V_2| + |T|\,,\ p_T = |T \cup ne_{G^*}(T)|\,,\ \eta_T = |T|\,,\\
&\rho_0 = \inf\{|\rho_{ij|S}| : i,j \ s.t.\ |S| \leq \eta_T \ \&\ |\rho_{ij|S}| > 0\}\,,\\
&\rho_1 = \inf\{|\rho_{ij|S}| : i \in V_1, j \in V_1 \cup T \ s.t.\ (i,j) \in E(G^*), S \subseteq V_1 \cup T, |S| \leq \eta\}\,,\\
&\rho_2 = \inf\{|\rho_{ij|S}| : i \in V_2, j \in V_2 \cup T \ s.t.\ (i,j) \in E(G^*), S \subseteq V_2 \cup T, |S| \leq \eta\}\,,\\
&\rho_T = \inf\{|\rho_{ij|S}| : i,j \in T \ s.t.\ (i,j) \in E, S \subseteq T \cup ne_{G^*}(T), \eta_T < |S| \leq \eta\}\,,
\end{aligned}
$$

The term ρ_0 is a measure of how hard it is to learn the graph H in Step 1 so that $E(G^*) \subseteq E(H)$ and all edges that have a separator of size less than $|T|$ are deleted in H. The terms ρ_1 and ρ_2 are measures of how hard it is learn the edges in $G^*[V_1 \cup T]\backslash K_T$ and $G^*[V_2 \cup T]\backslash K_T$ (Step 2), respectively, given that $E(G^*) \subseteq E(H)$. The term ρ_T is a measure of how hard it is learn the graph over the nodes T given that we know the edges that connect V_1 to T and V_2 to T.

Theorem 9 *Under Assumptions (A1)-(A6), there exists a conditional independence test such that if*

$$
n = \Omega\left(\max\left\{\rho_0^{-2}\eta_T \log(p), \rho_1^{-2}\eta \log(p_1), \rho_2^{-2}\eta \log(p_2), \rho_T^{-2}\eta \log(p_T)\right\}\right)\,, \tag{7}
$$

then $P(\widehat{G} \neq G) \to 0$ *as* $n \to \infty$.

Proof See Appendix E. ■

We now make several remarks regarding Theorem 9 and its consequences.

Remark 10 (Comparison to Necessary Conditions) Using results from Wang et al. (2010), it follows that a necessary condition for any algorithm to recover the graph G^* that satisfies Assumptions (A1) and (A6) is that $n = \Omega(\max\{\theta_1^{-2}\log(p_1 - d), \theta_2^{-2}\log(p_2 - d)\}$, where d is the maximum degree of the graph and θ_1 and θ_2 are defined as follows:

$$
\theta_k = \min_{(i,j) \in G^*[V_k \cup T]\backslash G^*[T]} \frac{|\Sigma^{-1}_{ij}|}{\sqrt{|\Sigma^{-1}_{ii}\Sigma^{-1}_{jj}|}}\,, k = 1, 2\,.
$$

If η is a constant and ρ_1 and ρ_2 are chosen so that the corresponding expressions dominate all other expressions, then (7) reduces to $n = \Omega(\max\{\rho_1^{-2}\log(p_1), \rho_2^{-2}\log(p_2)\})$. Furthermore, for certain classes of Gaussian graphical models, namely walk summable graphical models (Malioutov et al., 2006), the results in Anandkumar et al. (2012a) show that there exists conditions under which $\rho_1 = \Omega(\theta_1)$ and $\rho_2 = \Omega(\theta_2)$. In this case, (7) is equivalent to $n = \Omega(\max\{\theta_1^{-2}\log(p_1), \theta_2^{-2}\log(p_2)\})$. Thus, as long as $p_1, p_2 \gg d$, there exists a family

of graphical models for which the sufficient conditions in Theorem 9 nearly match the necessary conditions for asymptotically reliable estimation of the graph. We note that the particular family of graphical models is quite broad, and includes forests, scale-free graphs, and some random graphs. We refer to Anandkumar et al. (2012a) for a characterization of such graphical models.

Remark 11 (Choice of Regularization Parameters) We use the conditional independence test in (4) that thresholds the conditional correlation coefficient. From the proof in Appendix E, the thresholds, which we refer to as the regularization parameter, are chosen as follows:

$$\begin{aligned}
&\lambda_n^0 = O(\rho_0) \text{ and } \rho_0 = \Omega\left(\sqrt{\eta_T \log(p)/n}\right), \\
&\lambda_n^k = O(\rho_k) \text{ and } \rho_k = \Omega\left(\sqrt{\eta \log(p_k)/n}\right), k = 1, 2, \\
&\lambda_n^T = O(\rho_T) \text{ and } \rho_T = \Omega\left(\sqrt{\eta \log(p_T)/n}\right).
\end{aligned}$$

We clearly see that different regularization parameters are used to estimate different parts of the graph. Furthermore, just like in the traditional analysis of UGMS algorithms, the optimal choice of the regularization parameter depends on unknown parameters of the graphical model. In practice, we use model selection algorithms to select regularization parameters. We refer to Section 8 for more details.

Remark 12 (Weaker Condition) If we do not use the junction tree based approach outlined in Steps 1-4, and instead directly apply PC, the sufficient condition on the number of observations will be $n = \Omega(\rho_{min}^{-2} \eta \log(p))$, where

$$\rho_{min} := \inf\{|\rho_{ij|S}| : (i,j) \in E(G^*), |S| \leq \eta\}.$$

This result is proved in Appendix D using results from Kalisch and Bühlmann (2007) and Anandkumar et al. (2012a). Since $\rho_{min} \leq \min\{\rho_0, \rho_1, \rho_2, \rho_T\}$, it is clear that (7) is a weaker condition. The main reason for this difference is that the junction tree approach defines an *ordering* on the edges to test if an edge belongs to the true graph. This ordering allows for a reduction in separator search space (see $\mathcal{S}_{ij}$ in Algorithm 4) for testing edges over the set T. Standard analysis of PC assumes that the edges are tested randomly, in which case, the separator search space is always upper bounded by the full set of nodes.

Remark 13 (Reduced Sample Complexity) Suppose η, ρ_0, and ρ_T are constants and $\rho_1 < \rho_2$. In this case, (7) reduces to

$$n = \Omega\left(\max\left\{\log(p), \rho_1^{-2}\log(p_1), \rho_2^{-2}\log(p_2)\right\}\right). \tag{8}$$

If $\rho_1^{-2} = \Omega\left(\max\left\{\rho_2^{-2}\log(p_2)/\log(p_1), \log(p)\right\}\right)$, then (8) reduces to

$$n = \Omega\left(\rho_1^{-2}\log(p_1)\right).$$

On the other hand, if we do not use junction trees, $n = \Omega\left(\rho_{min}^{-2}\log(p)\right)$, where $\rho_{min} \leq \rho_1$. Thus, if $p_1 \ll p$, for example $p_1 = \log(p)$, then using the junction tree based PC

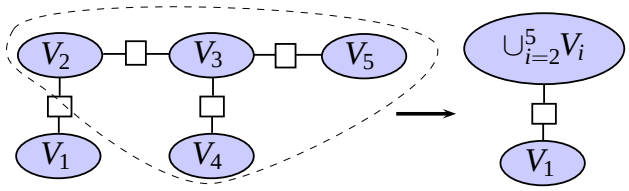


Figure 9: Junction tree representation with clusters $V_1, \ldots, V_5$ and separators denotes by rectangular boxes. We can cluster vertices in the junction tree to get a two cluster representation as in Figure 8.

requires lower number of observations for consistent UGMS. Informally, the above condition says that if the graph structure in (A6) is easy to identify, $p_1 \ll p_2$, and the minimal conditional correlation coefficient over the true edges lies in the smaller cluster (but not over the separator), the junction tree framework may accurately learn the graph using significantly less number of observations.

Remark 14 (Learning Weak Edges) We now analyze Theorem 9 to see how the conditional correlation coefficients scale for high-dimensional consistency. Under the assumption in Remark 13, it is easy to see that the minimal conditional correlation coefficient scales as $\Omega(\sqrt{\log(p_1)/n})$ when using junction trees and as $\Omega(\sqrt{\log(p)/n})$ when not using junction trees. This suggests that when $p_1 \ll p$, it may be possible to learn edges with weaker conditional correlation coefficients when using junction trees. Our numerical simulations in Section 8 empirically show this property of the junction tree framework.

Remark 15 (Computational complexity) It is easy to see that the worst case computational complexity of the PC-Algorithm is $O(p^{\eta+2})$ since there are $O(p^2)$ edges and testing for each edge requires a search over at most $O(p^\eta)$ separators. The worst case computational complexity of Steps 1-4 is roughly $O\left(p^{|T|+2} + p_1^{\eta+2} + p_2^{\eta+2} + p_T^{\eta+2}\right)$. Under the conditions in Remark 8.3 and when $p_1 \ll p$, this complexity is roughly $O(p^{\eta+2})$, which is the same as the standard PC-Algorithm. In practice, especially when the graph is sparse, the computational complexity is much less than $O(p^{\eta+2})$ since the PC-Algorithm restricts the search space for finding separators.

Remark 16 (Using other UGMS Algorithms) Although our analysis used the PC-Algorithm to derive sufficient conditions for accurately estimating the graph, we can easily use other algorithms, such as the graphical Lasso or the neighborhood selection based Lasso, for analysis. The main difference will be in the assumptions imposed on the graphical model.

Remark 17 (Extensions) We have analyzed the junction tree framework assuming that the junction tree of H only has two clusters. One way to generalize our analysis to junction trees with multiple clusters is to merge clusters so that the resulting junction tree admits

the structure in Figure 8. For example, suppose the graph G^* has a junction tree representation as in Figure 9 with five clusters. If $|V_1 \cap V_2| < \eta$, then we can merge the clusters $V_2, V_3, \ldots, V_5$ so that the resulting junction tree admits the two cluster representation in Figure 8. Furthermore, we can also generalize Theorem 9 to cases when $|T| = \eta$. The main change in the analysis will be in the definition of ρ_0. For example, if the graph is a chain so that the first p_1 vertices are associated with "weak edges", we can get similar results as in Theorem 9. Finally, we note that a full analysis of the junction tree framework, that also incorporates the step of updating the junction tree in Algorithm 3, is challenging and will be addressed in future work.

8. Numerical Simulations

In this section, we present numerical simulations that highlight the advantages of using the junction tree framework for UGMS. Throughout this section, we assume a Gaussian graphical model such that $P_X \sim \mathcal{N}(0, \Theta^{-1})$ is Markov on G^*. It is well known that this implies that $(i,j) \notin G^* \iff \Theta_{ij} = 0$ (Speed and Kiiveri, 1986). Some algorithmic details used in the simulations are described as follows.

Computing H: We apply Algorithm 4 with a suitable value of κ in such a way that the separator search space $\mathcal{S}_{ij}$ (see Line 4) is restricted to be small. In other words, we do not test for all possible conditional independence tests so as to restrict the computational complexity of the screening algorithm. We use the conditional partial correlation to test for conditional independence and choose a *separate threshold* to test for each edge in the graph. The thresholds for the conditional independence test are computed using 5-fold cross-validation. The computational complexity of this step is roughly $O(p^2)$ since there are $O(p^2)$ edges to be tested. Note that this method for computing H is equivalent to Step 1 in Section 7.2 with $|T| = \kappa$. Finally, we note that the above method does not guarantee that all edges in G^* will be included in H. This can result in false edges being included in the junction tree estimated graphs. To avoid this situation, once a graph estimate $\widehat{G}$ has been computed using the junction tree based UGMS algorithm, we apply conditional independence tests again to prune the estimated edge set.

Computing the junction tree: We use standard algorithms in the literature for computing close to optimal junction trees.[4] Once the junction tree is computed, we merge clusters so that the maximum size of the separator is at most $\kappa + 1$, where κ is the parameter used when computing the graph H. For example, in Figure 9, if the separator associated with V_2 and V_3 has cardinality greater than $\kappa + 1$, then we merge V_2 and V_3 and resulting junction tree is such that V_1, V_4, and V_5 all connect to the cluster $V_2 \cup V_3$.

UGMS Algorithms: We apply the junction tree framework in conjunction with graphical Lasso (gL) (Banerjee et al., 2008), neighborhood selection using Lasso (nL) (Meinshausen and Bühlmann, 2006), and the PC-Algorithm (PC) (Spirtes and Glymour, 1991). See Appendix B for a review of gL and nL and Algorithm 4 for PC. When using nL, we use the intersection rule to combine neighborhood estimates. Further, we use the adaptive Lasso (Zou, 2006) for finding neighbors of a vertex since this is known to give superior results for variable selection (van de Geer et al., 2011).

4. We use the GreedyFillin heuristic. This is known to give good results with reasonable computational time (Kjaerulff, 1990).

Choosing Regularization Parameters: An important step when applying UGMS algorithms is to choose a suitable regularization parameter. It is now well known that classical methods, such as cross-validation and information criterion based methods, tend to choose a much larger number of edges when compared to an oracle estimator for high-dimensional problems (Meinshausen and Bühlmann, 2010; Liu et al., 2010). Several alternative methods have been proposed in the literature; see for example stability selection (Meinshausen and Bühlmann, 2010; Liu et al., 2010) and extended Bayesian information (EBIC) criterion (Chen and Chen, 2008; Foygel and Drton, 2010). In all our simulations, we use EBIC since it is much faster than stability based methods when the distribution is Gaussian. EBIC selects a regularization parameter $\widehat{\lambda}_n$ as follows:

$$\widehat{\lambda}_n = \max_{\lambda_n > 0} \left\{ n \left[\log\det \widehat{\Theta}_{\lambda_n} - \mathrm{trace}(\widehat{S}\Theta) \right] + |E(\widehat{G}_{\lambda_n})| \log n + 4\gamma |E(\widehat{G}_{\lambda_n})| \log p \right\},$$

where $\widehat{S}$ is the empirical covariance matrix, $\widehat{\Theta}_{\lambda_n}$ is the estimate of the inverse covariance matrix and $|E(\widehat{G}_{\lambda_n})|$ is the number of edges in the estimated graph. The estimate $\widehat{\lambda}_n$ depends on a parameter $\gamma \in [0, 1]$ such that $\gamma = 0$ results in the BIC estimate and increasing γ produces sparser graphs. The authors in reference Foygel and Drton (2010) suggest that $\gamma = 0.5$ is a reasonable choice for high-dimensional problems. When solving subproblems using Algorithm 2, the $\log p$ term is replaced by $\log |\overline{R}|$, $\widehat{\Theta}_{\lambda_n}$ is replaced by the inverse covariance over the vertices $\overline{R}$, and $|\widehat{G}_{\lambda_n}|$ is replaced by the number of edges estimated from the graph H'_R.
Small subproblems: Whenever $|\overline{R}|$ is small (less than 8 in our simulations), we independently test whether each edge is in G^* using hypothesis testing. This shows the application of using different algorithms to learn different parts of the graph.

8.1 Results on Synthetic Graphs

We assume that $\Theta_{ii} = 1$ for all $i = 1, \ldots, p$. We refer to all edges connected to the first p_1 vertices as *weak edges* and the rest of the edges are referred to as *strong edges*. The different types of synthetic graphical models we study are described as follows:

- Chain (CH_1 and CH_2): $\Theta_{i,i+1} = \rho_1$ for $i = 1, \ldots, p_1 - 1$ (weak edges) and $\Theta_{i,i+1} = \rho_2$ for $i = p_1, p - 1$ (strong edges). For CH_1, $\rho_1 = 0.15$ and $\rho_2 = 0.245$. For CH_2, $\rho_1 = 0.075$ and $\rho_2 = 0.245$. Let $\Theta_{ij} = \Theta_{ji}$.

- Cycle (CY_1 and CY_2): $\Theta_{i,i+1} = \rho_1$ for $i = 1, \ldots, p_1 - 1$ (weak edges) and $\Theta_{i,i+1} = \rho_2$ for $i = p_1, p - 1$ (strong edges). In addition, $\Theta_{i,i+3} = \rho_1$ for $i = 1, \ldots, p_1 - 3$ and $\Theta_{i,i+3} = \rho_2$ for $i = p_1, p_1 + 1, \ldots, p - 3$. This introduces multiple cycles in the graph. For CY_1, $\rho_1 = 0.15$ and $\rho_2 = 0.245$. For CY_2, $\rho_1 = 0.075$ and $\rho_2 = 0.245$.

- Hub (HB_1 and HB_2): For the first p_1 vertices, construct as many star[5] graphs of size d_1 as possible. For the remaining vertices, construct star graphs of size d_2 (at most one may be of size less than d_2). The hub graph G^* is constructed by taking a union of all star graphs. For $(i, j) \in G^*$ s.t. $i, j \leq p_1$, let $\Theta_{i,j} = 1/d_1$. For the remaining edges, let $\Theta_{ij} = 1/d_2$. For HB_1, $d_1 = 8$ and $d_2 = 5$. For HB_2, $d_1 = 12$ and $d_2 = 5$.

5. A star is a tree where one vertex is connected all other vertices.

- Neighborhood graph ($\mathsf{NB_1}$ and $\mathsf{NB_2}$): Randomly place vertices on the unit square at coordinates $y_1, \ldots, y_p$. Let $\Theta_{ij} = 1/\rho_1$ with probability $(\sqrt{2\pi})^{-1} \exp(-4||y_i - y_j||_2^2)$, otherwise $\Theta_{ij} = 0$ for all $i, j \in \{1, \ldots, p_1\}$ such that $i > j$. For all $i, j \in \{p_1+1, \ldots, p\}$ such that $i > j$, $\Theta_{ij} = \rho_2$. For edges over the first p_1 vertices, delete edges so that each vertex is connected to at most d_1 other vertices. For the vertices $p_1 + 1, \ldots, p$, delete edges such that the neighborhood of each vertex is at most d_2. Finally, randomly add four edges from a vertex in $\{1, \ldots, p_1\}$ to a vertex in $\{p_1, p_1 + 1, \ldots, p\}$ such that for each such edge, $\Theta_{ij} = \rho_1$. We let $\rho_2 = 0.245$, $d_1 = 6$, and $d_2 = 4$. For $\mathsf{NB_1}$, $\rho_1 = 0.15$ and for $\mathsf{NB_2}$, $\rho_2 = 0.075$.

Notice that the parameters associated with the weak edges are lower than the parameters associated with the strong edges. Some comments regarding notation and usage of various algorithms is given as follows.

- The junction tree versions of the UGMS algorithms are denoted by JgL, JPC, and JnL.

- We use EBIC with $\gamma = 0.5$ to choose regularization parameters when estimating graphs using JgL and JPC. To objectively compare JgL (JPC) and gL (PC), we make sure that the number of edges estimated by gL (PC) is roughly the same as the number of edges estimated by JgL (JPC).

- The nL and JnL estimates are computed differently since it is difficult to control the number of edges estimated using both these algorithms.[6] We apply both nL and JnL for multiple different values of γ (the parameter for EBIC) and choose graphs so that the number of edges estimated is closest to the number of edges estimated by gL.

- When applying PC and JPC, we choose κ as 1, 2, 1, and 3 for Chain, Cycle, Hub, and Neighborhood graphs, respectively. When computing H, we choose κ as 0, 1, 0, and 2 for Chain, Cycle, Hub, and Neighborhood graphs, respectively.

Tables 2-5 summarize the results for the different types of synthetic graphical models. For an estimate $\widehat{G}$ of G^*, we evaluate $\widehat{G}$ using the weak edge discovery rate (WEDR), false discovery rate (FDR), true positive rate (TPR), and the edit distance (ED).

$$\begin{aligned}
\text{WEDR} &= \frac{\#\text{ weak edges in } \widehat{G}}{\#\text{ of weak edges in } G^*}, \\
\text{FDR} &= \frac{\#\text{ of edges in } \widehat{G} \backslash G^*}{\#\text{ of edges in } \widehat{G}}, \\
\text{TPR} &= \frac{\#\text{ of edges in } \widehat{G} \cap G^*}{\#\text{ of edges in } G^*}, \\
\text{ED} &= \{\#\text{ edges in } \widehat{G} \backslash G^*\} + \{\#\text{ edges in } G^* \backslash \widehat{G}\},
\end{aligned}$$

6. Recall that both these algorithms use different regularization parameters. Thus, there may exist multiple different estimates with the same number of edges.

Model	n	Alg	WEDR	FDR	TPR	ED	$\lvert\widehat{G}\rvert$
CH_1	300	JgL	**0.305** (0.005)	0.048 (0.001)	0.767 (0.002)	27.0 (0.176)	79.8
$p = 100$		gL	0.180 (0.004)	0.061 (0.001)	0.757 (0.001)	29.0 (0.153)	79.8
		JPC	**0.312** (0.004)	0.047 (0.001)	0.775 (0.001)	26.0 (0.162)	80.5
		PC	0.264 (0.005)	0.047 (0.001)	0.781 (0.001)	25.6 (0.169)	81.2
		JnL	**0.306** (0.005)	0.072 (0.001)	0.769 (0.002)	28.8 (0.188)	82.1
		nL	0.271 (0.005)	0.073 (0.001)	0.757 (0.001)	30.0 (0.197)	80.9
CH_2	300	JgL	**0.052** (0.002)	0.067 (0.001)	0.727 (0.001)	32.2 (0.173)	77.3
$p = 100$		gL	0.009 (0.001)	0.062 (0.001)	0.733 (0.002)	31.3 (0.162)	77.4
		JPC	**0.048** (0.002)	0.064 (0.001)	0.735 (0.001)	31.2 (0.169)	77.8
		PC	0.0337 (0.002)	0.055 (0.001)	0.748 (0.001)	29.3 (0.144)	78.4
		JnL	**0.052** (0.002)	0.077 (0.001)	0.733 (0.001)	32.5 (0.186)	78.7
		nL	0.039 (0.002)	0.086 (0.001)	0.723 (0.001)	34.2 (0.216)	78.4
CH_1	500	JgL	**0.596** (0.006)	0.021 (0.001)	0.916 (0.001)	10.2 (0.133)	92.6
$p = 100$		gL	0.44 (0.005)	0.050 (0.001)	0.889 (0.001)	15.6 (0.132)	92.7
		JPC	**0.612** (0.005)	0.022 (0.001)	0.921 (0.001)	9.86 (0.128)	93.2
		PC	0.577 (0.005)	0.032 (0.001)	0.916 (0.001)	11.4 (0.124)	93.7
		JnL	**0.623** (0.005)	0.059 (0.001)	0.922 (0.001)	13.5 (0.133)	97.0
		nL	0.596 (0.005)	0.069 (0.001)	0.918 (0.001)	14.9 (0.164)	97.6
CH_2	500	JgL	**0.077** (0.002)	0.044 (0.001)	0.816 (0.001)	22.0 (0.107)	84.5
$p = 100$		gL	0.0211 (0.001)	0.053 (0.001)	0.808 (0.000)	23.5 (0.082)	84.6
		JPC	**0.073** (0.002)	0.042 (0.001)	0.817 (0.001)	21.7 (0.082)	84.5
		PC	0.0516 (0.002)	0.049 (0.001)	0.815 (0.001)	22.5 (0.092)	84.9
		JnL	**0.076** (0.002)	0.070 (0.001)	0.818 (0.001)	24.2 (0.102)	87.2
		nL	0.066 (0.002)	0.077 (0.001)	0.815 (0.001)	25.1 (0.126)	87.5

Table 2: Results for Chain graphs: $p = 100$ and $p_1 = 20$

Recall that the weak edges are over the first p_1 vertices in the graph. Naturally, we want WEDR and TPR to be large and FDR and ED to be small. Each entry in the table shows the mean value and standard error (in brackets) over 50 observations. We now make some remarks regarding the results.

Remark 18 (Graphical Lasso) Of all the algorithms, graphical Lasso (gL) performs the worst. On the other hand, junction tree based gL significantly improves the performance of gL. Moreover, the performance of JgL is comparable, and sometimes even better, when compared to JPC and JnL. This suggests that when using gL in practice, it is beneficial to apply a screening algorithm to remove some edges and then use the junction tree framework in conjunction with gL.

Remark 19 (PC-Algorithm and Neighborhood Selection) Although using junction trees in conjunction with the PC-Algorithm (PC) and neighborhood selection (nL) does improve the graph estimation performance, the difference is not as significant as gL. The reason is because both PC and nL make use of the local Markov property in the graph H. The junction tree framework further improves the performance of these algorithms by making use of the global Markov property, in addition to the local Markov property.

Remark 20 (Chain Graph) Although the chain graph does not satisfy the conditions in (A6), the junction tree estimates still outperforms the non-junction tree estimates. This suggests the advantages of using junction trees beyond the graphs considered in (A6). We suspect that correlation decay properties, which have been studied extensively in Anandkumar et al. (2012b,a), can be used to weaken the assumption in (A6).

Model	n	Alg	WEDR	FDR	TPR	ED	$\lvert\widehat{G}\rvert$
CY_1	300	JgL	**0.314** (0.003)	0.036 (0.001)	0.814 (0.001)	28.5 (0.142)	111
$p = 100$		gL	0.105 (0.003)	0.057 (0.001)	0.798 (0.001)	32.9 (0.16)	112
		JPC	**0.326** (0.004)	0.030 (0.001)	0.819 (0.001)	27.2 (0.18)	112
		PC	0.307 (0.004)	0.027 (0.001)	0.826 (0.001)	26 (0.169)	112
		JnL	**0.342** (0.004)	0.043 (0.001)	0.813 (0.001)	29.5 (0.175)	112
		nL	0.299 (0.004)	0.044 (0.001)	0.793 (0.001)	32.3 (0.192)	110
CY_2	300	JgL	**0.047** (0.002)	0.045 (0.001)	0.762 (0.001)	36.2 (0.163)	105
$p = 100$		gL	0.001 (0.001)	0.049 (0.001)	0.759 (0.001)	37.0 (0.172)	105
		JPC	**0.043** (0.002)	0.042 (0.001)	0.764 (0.001)	35.6 (0.174)	105
		PC	0.027 (0.002)	0.036 (0.001)	0.773 (0.001)	33.7 (0.137)	106
		JnL	**0.042** (0.002)	0.058 (0.002)	0.754 (0.001)	38.6 (0.210)	106
		nL	0.035 (0.002)	0.057 (0.002)	0.743 (0.001)	39.9 (0.228)	104
CY_1	500	JgL	**0.532** (0.005)	0.022 (0.001)	0.907 (0.001)	15.1 (0.139)	122
$p = 100$		gL	0.278 (0.001)	0.071 (0.001)	0.862 (0.001)	26.9 (0.178)	122
		JPC	**0.61** (0.004)	0.012 (0.001)	0.925 (0.001)	11.9 (0.150)	124
		PC	0.609 (0.004)	0.020 (0.001)	0.925 (0.001)	12.5 (0.134)	125
		JnL	**0.612** (0.005)	0.028 (0.001)	0.924 (0.001)	13.6 (0.151)	125
		nL	0.584 (0.005)	0.041 (0.001)	0.919 (0.001)	15.9 (0.171)	126
CY_2	500	JgL	**0.086** (0.003)	0.039 (0.001)	0.821 (0.001)	28.1 (0.116)	113
$p = 100$		gL	0.004 (0.001)	0.058 (0.001)	0.805 (0.000)	32.3 (0.088)	113
		JPC	**0.087** (0.002)	0.034 (0.001)	0.825 (0.001)	27.0 (0.099)	113
		PC	0.074 (0.002)	0.040 (0.001)	0.823 (0.001)	27.9 (0.010)	113
		JnL	**0.085** (0.003)	0.045 (0.001)	0.824 (0.001)	28.4 (0.147)	114
		nL	0.069 (0.003)	0.053 (0.001)	0.821 (0.001)	29.8 (0.158)	114

Table 3: Results for Cycle graphs, $p = 100$ and $p_1 = 20$

Model	n	Alg	WEDR	FDR	TPR	ED	$\lvert\widehat{G}\rvert$
HB_1	300	JgL	**0.204** (0.004)	0.039 (0.001)	0.755 (0.002)	22.3 (0.151)	63.7
$p = 100$		gL	0.154 (0.004)	0.038 (0.001)	0.758 (0.002)	22.1 (0.130)	63.8
		JPC	**0.204** (0.004)	0.038 (0.001)	0.753 (0.002)	22.4 (0.160)	63.4
		PC	0.193 (0.004)	0.038 (0.001)	0.762 (0.002)	21.7 (0.143)	64.2
		JnL	0.245 (0.005)	0.089 (0.001)	0.750 (0.002)	26.2 (0.174)	66.7
		nL	**0.247** (0.005)	0.098 (0.002)	0.752 (0.002)	26.8 (0.198)	67.6
HB_2	300	JgL	**0.044** (0.002)	0.047 (0.001)	0.710 (0.001)	26.7 (0.116)	61.2
$p = 100$		gL	0.013 (0.002)	0.043 (0.001)	0.716 (0.001)	26.0 (0.121)	61.4
		JPC	**0.048** (0.002)	0.043 (0.001)	0.709 (0.001)	26.5 (0.108)	60.8
		PC	0.029 (0.002)	0.038 (0.001)	0.718 (0.001)	25.5 (0.121)	61.3
		JnL	**0.054** (0.003)	0.083 (0.001)	0.704 (0.001)	29.6 (0.146)	63.0
		nL	0.0467 (0.002)	0.096 (0.001)	0.700 (0.001)	30.7 (0.138)	63.5
HB_1	500	JgL	**0.413** (0.007)	0.026 (0.001)	0.870 (0.002)	12.4 (0.156)	72.4
$p = 100$		gL	0.364 (0.007)	0.035 (0.001)	0.863 (0.002)	13.7 (0.144)	72.5
		JPC	0.438 (0.007)	0.027 (0.001)	0.878 (0.002)	11.9 (0.148)	73.1
		PC	**0.448** (0.007)	0.027 (0.001)	0.882 (0.001)	11.6 (0.141)	73.4
		JnL	0.507 (0.006)	0.076 (0.001)	0.890 (0.001)	14.9 (0.152)	78.2
		nL	**0.52** (0.007)	0.091 (0.001)	0.893 (0.002)	15.9 (0.191)	79.6
HB_2	500	JgL	**0.086** (0.003)	0.042 (0.001)	0.794 (0.001)	19.8 (0.086)	68.0
$p = 100$		gL	0.050 (0.002)	0.047 (0.001)	0.789 (0.001)	20.6 (0.098)	68.0
		JPC	**0.097** (0.003)	0.040 (0.001)	0.798 (0.001)	19.3 (0.109)	68.2
		PC	0.087 (0.003)	0.044 (0.001)	0.797 (0.001)	19.7 (0.111)	68.4
		JnL	**0.123** (0.004)	0.084 (0.002)	0.804 (0.001)	22.2 (0.15)	72.1
		nL	0.106 (0.003)	0.105 (0.002)	0.801 (0.001)	24.1 (0.143)	73.4

Table 4: Results for Hub graphs:$p = 100$ and $p_1 = 20$

Model	n	Alg	WEDR	FDR	TPR	ED	$\lvert\widehat{G}\rvert$
NB_1	300	JgL	**0.251** (0.002)	0.030 (0.000)	0.813 (0.000)	126 (0.329)	498
$p = 100$		gL	0.102 (0.0015)	0.039 (0.000)	0.806 (0.001)	135 (0.345)	498
		JPC	**0.259** (0.002)	0.031 (0.000)	0.814 (0.000)	126 (0.260)	499
		PC	0.255 (0.002)	0.036 (0.000)	0.813 (0.000)	129 (0.330)	501
		JnL	**0.254** (0.002)	0.035 (0.000)	0.812 (0.001)	129 (0.461)	500
		nL	0.226 (0.002)	0.039 (0.000)	0.804 (0.001)	136 (0.458)	497
NB_1	300	JgL	**0.005** (0.000)	0.043 (0.000)	0.784 (0.001)	149 (0.385)	486
$p = 100$		gL	0.000 (0.000)	0.036 (0.000)	0.790 (0.000)	142 (0.259)	487
		JPC	**0.004** (0.000)	0.042 (0.000)	0.784 (0.001)	148 (0.376)	486
		PC	0.003 (0.000)	0.048 (0.000)	0.782 (0.000)	153 (0.239)	488
		JnL	**0.005** (0.000)	0.046 (0.000)	0.783 (0.000)	151 (0.356)	488
		nL	0.003 (0.000)	0.050 (0.000)	0.775 (0.000)	158 (0.374)	485
NB_1	500	JgL	**0.449** (0.001)	0.018 (0.000)	0.921 (0.000)	57.1 (0.199)	557
$p = 100$		gL	0.319 (0.002)	0.035 (0.000)	0.905 (0.000)	75.8 (0.242)	557
		JPC	0.489 (0.002)	0.019 (0.000)	0.925 (0.000)	52.8 (0.189)	558
		PC	**0.496** (0.002)	0.023 (0.000)	0.920 (0.000)	60.2 (0.214)	559
		JnL	**0.508** (0.003)	0.027 (0.000)	0.929 (0.000)	57.9 (0.348)	567
		nL	0.494 (0.003)	0.033 (0.000)	0.927 (0.000)	62.3 (0.400)	570
NB_2	500	JgL	**0.008** (0.000)	0.033 (0.000)	0.870 (0.000)	95.0 (0.206)	534
$p = 100$		gL	0.000 (0.000)	0.034 (0.000)	0.869 (0.000)	96.0 (0.214)	534
		JPC	**0.009** (0.000)	0.032 (0.000)	0.870 (0.000)	94.2 (0.215)	534
		PC	0.005 (0.000)	0.040 (0.000)	0.865 (0.000)	102 (0.207)	536
		JnL	**0.001** (0.000)	0.038 (0.000)	0.871 (0.000)	97.3 (0.220)	538
		nL	0.005 (0.000)	0.043 (0.000)	0.870 (0.000)	101 (0.234)	540

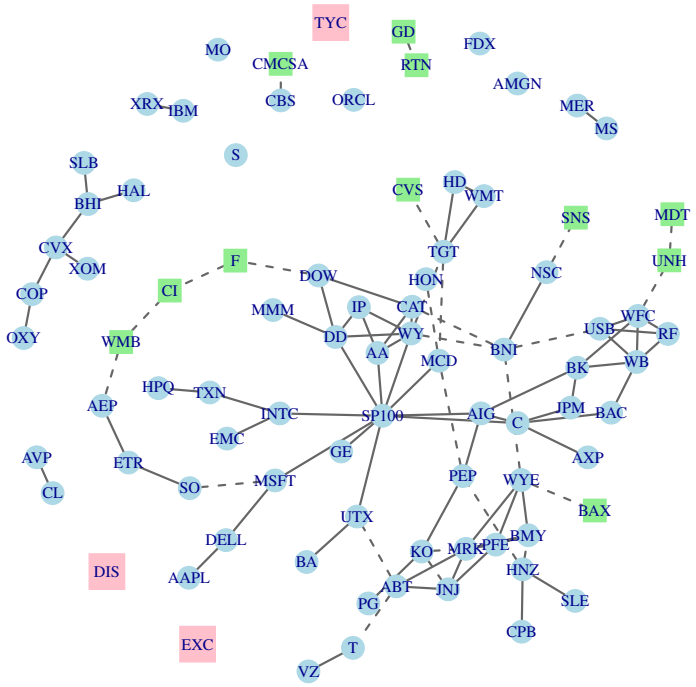
Table 5: Results for Neighborhood graph, $p = 300$ and $p_1 = 30$

Remark 21 (Hub Graph) For the hub graph HB_1, the junction tree estimate does not result in a significant difference in performance, especially for the PC and nL algorithms. This is mainly because this graph is extremely sparse with multiple components. For the number of observations considered, H removes a significant number of edges. However, for HB_2, the junction tree estimate, in general, performs slightly better. This is because the parameters associated with the weak edges in HB_2 are smaller than that of HB_1.

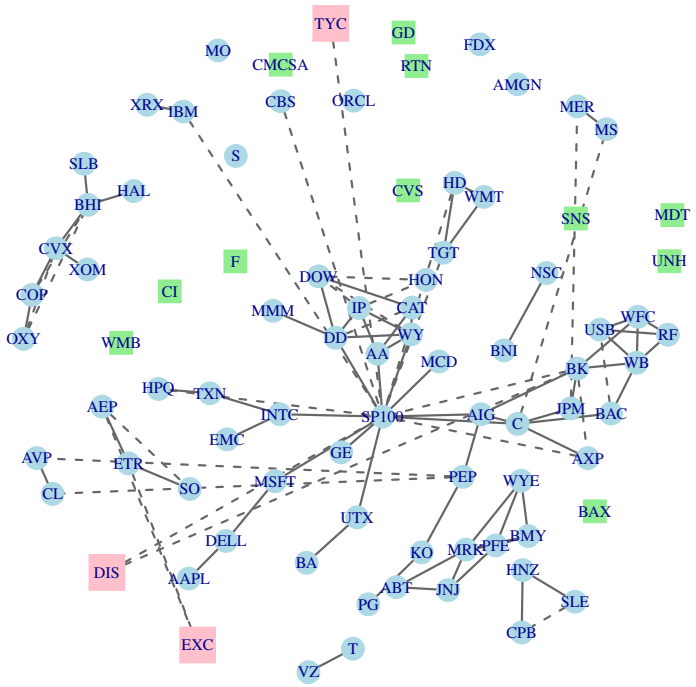
Remark 22 (General Conclusion) We see that, in general, the WEDR and TPR are higher, while the FDR and ED are lower, for junction tree based algorithms. This clearly suggests that using junction trees results in more accurate graph estimation. Moreover, the higher WEDR suggest that the main differences between the two algorithms are over the weak edges, that is, junction tree based algorithms are estimating more weak edges when compared to a non junction tree based algorithm.

8.2 Analysis of Stock Returns Data

We applied our methods to the data set in Choi et al. (2011) of $n = 216$ monthly stock returns of $p = 85$ companies in the S&P 100. We computed H using $\kappa = 1$. We applied JgL using EBIC with $\gamma = 0.5$ and applied gL so that both graphs have the same number of edges. This allows us to objectively compare the gL and JgL graphs. Figure 10 shows the two estimated graphs in such a way that the vertices are positioned so that the JgL graph looks aesthetically pleasing. In Figure 11, the vertices are positioned so that gL looks aesthetically pleasing. In each graph, we mark the common edges by bold lines and the

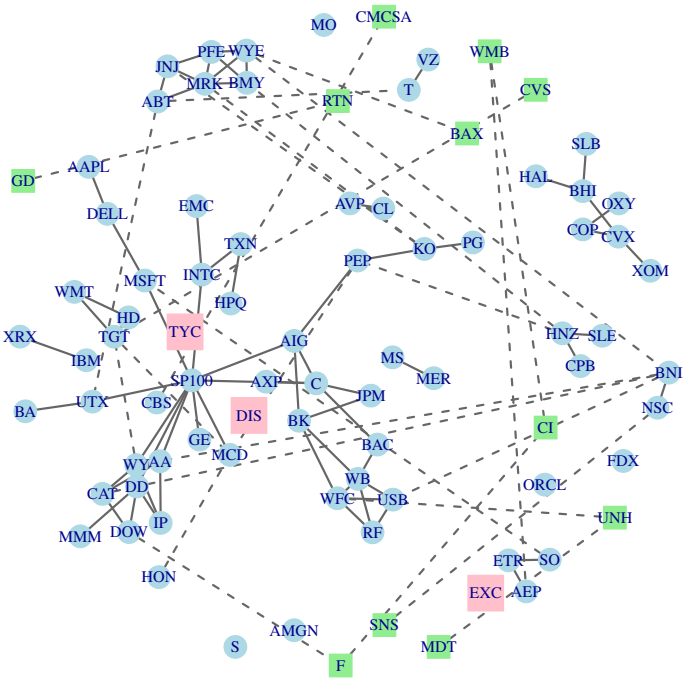


(a) Junction tree based graphical Lasso

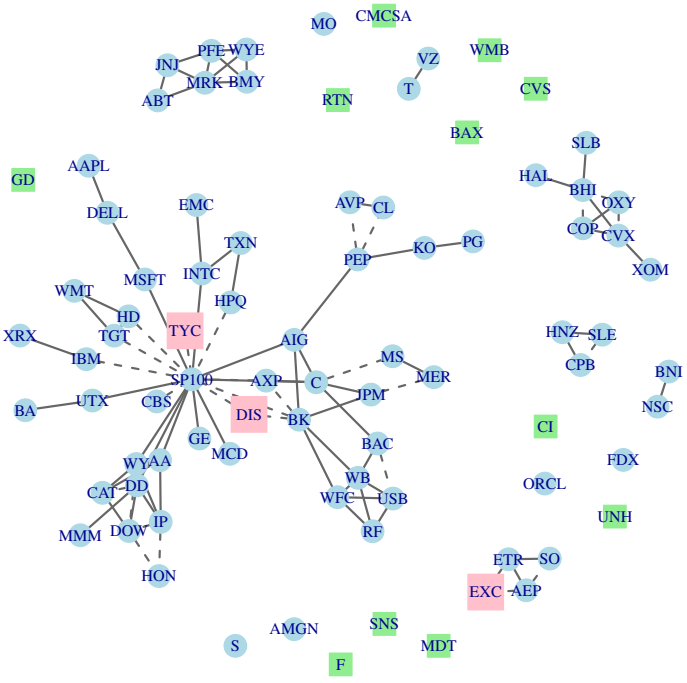


(b) Graphical Lasso

Figure 10: Graph over a subset of companies in the S&P 100. The positioning of the vertices is chosen so that the *junction tree based graph is aesthetically pleasing*. The edges common in (a) and (b) are marked by bold lines and the remaining edges are marked by dashed lines



(a) Junction tree based graphical Lasso



(b) Graphical Lasso

Figure 11: Graph over a subset of companies in the S&P 100. The positioning of the vertices is chosen so that the *graphical Lasso based graph is aesthetically pleasing*. The edges common in (a) and (b) are marked by bold lines and the remaining edges are marked by dashed lines.

remaining edges by dashed lines. Some conclusions that we draw from the estimated graphs are summarized as follows:

- The gL graph in Figure 11(b) seems well structured with multiple different clusters of nodes with companies that seem to be related to each other. A similar clustering is seen for the JgL graph in Figure 10(a) with the exception that there are now connections between the clusters. As observed in Choi et al. (2011) and Chandrasekaran et al. (2012), it has been hypothesized that the "actual" graph over the companies is dense since there are several unobserved companies that induce conditional dependencies between the observed companies. These induced conditional dependencies can be considered to be the weak edges of the "actual" graph. Thus, our results suggest that the junction tree based algorithm is able to detect such weak edges.

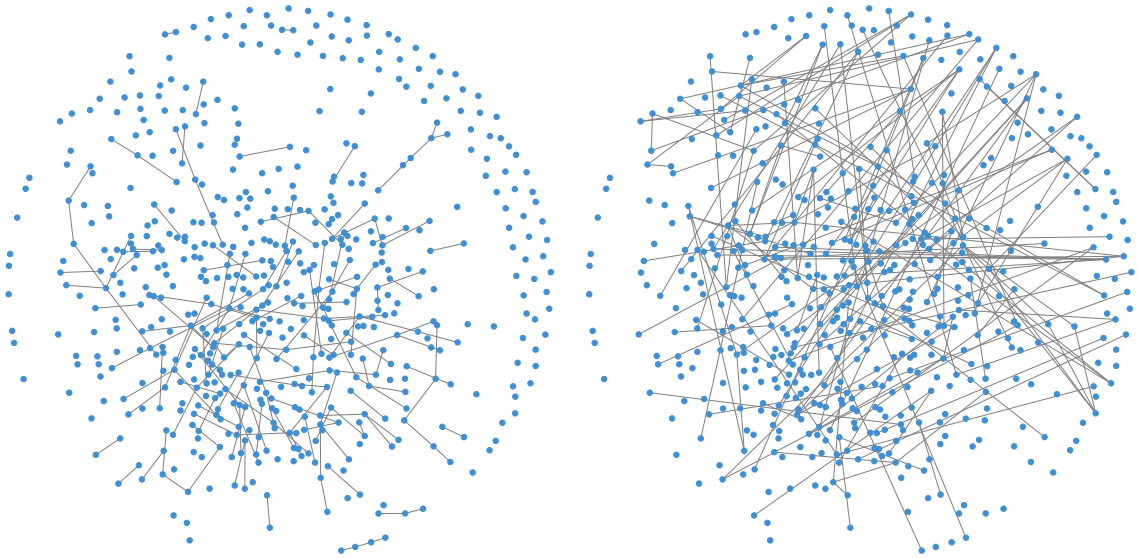
- We now focus on some specific edges and nodes in the graphs. The 11 vertices represented by smaller squares and shaded in green are not connected to any other vertex in gL. On the other hand, all these 11 vertices are connected to at least one other vertex in JgL (see Figure 10). Moreover, several of these edges are meaningful. For example, CBS and CMCSA are in the television industry, TGT and CVS are stores, AEP and WMB are energy companies, GD and RTN are defense companies, and MDT and UNH are in the healthcare industry. Finally, the three vertices represented by larger squares and shaded in pink, are not connected to any vertex in JgL and are connected to at least one other vertex in gL. Only the edges associated with EXC seem to be meaningful.

8.3 Analysis of Gene Expression Data

Graphical models have been used extensively for studying gene interactions using gene expression data (Nevins et al., 2004; Wille et al., 2004). The gene expression data we study is the Lymph node status data which contains $n = 148$ expression values from $p = 587$ genes (Li and Toh, 2010). Since there is no ground truth available, the main aim in this section is to highlight the differences between the estimates JgL (junction tree estimate) and gL (non junction tree estimate). Just like in the stock returns data, we compute the graph H using $\kappa = 1$. Both the JgL and gL graphs contain 831 edges. Figure 12 shows the graphs JgL and gL under different placements of the vertices. We clearly see significant differences between the estimated graphs. This suggests that using the junction tree framework may lead to new scientific interpretations when studying biological data.

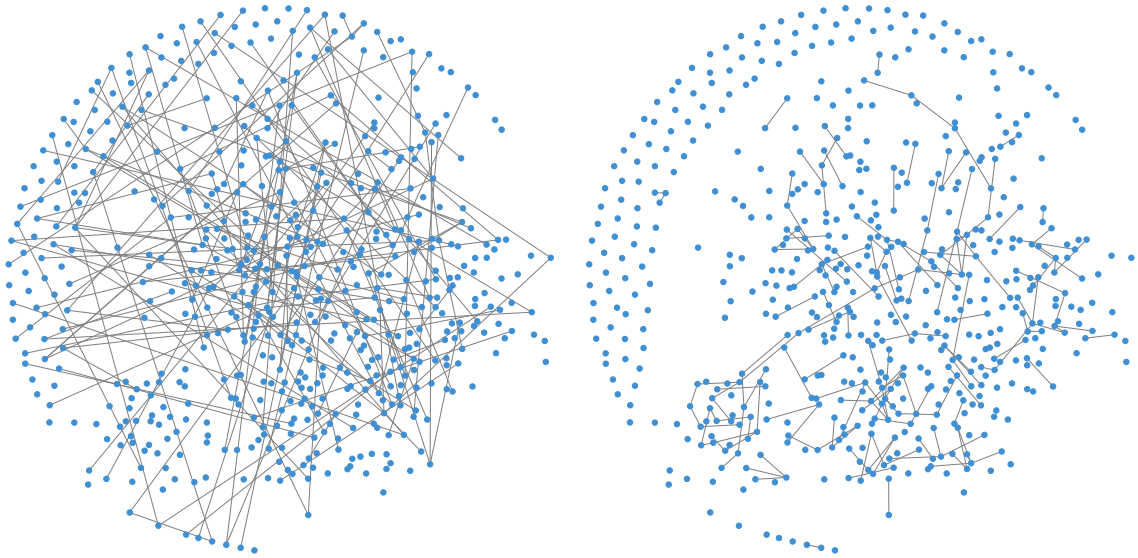
9. Summary and Future Work

We have outlined a general framework that can be used as a wrapper around any arbitrary undirected graphical model selection (UGMS) algorithm for improved graph estimation. Our framework takes as input a graph H that contains all (or most of) the edges in G^*, decomposes the UGMS problem into multiple subproblems using a junction tree representation of H, and then solves subprolems iteratively to estimate a graph. Our theoretical results show that certain weak edges, which cannot be estimated using standard algorithms, can be estimated when using the junction tree framework. We supported the



(a) Junction tree based graphical Lasso

(b) Graphical Lasso



(c) Junction tree based graphical Lasso

(d) Graphical Lasso

Figure 12: Graph over genes computed using gene expression data. For (a) and (b), the vertices are chosen so that the junction tree estimate is aesthetically pleasing. For (c) and (d), the vertices are chosen so that the graphical Lasso estimate is aesthetically pleasing. Further, in (a) and (c), we only show edges that are estimated in the junction tree estimate, but not estimated using graphical Lasso. Similarly, for (b) and (c), we only show edges that are estimated by graphical Lasso, but not by the junction tree estimate.

theory with numerical simulations on both synthetic and real world data. All the data and code used in our numerical simulations can be found at `http://www.ima.umn.edu/~dvats/JunctionTreeUGMS.html`.

Our work motivates several interesting future research directions. In our framework, we used a graph H to decompose the UGMS problem into multiple subproblems. Alternatively, we can also focus on directly finding such decompositions. Another interesting research direction is to use the decompositions to develop parallel algorithms for UGMS for estimating extremely large graphs. Finally, motivated by the differences in the graphs obtained using gene expression data, another research problem of interest is to study the scientific consequences of using the junction tree framework on various computational biology data sets.

Acknowledgments

The first author thanks the Institute for Mathematics and its Applications (IMA) for financial support in the form of a postdoctoral fellowship. The authors thank Vincent Tan for discussions and comments on an earlier version of the paper. The authors thank the anonymous reviewers for comments which significantly improved this manuscript.

Appendix A. Marginal Graph

Definition 23 *The marginal graph $G^{*,m}[A]$ of a graph G^* over the nodes A is defined as a graph with the following properties*

1. *$E(G^*[A]) \subseteq E(G^{*,m}[A])$.*
2. *For an edge $(i,j) \in E(K_A) \backslash E(G^*[A])$, if all paths from i to j in G^* pass through a subset of the nodes in A, then $(i,j) \notin G^{*,m}[A]$.*
3. *For an edge $(i,j) \in E(K_A) \backslash E(G^*[A])$, if there exists a path from i to j in G^* such that all nodes in the path, except i and j, are in $V \backslash A$, then $(i,j) \in G^{*,m}[A]$.*

The graph K_A is the complete graph over the vertices A. The first condition in Definition 23 says that the marginal graph contains all edges in the induced subgraph over A. The second and third conditions say which edges not in $G^*[A]$ are in the marginal graph. As an example, consider the graph in Figure 13(a) and let $A = \{1,2,3,4,5\}$. From the second condition, the edge $(3,4)$ is not in the marginal graph since all paths from 3 to 4 pass through a subset of the nodes in A. From the third condition, the edge $(4,5)$ is in the marginal graph since there exists a path $\{4,8,5\}$ that does not go through any nodes in $A \backslash \{4,5\}$. Similarly, the marginal graph over $A = \{4,5,6,7,8\}$ can be constructed as in Figure 13(c). The importance of marginal graphs is highlighted in the following proposition.

Proposition 24 *If $P_X > 0$ is Markov on $G^* = (V, E(G^*))$ and not Markov on any subgraph of G^*, then for any subset of vertices $A \subseteq V$, P_{X_A} is Markov on the marginal graph $G^{*,m}[A]$ and not Markov on any subgraph of $G^{*,m}[A]$.*

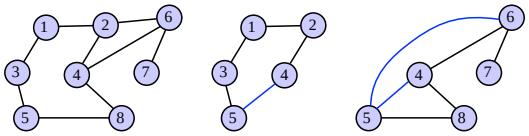


Figure 13: (a) A graph over eight nodes. (b) The marginal graph over $\{1, 2, 3, 4, 5\}$. (c) The marginal graph over $\{4, 5, 6, 7, 8\}$.

Proof Suppose P_{X_A} is Markov on the graph $\check{G}_A$ and not Markov on any subgraph of $\check{G}_A$. We will show that $\check{G}_A = G^m[A]$.

- If $(i, j) \in G$, then $X_i \not\perp\!\!\!\perp X_j | X_S$ for every $S \subseteq V \backslash \{i, j\}$. Thus, $G[A] \subset \check{G}_A$.
- For any edge $(i, j) \in K_A \backslash G[A]$, suppose that for every path from i to j contains at least one node from $A \backslash \{i, j\}$. Then, there exists a set of nodes $S \subseteq A \backslash \{i, j\}$ such that $X_i \perp\!\!\!\perp X_j | X_S$ and $(i, j) \notin \check{G}_A$.
- For any edge $(i, j) \in K_A \backslash G[A]$, suppose that there exists a path from i to j such that all nodes in the path, except i and j, are in $V \backslash A$. This means we cannot find a separator for i and j in the set A, so $(i, j) \in \check{G}_A$.

From the construction of $\check{G}_A$ and Definition 23, it is clear that $\check{G}_A = G^m[A]$. ■

Using Proposition 24, it is clear that if the UGMS algorithm Ψ in Assumption 1 is applied to a subset of vertices A, the output will be a consistent estimator of the marginal graph $G^{*,m}[A]$. Note that from Definition 23, although the marginal graph contains all edges in $G^*[A]$, it may contain additional edges as well. Given only the marginal graph $G^{*,m}[A]$, it is not clear how to identify edges that are in $G^*[A]$. For example, suppose G^* is a graph over four nodes and let the graph be a single cycle. The marginal graph over any subset of three nodes is always the complete graph. Given the complete graph over three nodes, computing the induced subgraph over the three nodes is nontrivial.

Appendix B. Examples of UGMS Algorithms

We give examples of standard UGMS algorithms and show how they can be used to implement step 3 in Algorithm 2 when estimating edges in a region of a region graph. For simplicity, we review algorithms for UGMS when P_X is a Gaussian distribution with mean zero and covariance Σ^*. Such distributions are referred to as Gaussian graphical models. It is well known (Speed and Kiiveri, 1986) that that the inverse covariance matrix $\Theta^* = (\Sigma^*)^{-1}$, also known as precision matrix, is such that for all $i \neq j$, $\Theta^*_{ij} \neq 0$ if and only if $(i, j) \in E(G^*)$. In other words, the graph G^* can be estimated given an estimate of the covariance or inverse covariance matrix of X. We review two standard algorithms for estimating G^*: graphical Lasso and neighborhood selection using Lasso (nLasso).

B.1 Graphical Lasso (gLasso)

Define the empirical covariance matrix $\widehat{S}_A$ over a set of vertices $A \subset V$ as follows:

$$\widehat{S}_A = \frac{1}{n}\sum_{k=1}^{n} X_A^{(k)} \left(X_A^{(k)}\right)^T .$$

Recall from Algorithm 2, we apply a UGMS algorithm $\overline{R}$ to estimate edges in H'_R defined in (3). The graphical Lasso (gLasso) estimates $\widehat{E}_R$ by solving the following convex optimization problem:

$$\widehat{\Theta} = \arg \max_{\Theta \succ 0, \Theta_{ij}=0 \; \forall \; (i,j) \notin H^m\left[\overline{R}\right]} \left\{ \log\det(\Theta) - \text{trace}\left(\widehat{S}_{\overline{R}}\,\Theta\right) - \lambda \sum_{(i,j)\in H'_R} \Theta_{ij} \right\}, \quad (9)$$

$$\widehat{E}_R = \left\{(i,j) \in H'_A : \widehat{\Theta}_{ij} \neq 0\right\}.$$

The graph $H^m[\overline{R}]$ is the marginal graph over $\overline{R}$ (see Appendix A). When $\overline{R} = V$, $H = K_V$, and $H'_A = K_V$, the above equations recover the standard gLasso estimator, which was first proposed in Banerjee et al. (2008). Equation (9) can be solved using algorithms in Yuan and Lin (2007), Banerjee et al. (2008), Scheinberg et al. (2010) and Hsieh et al. (2011). Theoretical properties of the estimates $\widehat{\Theta}$ and $\widehat{E}_R$ have been studied in Ravikumar et al. (2011). Note that the regularization parameter in (9) controls the sparsity of $\widehat{E}_R$. A larger λ corresponds to a sparser solution. Further, we only regularize the terms in Θ_{ij} corresponding to the edges that need to be estimated, that is, the edges in H'_R. Finally, Equation (9) also accounts for the edges H by computing the marginal graph over $\overline{R}$. In general, $H^m\left[\overline{R}\right]$ can be replaced by any graph that is superset of $H^m\left[\overline{R}\right]$.

B.2 Neighborhood Selection (nLasso)

Using the local Markov property of undirected graphical models (see Definition 1), we know that if P_X is Markov on G^*, then $P\left(X_i \,|\, X_{V\setminus i}\right) = P\left(X_i \,|\, X_{ne_{G^*}(i)}\right)$. This motivates an algorithm for estimating the neighborhood of each node and then combining all these estimates to estimate G^*. For Gaussian graphical models, this can be achieved by solving a Lasso problem (Tibshirani, 1996) at each node (Meinshausen and Bühlmann, 2006). Recall that we are interested in estimating all edges in H'_R by applying a UGMS algorithm to $\overline{R}$. The neighborhood selection using Lasso (nLasso) algorithm is given as follows:

$$H'' = K_{\overline{R}} \backslash H^m\left[\overline{R}\right],$$

$$\widehat{\beta}^k = \arg \min_{\beta_i = 0, i \in ne_{H''}(k) \cup k \cup V\setminus A} \left\{ \|\mathfrak{X}_k^n - \mathfrak{X}^n \beta\|_2^2 + \lambda \sum_{i \in ne_{H'_R}(k)} |\beta_i| \right\}, \quad (10)$$

$$\widehat{ne}^k = \left\{ i : \widehat{\beta}_i^k \neq 0 \right\},$$

$$\widehat{E}_R = \bigcup_{k \in \overline{R}} \left\{ (k,i) : i \in \widehat{ne}^k \right\}.$$

Notice that in the above algorithm if i is estimated to be a neighbor of j, then we include the edge (i,j) even if j is not estimated to be a neighbor of i. This is called the union rule for combining neighborhood estimates. In our numerical simulations, we use the intersection rule to combine neighborhood estimates, that is, (i,j) is estimated only if i is estimated to be a neighbor of j and j is estimated to be a neighbor of i. Theoretical analysis of nLasso has been carried out in Meinshausen and Bühlmann (2006) and Wainwright (2009). Note that, when estimating the neighbors of a node k, we only penalize the neighbors in H'_R. Further, we use prior knowledge about some of the edges by using the graph H in (10). References Bresler et al. (2008), Netrapalli et al. (2010) and Ravikumar et al. (2010) extend the neighborhood selection based method to discrete valued graphical models.

Appendix C. Proof of Proposition 8

We first prove the following result.

Lemma 25 *For any $(i,j) \in H'_R$, there either exists no non-direct path from i to j in H or all non-direct paths in H pass through a subset of $\overline{R}$.*

Proof We first show the result for $R \in \mathcal{R}^1$. This means that R is one of the clusters in the junction tree used to construct the region graph and $ch(R)$ is the set of all separators of cardinality greater than one connected to the cluster R in the junction tree. Subsequently, $\overline{R} = R$. If $ch(R) = \emptyset$, the claim trivially holds. Let $ch(R) \neq \emptyset$ and suppose there exists a non-direct path from i to j that passes through a set of vertices $\bar{S}$ not in $\overline{R}$. Then, there will exist a separator S in the junction tree such that S separates $\{i,j\}$ and $\bar{S}$. Thus, all paths in H from i and j to $\bar{S}$ pass through S. This implies that either there is no non-direct path from i to j in H or else we have reached a contradiction about the existence of a non-direct path from i to j that passes through the set $\bar{S}$ not in $\overline{R}$.

Now, suppose $R \in \mathcal{R}^l$ for $l > 1$. The set $an(R)$ contains all the clusters in the junction tree than contain R. From the running intersection property of junction trees, all these clusters must form a subtree in the original junction tree. Merge $\overline{R}$ into one cluster and find a new junction tree $\mathcal{J}'$ by keeping the rest of the clusters the same. It is clear $\overline{R}$ will be in the first row of the updated region graph. The arguments used above can be repeated to prove the claim. ■

We now prove Proposition 8.

Case 1: Let $(i,j) \in H'_R$ and $(i,j) \notin G^*$. If there exists no non-direct path from i to j in H, then the edge (i,j) can be estimated by solving a UGMS problem over i and j. By the definition of $\overline{R}$, $i,j \in \overline{R}$. Suppose there does exist non-direct paths from i to j in H. From Lemma 25, all such paths pass through $\overline{R}$. Thus, the conditional independence of X_i and X_j can be determined from $X_{\overline{R}\setminus\{i,j\}}$.

Case 2: Let $(i,j) \in H'_R$ and $(i,j) \in G^*$. From Lemma 25 and using the fact that $E(G^*) \subseteq E(H)$, we know that all paths from i to j pass through $\overline{R}$. This means that if $X_i \not\perp\!\!\!\perp X_j | X_{\overline{R}\setminus\{i,j\}}$, then $X_i \not\perp\!\!\!\perp X_j | X_{V\setminus\{i,j\}}$.

Appendix D. Analysis of the PC-Algorithm in Algorithm 4

In this section, we present the analysis of Algorithm 4 using results from Anandkumar et al. (2012a) and Kalisch and Bühlmann (2007). The analysis presented here is for the non-junction tree based algorithm. Throughout this section, assume

$$\widehat{G} = \mathsf{PC}(\eta, \mathfrak{X}^n, K_V, K_V)\,,$$

where K_V is the complete graph over the vertices V. Further, let the threshold for the conditional independence test in (4) be λ_n. We are interested in finding conditions under which $\widehat{G} = G^*$ with high probability.

Theorem 26 *Under Assumptions (A1)-(A5), there exists a conditional independence test such that if*

$$n = \Omega(\rho_{min}^{-2}\eta\log(p)) \text{ or } \rho_{min} = \Omega(\sqrt{\eta\log(p)/n}),$$

then $P(\widehat{G} \neq G) \to 0$ *as* $n \to \infty$.

We now prove Theorem 26. Define the set B_η as follows:

$$B_\eta = \{(i,j,S) : i,j \in V, i \neq j, S \subseteq V\backslash\{i,j\}, |S| \leq \eta\}\,.$$

The following concentration inequality follows from Anandkumar et al. (2012a).

Lemma 27 *Under Assumption (A4), there exists constants* c_1 *and* c_2 *such that for* $\epsilon < M$,

$$\sup_{(i,j,S)\in B_\eta} P\left(||\rho_{ij|S}| - |\widehat{\rho}_{ij|S}|| > \xi\right) \leq c_1 \exp\left(-c_2(n-\eta)\xi^2\right)\,,$$

where n *is the number of vector valued measurements made of* X_i, X_j, *and* X_S.

Let $P_e = P(\widehat{G} \neq G)$, where the probability measure P is with respect to P_X. Recall that we threshold the empirical conditional partial correlation $\widehat{\rho}_{ij|S}$ to test for conditional independence, that is, $\widehat{\rho}_{ij|S} \leq \lambda_n \implies X_i \perp\!\!\!\perp X_j | X_S$. An error may occur if there exists two distinct vertices i and j such that either $\rho_{ij|S} = 0$ and $|\widehat{\rho}_{ij|S}| > \lambda_n$ or $|\rho_{ij|S}| > 0$ and $|\widehat{\rho}_{ij|S}| \leq \lambda_n$. Thus, we have

$$\begin{aligned}
P_e &\leq P(\mathcal{E}_1) + P(\mathcal{E}_2)\,,\\
P(\mathcal{E}_1) &= P\left(\bigcup_{(i,j)\notin G} \{\exists\ S \text{ s.t. } |\widehat{\rho}_{ij|S}| > \lambda_n\}\right)\,,\\
P(\mathcal{E}_2) &= P\left(\bigcup_{(i,j)\in G} \{\exists\ S \text{ s.t. } |\widehat{\rho}_{ij|S}| \leq \lambda_n\}\right)\,.
\end{aligned}$$

We will find conditions under which $P(\mathcal{E}_1) \to 0$ and $P(\mathcal{E}_2) \to 0$ which will imply that $P_e \to 0$. The term $P(\mathcal{E}_1)$, the probability of including an edge in $\widehat{G}$ that does not belong to

the true graph, can be upper bounded as follows:

$$\begin{aligned}
P(\mathcal{E}_1) &\leq P\left(\bigcup_{(i,j)\notin G}\{\exists\ S \text{ s.t. } |\widehat{\rho}_{ij|S}| > \lambda_n\}\right) \leq P\left(\bigcup_{(i,j)\notin G, S\subset V\setminus\{i,j\}}\{|\widehat{\rho}_{ij|S}| > \lambda_n\}\right), \\
&\leq p^{\eta+2}\sup_{(i,j,S)\in B_\eta} P\left(|\widehat{\rho}_{ij|S}| > \lambda_n\right), \\
&\leq c_1 p^{\eta+2}\exp\left(-c_2(n-\eta)\lambda_n^2\right) = c_1\exp\left((\eta+2)\log(p) - c_2(n-\eta)\lambda_n^2\right).
\end{aligned}$$

The terms $p^{\eta+2}$ comes from the fact that there are at most p^2 number of edges and the algorithm searches over at most p^η number of separators for each edge. Choosing λ_n such that

$$\lim_{n,p\to\infty}\frac{(n-\eta)\lambda_n^2}{(\eta+2)\log(p)} = \infty, \tag{11}$$

ensures that $P(\mathcal{E}_1)\to 0$ as $n,p\to\infty$. Further, choose λ_n such that for $c_3 < 1$

$$\lambda_n < c_3\rho_{min}. \tag{12}$$

The term $P(\mathcal{E}_2)$, the probability of not including an edge in $\widehat{G}$ that does belong to the true graph, can be upper bounded as follows:

$$\begin{aligned}
P(\mathcal{E}_2) &\leq P\left(\bigcup_{(i,j)\in G}\{\exists\ S \text{ s.t. } |\widehat{\rho}_{ij|S}| \leq \lambda_n\}\right), \\
&\leq P\left(\bigcup_{(i,j)\in G, S\subset V\setminus\{i,j\}} |\rho_{ij|S}| - |\widehat{\rho}_{ij|S}| > |\rho_{ij|S}| - \lambda_n\right), \\
&\leq p^{\eta+2}\sup_{(i,j,S)\in B_\eta} P\left(|\rho_{ij|S}| - |\widehat{\rho}_{ij|S}| > |\rho_{ij|S}| - \lambda_n\right), \\
&\leq p^{\eta+2}\sup_{(i,j,S)\in B_\eta} P\left(||\rho_{ij|S}| - |\widehat{\rho}_{ij|S}|| > \rho_{min} - \lambda_n\right), \\
&\leq c_1 p^{\eta+2}\exp\left(-c_2(n-\eta)(\rho_{min}-\lambda_n)^2\right) = c_1\exp\left((\eta+2)\log(p) - c_4(n-\eta)\rho_{min}^2\right).
\end{aligned} \tag{13}$$

To get (13), we use (12) so that $(\rho_{min}-\lambda_n) > (1-c_3)\rho_{min}$. For some constant $c_5 > 0$, suppose that for all $n > n'$ and $p > p'$,

$$c_4(n-\eta)\rho_{min}^2 > (\eta+2+c_5)\log(p). \tag{14}$$

Given (14), $P(\mathcal{E}_2)\to 0$ as $n,p\to\infty$. In asymptotic notation, we can write (14) as

$$n = \Omega(\rho_{min}^{-2}\eta\log(p)),$$

which proves the Theorem. The conditional independence test is such that λ_n is chosen to satisfy (11) and (12). In asymptotic notation, we can show that $\lambda_n = O(\rho_{min})$ and $\lambda_n^2 = \Omega\left(\eta\log(p)/n\right)$ satisfies (11) and (12).

Appendix E. Proof of Theorem 9

To prove the theorem, it is sufficient to establish that

$$\rho_0 = \Omega\left(\sqrt{\eta_T \log(p)/n}\right), \tag{15}$$

$$\rho_1 = \Omega\left(\sqrt{\eta \log(p_1)/n}\right), \tag{16}$$

$$\rho_2 = \Omega\left(\sqrt{\eta \log(p_2)/n}\right), \tag{17}$$

$$\rho_T = \Omega\left(\sqrt{\eta \log(p_T)/n}\right). \tag{18}$$

Let H be the graph estimated in Step 1. An error occurs if for an edge $(i,j) \in G^*$ there exists a subset of vertices S such that $|S| \leq \eta_T$ and $|\widehat{\rho}_{ij|S}| \leq \lambda_n^0$. Using the proof of Theorem 26 (see analysis of $P(\mathcal{E}_2)$), it is easy to see that $n = \Omega(\rho_0^{-2}\eta_T \log(p))$ is sufficient for $P(E(G^*) \not\subset E(H)) \to 0$ as $n \to 0$. Further, the threshold is chosen such that $\lambda_n^0 = O(\rho_0)$ and $(\lambda_n^0)^2 = \Omega\left(\eta_T \log(p)/n\right)$. This proves (15).

In Step 2, we estimate the graphs $\widehat{G}_1$ and $\widehat{G}_2$ by applying the PC-Algorithm to the vertices $V_1 \cup T$ and $V_2 \cup T$, respectively. For $\widehat{G}_1$, given that all edges that have a separator of size η_T have been removed, we can again use the analysis in the proof of Theorem 26 to show that for $\lambda_n^1 = O(\rho_1)$ and $(\lambda_n^1)^2 = \Omega\left(\eta \log(p_1)/n\right)$, $n = \Omega(\rho_1^{-2}\eta \log(p_1))$ is sufficient for $P(\widehat{G}_1 \neq G^*[V_1 \cup T] \backslash K_T) | G^* \subset H) \to 0$ as $n \to \infty$. This proves (16). Using similar analysis, we can prove (17) and (18).

The probability of error can be written as

$$\begin{aligned} P_e \leq P(G^* \not\subset H) + \sum_{k=1}^{2} P(\widehat{G}_k \neq G^*[V_k \cup T] \backslash K_T | G^* \subset H) \\ + P(\widehat{G}_T \neq G^*[T] | G^* \subset H, \widehat{G} = G[V_1 \cup T]^* \backslash K_T, G^*[V_2 \cup T] = G[V_2 \cup T] \backslash K_T). \end{aligned}$$

Given (15)-(18), each term on the right goes to 0 as $n \to \infty$, so $P_e \to 0$ as $n \to \infty$.

References

A. Anandkumar, V. Y. F. Tan, F. Huang, and A. S. Willsky. High-dimensional Gaussian graphical model selection: Walk summability and local separation criterion. *Journal of Machine Learning Research*, 13:2293–2337, 2012a.

A. Anandkumar, V. Y. F. Tan, F. Huang, and A. S. Willsky. High-dimensional structure learning of Ising models: Local separation criterion. *Annals of Statistics*, 40(3):1346–1375, 2012b.

S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in ak-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11–24, April 1989.

F. R. Bach and M. I. Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems (NIPS)*, pages 569–576. MIT Press, 2001.

O. Banerjee, L. E. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, June 2008.

A. Berry, P. Heggernes, and G. Simonet. The minimum degree heuristic and the minimal triangulation process. In *Graph-Theoretic Concepts in Computer Science*, pages 58–70. Springer, 2003.

G. Bresler, E. Mossel, and A. Sly. Reconstruction of Markov random fields from samples: Some observations and algorithms. In Ashish Goel, Klaus Jansen, Jos Rolim, and Ronitt Rubinfeld, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, volume 5171 of *Lecture Notes in Computer Science*, pages 343–356. Springer Berlin, 2008.

F. Bromberg, D. Margaritis, and V. Honavar. Efficient Markov network structure discovery using independence tests. *Journal of Artificial Intelligence Research (JAIR)*, 35:449–484, 2009.

T. Cai, W. Liu, and X. Luo. A constrained ℓ_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011.

V. Chandrasekaran, P.A. Parrilo, and A.S. Willsky. Latent variable graphical model selection via convex optimization. *Annals of Statistics*, 40(4):1935–1967, 2012.

A. Chechetka and C. Guestrin. Efficient principled learning of thin junction trees. In *Advances in Neural Information Processing Systems (NIPS)*, pages 273–280, December 2007.

J. Chen and Z. Chen. Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3):759–771, 2008.

M. J. Choi, V. Y. F. Tan, A. Anandkumar, and A. S. Willsky. Learning latent tree graphical models. *Journal of Machine Learning Research*, 12:1771–1812, May 2011.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, May 1968.

J. Fan and J. Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(5):849–911, 2008.

R. Foygel and M. Drton. Extended Bayesian information criteria for Gaussian graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 604–612, 2010.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical Lasso. *Biostatistics*, 9(3):432–441, July 2008.

P. Giudici and P. J. Green. Decomposable graphical gaussian model determination. *Biometrika*, 86(4):785–801, 1999.

W. Hoeffding. A non-parametric test of independence. *The Annals of Mathematical Statistics*, 19(4):546–557, 1948.

C. Hsieh, M. A. Sustik, I. S. Dhillon, and P. Ravikumar. Sparse inverse covariance matrix estimation using quadratic approximation. In *Advances in Neural Information Processing Systems 24*, pages 2330–2338, 2011.

A. Jalali, C. Johnson, and P. Ravikumar. On learning discrete graphical models using greedy methods. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1935–1943, 2011.

C. C. Johnson, A. Jalali, and P. Ravikumar. High-dimensional sparse inverse covariance estimation using greedy methods. *Journal of Machine Learning Research - Proceedings Track*, 22:574–582, 2012.

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. Karger and N. Srebro. Learning Markov networks: Maximum bounded tree-width graphs. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 392–401, 2001.

U. B. Kjaerulff. Triangulation of graphs - algorithms giving small total state space. Technical Report Research Report R-90-09, Department of Mathematics and Computer Science, Aalborg University, Denmark, 1990.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

K. S. S. Kumar and F. Bach. Convex relaxations for learning bounded-treewidth decomposable graphs. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

J. Lafferty, H. Liu, and L. Wasserman. Sparse nonparametric graphical models. *Statistical Science*, 27(4):519–537, 2012.

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 (Methodological)*, 50(2):157–224, 1988.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, USA, 1996.

L. Li and K. C. Toh. An inexact interior point method for ℓ_1-regularized sparse covariance selection. *Mathematical Programming Computation*, 2(3):291–315, 2010.

H. Liu, K. Roeder, and L. Wasserman. Stability approach to regularization selection (stars) for high dimensional graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

H. Liu, F. Han, M. Yuan, J. Lafferty, and L. Wasserman. High dimensional semiparametric Gaussian copula graphical models. *Annals of Statistics*, 40(4):2293–2326, 2012a.

H. Liu, J. Lafferty, and L. Wasserman. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *The Journal of Machine Learning Research*, 10: 2295–2328, 2009.

H. Liu, F. Han, and C. Zhang. Transelliptical graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 809–817, 2012b.

P. Loh and M. J. Wainwright. Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2096–2104, 2012.

Z. Ma, X. Xie, and Z. Geng. Structural learning of chain graphs via decomposition. *Journal of Machine Learning Research*, 9:2847–2880, December 2008.

D. M. Malioutov, J. K. Johnson, and A. S. Willsky. Walk-sums and belief propagation in gaussian graphical models. *The Journal of Machine Learning Research*, 7:2031–2064, 2006.

F. M. Malvestuto. Approximating discrete probability distributions with decomposable models. *IEEE Transactions on Systems, Man and Cybernetics*, 21(5):1287–1294, 1991.

R. Mazumder and T. Hastie. Exact covariance thresholding into connected components for large-scale graphical lasso. *Journal of Machine Learning Research*, 13:781–794, March 2012. ISSN 1532-4435.

N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

P. Netrapalli, S. Banerjee, S. Sanghavi, and S. Shakkottai. Greedy learning of Markov network structure. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1295–1302, 2010.

A. Dobra, C. Hans, B. Jones, J. R. Nevins, and G. Yao, and M. West. Sparse graphical models for exploring gene expression data. *Journal of Multivariate Analysis*, 90:196–212, 2004.

M. J Rasch, A. Gretton, Y. Murayama, W. Maass, N. K Logothetis, L. Wiskott, G. Kempermann, L. Wiskott, G. Kempermann, B. Schölkopf, et al. A kernel two-sample test. *Journal of Machine Learning Research*, 2:299, 2012.

P. Ravikumar, M. J. Wainwright, and J. Lafferty. High-dimensional Ising model selection using ℓ_1-regularized logistic regression. *Annals of Statistics*, 38(3):1287–1319, 2010.

P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu. High-dimensional covariance estimation by minimizing ℓ_1-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011.

N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309 – 322, 1986.

K. Scheinberg, S. Ma, and D. Goldfarb. Sparse inverse covariance selection via alternating linearization methods. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2101–2109, 2010.

T. P. Speed and H. T. Kiiveri. Gaussian Markov distributions over finite graphs. *The Annals of Statistics*, 14(1):138–150, 1986.

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. Causality from probability. In *Advanced Computing for the Social Sciences*, 1990.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

S. van de Geer, P. Bühlmann, and S. Zhou. The adaptive and the thresholded Lasso for potentially misspecified models (and a lower bound for the lasso). *Electronic Journal of Statistics*, 5:688–749, 2011.

D. Vats. High-dimensional screening using multiple grouping of variables. *IEEE Transactions On Signal Processing*, to appear.

D. Vats and J. M. F. Moura. Finding non-overlapping clusters for generalized inference over graphical models. *IEEE Transactions on Signal Processing*, 60(12):6368 –6381, Dec. 2012.

M. J. Wainwright. *Stochastic Processes on Graphs: Geometric and Variational Approaches*. PhD thesis, Department of EECS, Massachusetts Institute of Technology, 2002.

M. J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1-constrained quadratic programming (Lasso). *IEEE Transactions on Information Theory*, 55(5):2183–2202, 2009. ISSN 0018-9448.

W. Wang, M. J. Wainwright, and K. Ramchandran. Information-theoretic bounds on model selection for Gaussian Markov random fields. In *IEEE International Symposium on Information Theory (ISIT)*, 2010.

A. Wille, P. Zimmermann, E. Vranová, A. Fürholz, O. Laule, S. Bleuler, L. Hennig, A. Prelic, P. Von Rohr, L. Thiele, et al. Sparse graphical Gaussian modeling of the isoprenoid gene network in arabidopsis thaliana. *Genome Biol*, 5(11):R92, 2004.

D. M. Witten, J. H. Friedman, and N. Simon. New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, 20(4):892–900, 2011.

X. Xie and Z. Geng. A recursive method for structural learning of directed acyclic graphs. *Journal of Machine Learning Research*, 9:459–483, 2008.

L. Xue and H. Zou. Regularized rank-based estimation of high-dimensional nonparanormal graphical models. *The Annals of Statistics*, 40(5):2541–2571, 2012.

E. Yang, G. Allen, Z. Liu, and P. Ravikumar. Graphical models via generalized linear models. In *Advances in Neural Information Processing Systems*, pages 1367–1375, 2012.

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.

M. Yuan and Yi Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

K. Zhang, J. Peters, D. Janzing, and B. Schölkopf. Kernel-based conditional independence test and application in causal discovery. *Arxiv preprint arXiv:1202.3775*, 2012.

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

Journal of Machine Learning Research 15 (2014) 193-215 Submitted 7/13; Revised 11/13; Published 1/14

Axioms for Graph Clustering Quality Functions

Twan van Laarhoven TVANLAARHOVEN@CS.RU.NL
Elena Marchiori ELENAM@CS.RU.NL
Institute for Computing and Information Sciences
Radboud University Nijmegen
Postbus 9010
6500 GL Nijmegen, The Netherlands

Editor: Vahab Mirrokni

Abstract

We investigate properties that intuitively ought to be satisfied by graph clustering quality functions, that is, functions that assign a score to a clustering of a graph. Graph clustering, also known as network community detection, is often performed by optimizing such a function. Two axioms tailored for graph clustering quality functions are introduced, and the four axioms introduced in previous work on distance based clustering are reformulated and generalized for the graph setting. We show that modularity, a standard quality function for graph clustering, does not satisfy all of these six properties. This motivates the derivation of a new family of quality functions, adaptive scale modularity, which does satisfy the proposed axioms. Adaptive scale modularity has two parameters, which give greater flexibility in the kinds of clusterings that can be found. Standard graph clustering quality functions, such as normalized cut and unnormalized cut, are obtained as special cases of adaptive scale modularity.

In general, the results of our investigation indicate that the considered axiomatic framework covers existing 'good' quality functions for graph clustering, and can be used to derive an interesting new family of quality functions.

Keywords: graph clustering, modularity, axiomatic framework

1. Introduction

Following the work by Kleinberg (2002) there have been various contributions to the theoretical foundation and analysis of clustering, such as axiomatic frameworks for quality functions (Ackerman and Ben-David, 2008), for criteria to compare clusterings (Meila, 2005), uniqueness theorems for specific types of clustering (Zadeh and Ben-David, 2009; Ackerman and Ben-David, 2013; Carlsson, Mémoli, Ribeiro, and Segarra, 2013), taxonomy of clustering paradigms (Ackerman et al., 2010a), and characterization of diversification systems (Gollapudi and Sharma, 2009).

Kleinberg focused on clustering functions, which are functions from a distance function to a clustering. He showed that there are no clustering functions that simultaneously satisfy three intuitive properties: scale invariance, consistency and richness. Ackerman and Ben-David (2008) continued on this work, and showed that the impossibility result does not apply when formulating these properties in terms of quality functions instead of clustering functions, where consistency is replaced with a weaker property called monotonicity.

Both of these previous works are formulated in terms of distance functions over a fixed domain. In this paper we focus on weighted graphs, where the weight of an edge indicates the strength of a connection. The clustering problem on graphs is also known as network community detection.

Graphs provide additional freedoms over distance functions. In particular, it is possible for two points to be unrelated, indicated by a weight of 0. These zero-weight edges in turn make it natural to consider graphs over different sets of nodes as part of a larger graph. Secondly, we can allow for self loops. Self loops can indicate internal edges in a node. This notation is used for instance by Blondel et al. (2008), where a graph is contracted based on a fine-grained clustering.

In this setting, where edges with weight 0 are possible, Kleinberg's impossibility result does not apply. This can be seen by considering the connected components of a graph. This is a graph clustering function that satisfies all three of Kleinberg's axioms: scale invariance, consistency and richness (see Section 4.2).

Our focus is on the investigation of graph clustering quality functions, which are functions from a graph and a clustering to a real number 'quality'. A notable example is modularity (Newman and Girvan, 2004). In particular we ask which properties of quality functions intuitively ought to hold, and which are often assumed to hold when reasoning informally about graph clustering. Such properties might be called axioms for graph clustering.

The rest of this paper is organized as follows: Section 2 gives basic definitions. Next, section 3 discusses different ways in which properties could be formulated.

In Section 4 of this paper we propose an axiomatic framework that consists of six properties of graph clustering quality functions: the (adaption of) the four axioms from Kleinberg (2002) and Ackerman and Ben-David (2008) (permutation invariance, scale invariance, richness and monotonicity); and two additional properties specific for the graph setting (continuity and the locality).

Then, in Section 5, we show that modularity does not satisfy the monotonicity and locality properties.

This result motivates the analysis of variants of modularity, leading to the derivation of a new parametric quality function in Section 6, that satisfies all properties. This quality function, which we call adaptive scale modularity, has two parameters, M and γ which can be tuned to control the resolution of the clustering. We show that quality functions similar to normalized cut and unnormalized cut are obtained in the limit when M goes to zero and to infinity, respectively. Furthermore, setting γ to 0 yields a parametric quality function similar to that proposed by Reichardt and Bornholdt (2004).

1.1 Related Work

Previous axiomatic studies of clustering quality functions have focused mainly on hierarchical clustering and on weakest and strongest link style quality functions (Kleinberg, 2002; Ackerman and Ben-David, 2008; Zadeh and Ben-David, 2009; Carlsson et al., 2013). Papers in this line of work that focussed also on the partitional setting include Puzicha et al. (1999), Ackerman et al. (2012) and Ackerman et al. (2013). Puzicha et al. (1999) investigated a particular class of clustering quality functions obtained by requiring the function to

decompose into a certain additive form. Ackerman et al. (2012) considered clustering in the weighted setting, in which every data point is assigned a real valued weight. They performed a theoretical analysis on the influence of weighted data on standard clustering algorithms. Ackerman et al. (2013) analyzed robustness of clustering algorithms to the addition of a small set of points, and investigated the robustness of popular clustering methods.

All these studies are framed in terms of distance (or similarity and dissimilarity) functions.

Bubeck and Luxburg (2009) studied statistical consistency of clustering methods. They introduced the so-called nearest neighbor clustering and showed its consistency also for standard graph based quality functions, such as normalized cut, ratio cut, and modularity. Here we do not focus on properties of methods to optimize clustering quality, but on natural properties that quality functions for graph clustering should satisfy.

Related works on graph clustering quality functions mainly focus on the so-called resolution limit, that is, the tendency of a quality function to prefer either small or large clusters. In particular, Fortunato and Barthélemy (2007) proved that modularity may not detect clusters smaller than a scale which depends on the total size of the network and on the degree of interconnectedness of the clusters. van Laarhoven and Marchiori (2013) showed that the resolution limit is the most important difference between quality functions in graph clustering optimized using local search optimization.

To mitigate the resolution limit phenomenon, the quality function may be extended with a so-called resolution parameter. For example, Reichardt and Bornholdt (2006) proposed a formulation of graph clustering (therein called network community detection) based on principles from statistical mechanics. This interpretation leads to the introduction of a family of quality functions with a parameter that allows to control the clustering resolution. In Section 6.1 we will show that this extension is a special case of adaptive scale modularity.

Traag, Van Dooren, and Nesterov (2011) formalized the notion of resolution-free quality functions, that is, not suffering from the resolution limit, and provided a characterization of this class of quality functions. Their notion is essentially an axiom, and we will discuss the relation to our axioms in Section 4.1.1.

2. Definitions and Notation

A *symmetric weighted graph* is a pair (V, E) of a finite set V of nodes and a function $E : V \times V \to \mathbb{R}_{\geq 0}$ of edge weights, where $E(i, j) = E(j, i)$ for all $i, j \in V$. Edges with larger weights represent stronger connections, so missing edges can get weight 0. Note that this is the opposite of the convention used in distance based clustering. We explicitly allow for self loops, that is, nodes for which $E(i, i) > 0$.

A *clustering* C of a graph $G = (V, E)$ is a partition of its nodes. That is, $\bigcup C = V$ and for all $c_1, c_2 \in C$, $c_1 \cap c_2 \neq \emptyset$ if and only if $c_1 = c_2$. When two nodes i and j are in the same cluster in clustering C, that is, when $i, j \in c$ for some $c \in C$, then we write $i \sim_C j$. Otherwise we write $i \not\sim_C j$.

A clustering C is a *refinement* of a clustering D, written $C \sqsubseteq D$, when for every cluster $c \in C$ there is a cluster $d \in D$ such that $c \subseteq d$.

A *graph clustering quality function* (or objective function) Q is a function from graphs G and clusterings of G to real numbers. We adopt the convention that a higher quality

indicates a 'better' clustering. As a generalization, we will sometimes work with parameterized *families of quality functions*. A single quality function can be seen as a family with no parameters.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs and let $V_a \subseteq V_1 \cap V_2$ be a subset of the common nodes. We say that the graphs *agree on* V_a if $E_1(i,j) = E_2(i,j)$ for all $i, j \in V_a$. We say that the graphs also *agree on the neighborhood of* V_a If

- $E_1(i,j) = E_2(i,j)$ for all $i \in V_a$ and $j \in V_1 \cap V_2$,
- $E_1(i,j) = 0$ for all $i \in V_a$ and $j \in V_1 \setminus V_2$, and
- $E_2(i,j) = 0$ for all $i \in V_a$ and $j \in V_2 \setminus V_1$.

This means that for nodes in V_a the weights and endpoints of incident edges are exactly the same in the two graphs.

3. On the Form of Axioms

There are three different ways to state potential axioms for clustering:

1. As a property of clustering functions, as in Kleinberg (2002). For example, scale invariance of a clustering function $\hat{C}$ would be written as "$\hat{C}(G) = \hat{C}(\alpha G)$, for all graphs G, $\alpha > 0$". I.e. the optimal clustering is invariant under scaling of edge weights.
2. As a property of the values of a quality function Q, as in Ackerman and Ben-David (2008). For example "$Q(G, C) = Q(\alpha G, C)$, for all graphs G, all clustering C of G, and $\alpha > 0$". I.e. the quality is invariant under scaling of edge weights.
3. As a property of the relation between qualities of different clustering, or equivalently, as a property of an ordering of clusterings for a particular graph. For example "$Q(G, C) \geq Q(G, D) \Rightarrow Q(\alpha G, C) \geq Q(\alpha G, D)$".I.e. the 'better than' relation for clusterings is invariant under scaling of edge weights.

The third form is slightly more flexible than the other two. Any quality function that satisfies a property in the second style will also satisfy the corresponding property in the third style, but the converse is not true. Note also that if D is not restricted in a property in the third style, then one can take $\hat{C}(G) = \operatorname{argmax}_C Q(G, C)$ to obtain a clustering function and an axiom in the first style.

Most properties are more easily stated and proved in the second, absolute, style. Therefore, we adopt the second style unless doing so requires us to make specific choices.

4. Axioms for Graph Clustering Quality Functions

Kleinberg defined three axioms for distance based clustering functions. In Ackerman and Ben-David (2008) the authors reformulated these into four axioms for clustering quality functions. These axioms can easily be adapted to the graph setting.

The first property that one expects for graph clustering is that the quality of a clustering depends only on the graph, that is, only on the weight of edges between nodes, not on the identity of nodes. We formalize this in the permutation invariance axiom,

Definition 1 (Permutation invariance) *A graph clustering quality function Q is* permutation invariant *if for all graphs $G = (V, E)$ and all isomorphisms $f : V \to V'$, it is the case that $Q(G, C) = Q(f(G), f(C))$; where f is extended to graphs and clusterings by $f(C) = \{\{f(i) \mid i \in c\} \mid c \in C\}$ and $f((V, E)) = (V', (i, j) \mapsto E(f^{-1}(i), f^{-1}(j)))$.*

The second property, scale invariance, requires that the quality doesn't change when edge weights are scaled uniformly. This is an intuitive axiom when one thinks in terms of units: a graph with edges in "m/s" can be scaled to a graph with edges in "km/h". The quality should not be affected by such a transformation, perhaps up to a change in units.

Ackerman and Ben-David (2008) defined scale invariance by insisting that the quality stays equal when distances are scaled. In contrast, in Puzicha et al. (1999) the quality should scale proportional with the scaling of distances. We generalize both of these previous definitions by only considering the relations between the quality of two clusterings.

Definition 2 (Scale invariance) *A graph clustering quality function Q is* scale invariant *if for all graphs $G = (V, E)$, all clusterings C_1, C_2 of G and all constants $\alpha > 0$, $Q(G, C_1) \leq Q(G, C_2)$ if and only if $Q(\alpha G, C_1) \leq Q(\alpha G, C_2)$. Where $\alpha G = (V, (i, j) \mapsto \alpha E(i, j))$ is a graph with edge weights scaled by a factor α.*

This formulation is flexible enough for single quality functions. However, families of quality functions could have parameters that are also scale dependent. For such families we therefore propose to use as an axiom a more flexible property that also allows the parameters to be scaled,

Definition 3 (Scale invariant family) *A family of quality function Q_P parameterized by $P \in \mathcal{P}$ is* scale invariant *if for all constants $P \in \mathcal{P}$ and $\alpha > 0$ there is a $P' \in \mathcal{P}$ such that for all graphs $G = (V, E)$, and all clusterings C_1, C_2 of G, $Q_P(G, C_1) \leq Q_P(G, C_2)$ if and only if $Q_{P'}(\alpha G, C_1) \leq Q_{P'}(\alpha G, C_2)$.*

Thirdly, we want to rule out trivial quality functions. This is done by requiring richness, that is, that by changing the edge weights any clustering can be made optimal for that quality function.

Definition 4 (Richness) *A graph clustering quality function Q is* rich *if for all sets V and all non-trivial partitions C^* of V, there is a graph $G = (V, E)$ such that C^* is the Q-optimal clustering of V, that is,* $\operatorname{argmax}_C Q(G, C) = C^*$.

The last axiom that Ackerman and Ben-David consider is by far the most interesting. Intuitively, we expect that when the edges within a cluster are strengthened, or when edges between clusters are weakened, that this does not decrease the quality. Formally we call such a change of a graph a consistent improvement,

Definition 5 (Consistent improvement) *Let $G = (V, E)$ be a graph and C a clustering of G. A graph $G' = (V, E')$ is a C-*consistent improvement *of G if for all nodes i and j, $E'(i, j) \geq E(i, j)$ whenever $i \sim_C j$ and $E'(i, j) \leq E(i, j)$ whenever $i \not\sim_C j$.*

We say that a quality function that does not decrease under consistent improvement is monotonic. In previous work this axiom is often called consistency.

Definition 6 (Monotonicity) *A graph clustering quality function Q is* monotonic *if for all graphs G, all clusterings C of G and all C-consistent improvements G' of G it is the case that $Q(G', C) \geq Q(G, C)$.*

4.1 Locality

In the graph setting it also becomes natural to look at combining different graphs. With distance functions this is impossible, since it is not clear what the distance between nodes from the two different sets should be. But for graphs we can take the edge weight between nodes not in both graphs to be zero, which is the case when the graphs agree on the neighborhood of some set.

Consider adding nodes to one side of a large network, then we would not want the clustering on the other side of the network to change if there is no direct connection. For example, if a new protein is discovered in yeast, then the clustering of unrelated proteins in humans should remain the same. Similarly, we can consider any two graphs with disjoint node sets as one larger graph. Then the quality of clusterings of the two original graphs should relate directly to quality on the combined graph.

In general, local changes to a graph should have only local consequences to a clustering. Or in other words, the contribution of a single cluster to the total quality should only depend on nodes in the neighborhood of that cluster.

Definition 7 (Locality) *A graph clustering quality function Q is* local *if for all graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ that agree on a set V_a and its neighborhood, and for all clusterings C_a, D_a of V_a, C_1 of $V_1 \setminus V_a$ and C_2 of $V_2 \setminus V_a$, if $Q(G_1, C_a \cup C_1) \geq Q(G_1, D_a \cup C_1)$ then $Q(G_2, C_a \cup C_2) \geq Q(G_2, D_a \cup C_2)$.*

Any quality function that has a preference for a fixed number of clusters will not be local. On the other hand, a quality function that is written as a sum over clusters, where each summand depends only on properties of nodes and edges in one cluster and not on global properties, is local.

Ackerman et al. (2010b) defined a similar locality property for clustering functions. Their definition differs from ours in three ways. First of all, they looked at k-clustering, where the number of clusters is given and fixed. Secondly, their locality property only implies a consistent clustering when the rest of the graph is removed, corresponding to $V_2 = V_1 \cap V_a$. They do not consider the other direction, where more nodes and edges are added. Finally, their locality property requires only agreement of the overlapping set V_a, not on its neighborhood. That means that clustering functions should also give the same results if edges with one endpoint in V_a are removed.

4.1.1 Relation to Resolution-Limit-Free Quality Functions

Traag et al. (2011) introduced the notion of *resolution-limit-free* quality functions, which is similar to locality. They then showed that resolution-limit-free quality functions do not suffer from the resolution limit as described by Fortunato and Barthélemy (2007). Their definition is as follows.

Definition 8 (Resolution-limit-free) *Call a clustering C of a graph G Q-optimal if for all clustering C' of G we have that $Q(G, C) \geq Q(G, C')$. Let C be a Q-optimal clustering of a graph G_1. Then the quality function Q is called resolution-limit-free if for each subgraph G_2 induced by $D \subset C$, the partition D is also Q-optimal.*

There are three differences compared to our locality property. First of all, Definition 8 refers only to the optimal clustering, not to the quality, that is, it is a property in the style of Kleinberg. Secondly, locality does not require that G_2 be a subgraph of G_1. Locality is stronger in that sense. Thirdly, and perhaps most importantly, in the subgraph G_2 induced by $D \subset C$, edges from a node in D to nodes not in D will be removed. That means that while G_1 and G_2 agree on the set of common nodes, they do not also agree on their neighborhood. So in this sense locality is weaker than resolution-limit-freedom.

The notion of resolution-limit-free quality functions was born out of the need to avoid the resolution limit of graph clustering. And indeed locality is not enough to guarantee that a quality function is free from this resolution limit.

We could look at a stronger version of locality, which replaces agreement on the neighborhood of a set V_a by plain agreement on that set. Such a *strong locality* property would imply resolution-limit-freedom. However, it is a very strong property in that it rules out many sensible quality functions. In particular, a strongly local quality function can not depend on the weight of edges entering or leaving a cluster, because that weight can be different in another graph that agrees only on that cluster.

The solution used by Traag et al. is to use the number of nodes instead of the volume of a cluster. In this way they obtain a resolution-limit-free variant of the Potts model by Reichardt and Bornholdt (2004), which they call the constant Potts model. But this comes at the cost of scale invariance.

4.2 Continuity

In the context of graphs, perhaps the most intuitive clustering function is finding the connected components of a graph. As a quality function, we could write

$$Q_{\text{coco}}(G, C) = \mathbf{1}[C = \hat{C}_{\text{coco}}(G)],$$

where the function $\hat{C}_{\text{coco}}$ yields the connected components of a graph.

This quality function is clearly permutation invariant, scale invariant, rich, and local. Since a consistent change can only remove edges between clusters and add edges within clusters, the coco quality function is also monotonic.

In fact, all of Kleinberg's axioms (reformulated in terms of graphs) also hold for $\hat{C}_{\text{coco}}$, which seems to refute their impossibility result. However, the impossibility proof can not be directly transfered to graphs, because it involves a multiplication and division by a maximum distance. In the graph setting this would be multiplication and division by a minimum edge weight, which can be zero.

Still, despite connected components satisfying all previously defined properties (except for strong locality), it is not a very useful quality function. In many real-world graphs, most nodes are part of one giant connected component (Bollobás, 2001). We would also like the clustering to be influenced by the weight of edges, not just by their existence. A natural way to rule out such degenerate quality functions is to require continuity.

Definition 9 (Continuity) *A quality function Q is* continuous *if a small change in the graph leads to a small change in the quality. Formally, Q is continuous if for every $\epsilon > 0$ and every graph $G = (V, E)$ there exists a $\delta > 0$ such that for all graphs $G' = (V, E')$, if $E(i,j) - \delta < E'(i,j) < E(i,j) + \delta$ for all nodes i and j, then $Q(G', C) - \epsilon < Q(G, C) < Q(G', C) + \epsilon$ for all clusterings C of G.*

Connected components clustering is not continuous, because adding an edge with a small weight δ between clusters changes the connected components, and hence dramatically changes the quality.

Continuous quality functions have an important property in practice, in that they provide a degree of robustness to noise. A clustering that is optimal with regard to a continuous quality function will still be close to optimal after a small change to the graph.

4.3 Summary of Axioms

We propose to consider the following six properties as axioms for graph clustering quality functions,

1. Permutation invariance (definition 1),
2. Scale invariance (definition 2),
3. Richness (definition 4),
4. Monotonicity (definition 6),
5. Locality (definition 7), and
6. Continuity (definition 9).

As mentioned previously, for families of quality functions we replace scale invariance by scale invariance for families (definition 3).

In the next section we will show that this set of axioms is consistent by defining a quality function and a family of quality functions that satisfies all of them. Additionally, the fact that there are quality functions that satisfy only some of the axioms shows that they are (at least partially) independent.

5. Modularity

For graph clustering one of the most popular quality functions is modularity (Newman and Girvan, 2004), despite its limitations (Good et al., 2010; Traag et al., 2011),

$$Q_{\text{modularity}}(G, C) = \sum_{c \in C} \left(\frac{w_c}{v_V} - \left(\frac{v_c}{v_V} \right)^2 \right). \tag{1}$$

In this expression $v_c(G) = \sum_{i \in c} \sum_{j \in V} E(i,j)$ is the volume of a cluster, while $w_c(G) = \sum_{i,j \in c} E(i,j)$ is the within cluster weight. v_V is the volume of the entire graph. We leave the argument G implicit for readability.

It is easy to see that modularity is permutation invariant, scale invariant and continuous.

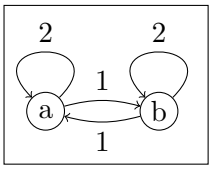
Theorem 1 *Modularity is rich.*

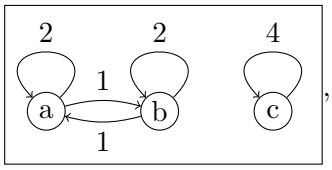
The proof of Theorem 1 is in appendix A.

An important aspect of modularity is that volume and within weight are normalized with respect to the total volume of the graph. This ensures that the quality function is scale invariant, but it also means that the quality can change in unexpected ways when the total volume of the graph changes. This leads us to Theorem 2.

Theorem 2 *Modularity is not local.*

Proof Consider the graphs

$G_1 =$ 

$G_2 =$ 

which agree on the set $V_a = \{a, b\}$. Note that we draw the graphs as directed graphs, to make it clear that each undirected edge is counted twice for the purposes of volume and within cluster weight. Now take the clusterings $C_a = \{\{a\}, \{b\}\}$ and $D_a = \{\{a, b\}\}$ of V_a; $C_1 = \{\}$ of $V_1 \setminus V_a$; and $C_2 = \{\{c\}\}$ of $V_2 \setminus V_a$. Then

$$Q_{\text{modularity}}(G_1, C_a \cup C_1) = 1/6 > 0 = Q_{\text{modularity}}(G_1, D_a \cup C_1),$$

while

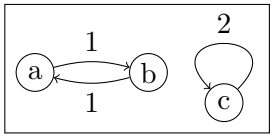
$$Q_{\text{modularity}}(G_2, C_a \cup C_2) = 23/50 < 24/50 = Q_{\text{modularity}}(G_2, D_a \cup C_2).$$

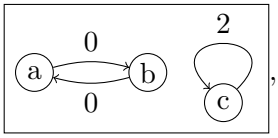
This counterexample shows that modularity is not local. ■

Even without changing the node set, changes in the total volume can be problematic, as shown by the following theorem.

Theorem 3 *Modularity is not monotonic.*

Proof Consider the graphs

$G =$ 

$G' =$ 

and the clustering $C = \{\{a\}, \{b\}, \{c\}\}$. G' is a C-consistent improvement of G, because the weight of a between-cluster edge is decreased. The modularity of C in G is $Q_{\text{modularity}}(G, C) = 1/8$, while the modularity of C in G' is $Q_{\text{modularity}}(G', C) = 0$. So modularity can decrease with a consistent change of a graph, and hence it is not a monotonic quality function. ■

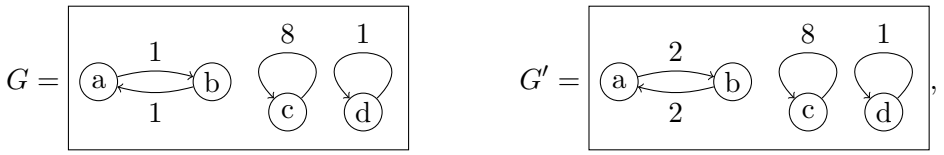
Monotonicity might be too strong a condition. When the goal is to find a clustering of a single graph, we are not actually interested in the absolute value of a quality function. Rather, what is of interest is the optimal clustering, and which changes to the graph preserve this optimum. At a smaller scaler, we can look at the relation between two clusterings. If C is better then D on a graph G, then on what other graphs is C better then D?

We therefore define a relative version of monotonicity, in the hopes that modularity does satisfy this weaker version.

Definition 10 (Relative monotonicity) *A quality function Q is* relatively monotonic *if for all graphs G and G' and clusterings C and D, if G' is a C-consistent improvement of G and G is a D-consistent improvement of G' and $Q(G,C) \geq Q(G,D)$ then $Q(G',C) \geq Q(G',D)$.*

Theorem 4 *Modularity is not relatively monotonic.*

Proof Take the graphs



and the clusterings $C = \{\{a,b,c\},\{d\}\}$ and $D = \{\{a\},\{b\},\{c,d\}\}$. G' is a C-consistent improvement of G, because the weight of a within cluster edge is increased. G is a D-consistent improvement of G', because the weight of a between cluster edge is decreased. However $Q_{\text{modularity}}(G,C) = 20/121 > 16/121 = Q_{\text{modularity}}(G,D)$ while $Q_{\text{modularity}}(G',C) = 24/169 < 28/121 = Q_{\text{modularity}}(G',D)$. This counterexample shows that modularity is not relatively monotonic. ■

6. Adaptive Scale Modularity

The problems with modularity stem from the fact that the total volume can change when changes are made to the graph. It is therefore natural to look at a variant of modularity where the total volume is replaced by a constant M,

$$Q_{M\text{-fixed}}(G,C) = \sum_{c \in C} \left(\frac{w_c}{M} - \left(\frac{v_c}{M} \right)^2 \right).$$

This quality function is obviously local. It is also a scale invariant family parameterized by M. However, this fixed scale modularity quality function is *not* scale invariant for any fixed scale $M > 0$.

We might hope that fixed scale modularity would be monotonic, because it doesn't suffer from the problem where changes in the edge weights affect the total volume. Unfortunately, fixed scale modularity has problems when the volume of a cluster starts to exceed $M/2$.

In that case, increasing the weight of within cluster edges starts to decrease the fixed scale modularity. Looking at a cluster c with volume $v_c = w_c + b_c$,

$$\frac{\partial Q_{M\text{-fixed}}(G, C)}{\partial w_c} = \frac{1}{M} - \frac{2v_c}{M^2}.$$

This derivative is negative when $2v_c > M$, so in that case increasing the weight of a within-cluster edge will decrease the quality. Hence fixed scale modularity is not monotonic.

The above argument also suggests a possible solution: add $2v_c$ to the normalization factor M. Or more generally, add γv_c with $\gamma \geq 2$, which leads to the quality function

$$Q_{M,\gamma}(G, C) = \sum_{c \in C} \Big(\frac{w_c}{M + \gamma v_c} - \Big(\frac{v_c}{M + \gamma v_c} \Big)^2 \Big).$$

This *adaptive scale modularity* quality function is clearly still permutation invariant, continuous and local. For $M = 0$ it is also scale invariant. Since the value of M should scale along with the edge weights, adaptive scale modularity is a scale invariant family parameterized by M. Additionally, we have the following two theorems:

Theorem 5 *Adaptive scale modularity is rich for all $M \geq 0$ and $\gamma \geq 1$.*

Theorem 6 *Adaptive scale modularity is monotonic for all $M \geq 0$ and $\gamma \geq 2$.*

The proofs of these theorems can be found in appendices B and C.

This shows that adaptive scale modularity satisfies all six axioms we have defined for families of graph clustering quality functions, and the six axioms for single quality functions when $M = 0$. This shows that our extended set of axioms is consistent.

6.1 Relation to Other Quality Functions

Interestingly, in the limit as M goes to 0, the adaptive-scale quality function becomes similar to normalized cut (Shi and Malik, 2000) with an added constant,

$$Q_{0,\gamma}(G, C) = \frac{1}{\gamma} \sum_{c \in C} \Big(\frac{w_c}{v_c} - \frac{1}{\gamma} \Big).$$

This 0-adaptive modularity is also scale invariant as a single quality function.

Conversely, when M goes to infinity the quality goes to 0. However, the quality function approaches unnormalized cut in behavior:

$$\lim_{M \to \infty} M \cdot Q_{M,\gamma}(G, C) = \sum_{c \in C} w_c.$$

This expression is similar to the Constant Potts model (CPM) by Traag et al. (2011),

$$Q_{\text{cpm}}(G, C) = \sum_{c \in C} \Big(w_c - \gamma n_c^2 \Big).$$

In contrast to the quality functions discussed thus far, CPM uses the number of nodes instead of volume to control the size of clusters. Like adaptive scale modularity, the constant Potts model satisfies all six axioms (as a family).

As stated before, the fixed scale and adaptive scale modularity quality functions are a scale invariant family; they are not scale invariant for a fixed value of M (except for $M = 0$). This is not a large problem in practice, since scale invariance is often sacrificed to overcome the resolution limit of modularity (Fortunato and Barthélemy, 2007). In fact, fixed scale modularity is proportional to the quality function introduced by Reichardt and Bornholdt (2004),

$$Q_{\mathrm{RB}}(G, C) = \sum_{c \in C} \Big(w_c - \gamma_{\mathrm{RB}} \frac{v_c^2}{v_V} \Big) = M \cdot Q_{M\text{-fixed}}(G, C),$$

with $M = v_V / \gamma_{\mathrm{RB}}$.

6.2 Parameter Dependence Analysis

There has been a lot of interest in the so called resolution limit of modularity.

This problem can be illustrated with a simple graph that consists of a ring of cliques, where each clique is connected to the next one with a single edge. We would like the clusters in the optimal clustering to correspond to the cliques in the ring. It was observed by Fortunato and Barthélemy (2007) that, as the number of cliques in the ring increases, at some point the clustering with the highest modularity will have multiple cliques per cluster.

This resolution problem stems from the fact that the behavior of modularity depends on the total volume of the graph. Both the fixed scale and adaptive scale modularity quality functions instead have a parameter M, and hence do not suffer from this problem. In fact, any local quality function will not have a resolution limit in the sense of Fortunato and Barthélemy. A similar observation was made by Traag et al. (2011) in the context of modularity like quality functions.

In real situations graphs are not uniform as in the ring-of-cliques model. But we can still take simple uniform problems as a building block for larger and more complex graphs, since for local quality functions the rest of the network doesn't matter. Therefore we will look at a simple problem with two subgraphs of varying sizes connected by a varying number of edges. More precisely, we take two cliques each with within weight w, connected by edges with weight b. The total volume of this (sub)graph is then $2w + 2b$.

There are three possible outcomes when clustering such a two-clique network: (1) the optimal solution has a single cluster; (2) the optimal solution has two clusters, corresponding to the two cliques; (3) the optimal solution has more than two clusters, splitting the cliques apart. See Figure 1 for an illustration. Which of these outcomes is desirable depends on the circumstances.

Another heterogeneous resolution limit model was proposed by Lancichinetti and Fortunato (2011). In this situation there are two cliques of equal size connected by a single edge, and a random subgraph. Now the ideal solution would be to find three clusters, one for each clique and one for the random subgraph. The optimal split of the random subgraph will roughly cut it in half, with a fixed fraction of the volume being between the two clusters (Reichardt and Bornholdt, 2007). So this model can be considered as a combination of two

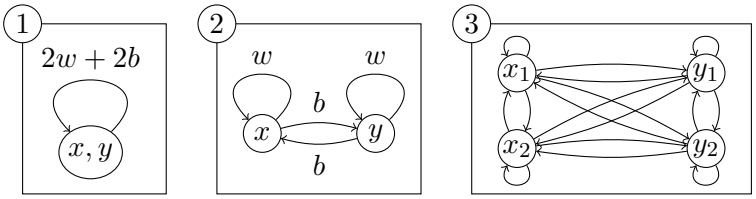


Figure 1: An illustration of the possible outcomes when clustering a two-clique network. Clusters are indicated by circles. In outcome (3), the vertical edges each have weight $w/4$, while the horizontal and diagonal ones have weight $b/4$.

instances of our simpler problem, one for the two cliques and one for the random subgraph.[1] Hence, we want outcome (2) for the cliques, and outcome (1) for the random subgraph.

In Figure 2 we show which graphs give which outcomes for adaptive scale modularity with various parameter settings. The first column, $\gamma = 0$, is of particular interest, since it corresponds to fixed scale modularity and hence also to Q_{RB} and to modularity in certain graphs. In the third row we can see that when $2v = 2w + 2b > M = 100$ the cliques are split apart. This is precisely the region in which monotonicity no longer holds. Overall, the parameter M has the effect of determining the scale; each row in this figure is merely the previous row magnified by a factor 10. Increasing M has the effect of merging small clusters. On the other hand, the γ parameter controls the slope of the boundary between outcomes (1) and (2), that is, the fraction of edges that should be within a cluster. This is most clearly seen when $M = 0$, while otherwise the effect of M dominates for small clusters.

7. Conclusion and Open Questions

In this paper we presented an axiomatic framework for graph clustering quality functions consisting of six properties. We showed that modularity does not satisfy the monotonicity property. This motivated the derivation of a new family of quality functions, adaptive scale modularity, that satisfies all properties and has standard graph clustering quality functions as special cases. Results of an experimental parameter dependence analysis showed the high flexibility of adaptive scale modularity. However, adaptive scale modularity should not be considered the solution to all the problems of modularity, but rather an example of how axioms can be used in practice.

An overview of the discussed axioms and quality functions can be found in table 1. Many more quality functions have been proposed in the literature, so this list is by no means exhaustive. An interesting topic for future research is to make a survey of which existing quality functions satisfy which of the proposed properties.

We also investigated resolution-limit-free quality functions as defined by Traag et al. (2011). As illustrated in section 6.2, adaptive scale modularity allows to perform clustering at various resolutions, by varying the values of its two parameters. However it is not resolution-limit-free.

1. Lancichinetti and Fortunato include edges between the cliques and the random subgraph to ensure that the entire network is connected, these edges are not relevant to the problem.

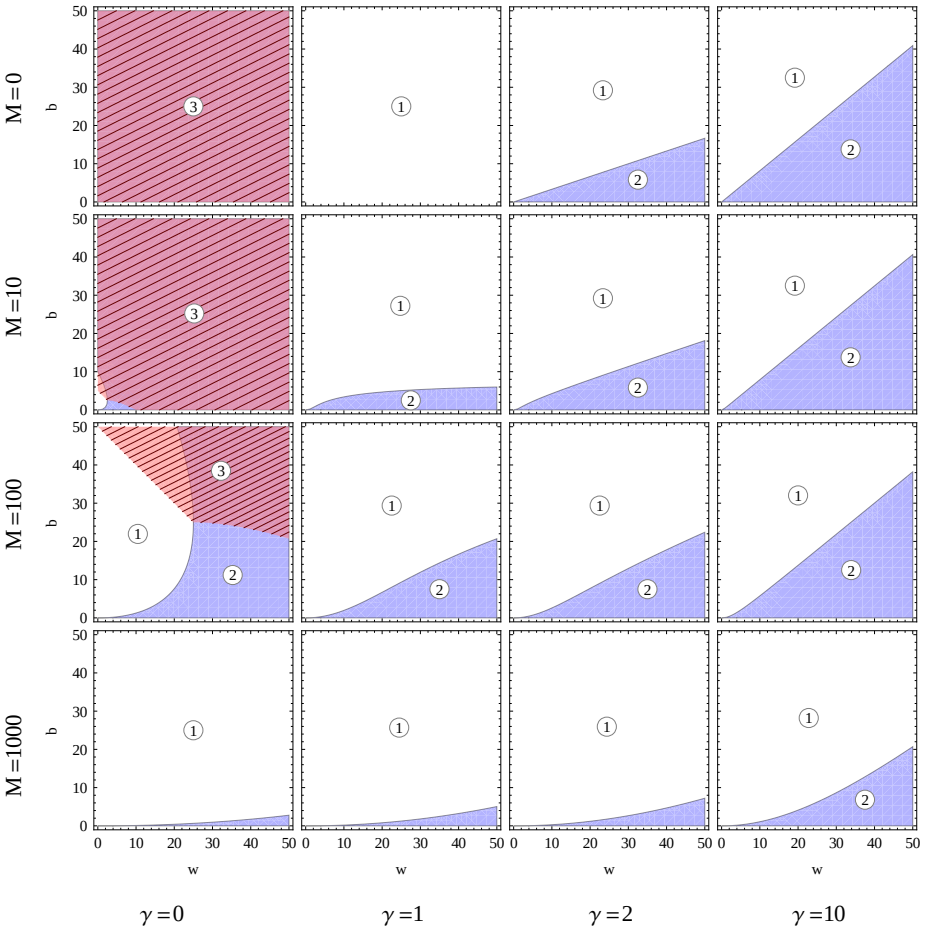


Figure 2: The behavior of $Q_{M,\gamma}$ for varying parameter values. The graph consists of two subgraphs with w internal weight each, connected by an edge with weigh b. Hence the volume of the total graph is $2w + 2b$. In region (1) the optimal clustering has a single cluster, In region (2) (light blue) the optimal clustering separates the subgraphs. In region (3) (red, hatched) the subgraphs themselves will be split apart.

Our paper did not address questions such as finding a best quality function (Almeida, Guedes, Jr., and Zaki, 2011), or selecting a significant resolution scale (Traag et al., 2013). The aim was to provide necessary conditions about what a good quality function is, in order to rule out and/or to improve quality functions. The proposed axioms and the introduction of adaptive scale modularity are an effort in this direction.

We also did not address the question of finding a clustering with the highest quality. Finding the optimal value of quality functions such as modularity is NP-hard (Brandes et al., 2008), but several heuristic and approximation algorithms have been developed. One class of algorithms uses a divisive approach, see for instance Newman (2006) and Ruan and Zhang (2008). For such a tactic to be valid, an optimal or close to optimal clustering of a subgraph

	Permutation invariance	Scale invariance	Scale invariance (family)	Richness	Monotonicity	Locality	Continuity
Connected components	✓	✓	n.a.	✓	✓	✓	–
Modularity	✓	✓	n.a.	✓	–	–	✓
Reichardt and Bornholdt (2004)	✓	✓	✓	✓	–	–	✓
Fixed scale modularity	✓	$M = 0$	✓	✓	–	✓	✓
Adaptive scale modularity	✓	$M = 0$	✓	$\gamma \geq 1$	$\gamma \geq 2$	✓	✓
Constant Potts Model (Traag et al., 2011)	✓	–	✓	$\gamma > 0$	✓	✓	✓
Normalized cut	✓	✓	n.a.	–	✓	✓	✓

Table 1: Overview of quality functions discussed in this paper and the properties they satisfy.

should also be a near optimal clustering of the entire graph. This is ensured by locality. Recently Dinh and Thai (2013) proposed polynomial-time approximation algorithms for the modularity maximization in the context of scale free networks. It would be interesting to investigate the suitability of these algorithms for adaptive scale modularity maximization.

In this work we have only looked at non-negative weights, undirected graphs, and only at hard partitioning. An extension to graphs with negative weights, to directed graphs and to overlapping clusters remains to be investigated. Another open problem is how to use these axioms for reasoning about quality functions and clustering algorithms.

Acknowledgments

We thank the reviewers for their comments. This work has been partially funded by the Netherlands Organization for Scientific Research (NWO) within the NWO project 612.066.927.

Appendix A. Proof of Theorem 1 (Modularity is Rich)

The proofs of richness rely on clique graphs,

Definition 11 (Clique graph) *Let V be a set of nodes, C be a partition of V, and k be a positive constant. The* clique graph of C with edge weight k *is defined as $G = (V, E)$ where $E(i, j) = k$ if $i \sim_C j$ and $E(i, j) = 0$ otherwise.*

Proof

Let V be a set of nodes and $C \neq \{V\}$ be a clustering of V. Let $G = (V, E)$ be a clique graph of C with edge weight 1. Note that $E(i, i) = 1$, so any possible cluster will have a positive volume. Let D be a clustering of G with maximal modularity.

Suppose that there is a cluster $d \in D$ that contains $i, j \in d$ with $i \not\sim_C j$. Then we can split the cluster into $d_1 = \{k \in d \mid k \sim_C i\}$ and $d_2 = \{k \in d \mid k \not\sim_C i\}$. Because there are no edges between nodes in d_1 and nodes in d_2, it is the case that $w_d = w_{d_1} + w_{d_2}$. Both d_1 and d_2 are non-empty and have a positive volume, so $v_d^2 = (v_{d_1} + v_{d_2})^2 < v_{d_1}^2 + v_{d_2}^2$. Therefore $Q_{\text{modularity}}(G, D) < Q_{\text{modularity}}(G, D \backslash \{d\} \cup \{d_1, d_2\})$. So D does not have maximal modularity, which is a contradiction.

Suppose, on the other hand that all clusters $d \in D$ are a subset of some cluster in C, that is, D is a refinement of C. Then either $D = C$, or there are two clusters $d_1, d_2 \in D$ that are both a subset of the same cluster $c \in C$. In the latter case we can combine the two clusters into $d = d_1 \cup d_2$. The within weight of this combined cluster is $w_d = |d|^2 = w_{d_1} + w_{d_2} + 2|d_1||d_2|$. The squared volume of the combined cluster is $v_d^2 = |d|^2|c|^2 = v_{d_1}^2 + v_{d_2}^2 + 2|d_1||d_2||c|^2$. So this changes increases the modularity by

$$\begin{aligned} &Q_{\text{modularity}}(G, D \setminus \{d_1, d_2\} \cup \{d\}) - Q_{\text{modularity}}(G, D) \\ &\quad = 2|d_1||d_2|/v_V - 2|d_1||d_2||c|^2/v_V^2 \\ &\quad = 2|d_1||d_2|(v_V - |c|^2)/v_V^2 > 0, \end{aligned}$$

which contradicts the assumption that D has maximal modularity. Therefore the only optimal clustering of G is C. Note that the above inequality only holds when $|c|^2 = v_c < v_V$, which is the case because $C \neq \{V\}$.

When $C = \{V\}$, a clique graph will not work; because both $\{V\}$ and the clustering that assigns half the nodes to one cluster, and half to another have modularity equal to 0. In this case, instead define $G = (V, E)$ by $E(i, j) = 1$ if $i \neq j$ and 0 if $i = j$. Then the modularity for C is $q(G, \{V\}) = 0$. Any cluster d in a clustering D will have $v_d = |d|(|V| - 1)$ and $w_d = |d|(|d| - 1)$. Therefore the contribution of this cluster to the total quality is $-|d|(|V| - |d|)/(|V|^2(|V| - 1))$, which is negative when $|d| < |V|$. So the modularity of any clustering other than $\{V\}$ will be negative, hence $\{V\}$ is the only optimal clustering.

Since for every C we can construct a graph where C is the only optimal clustering, modularity is rich. ■

Appendix B. Proof of Theorem 5 (Adaptive Scale Modularity is Rich)

Denote by $f_C(d)$ the largest fraction of any cluster from C that is contained in a cluster d.

$$f_C(d) = \max_{c \in C} \frac{|c \cap d|}{|c|}.$$

For any clustering D we have that

$$\sum_{d \in D} f_C(d) = \sum_{d \in D} \max_{c \in C} \frac{|c \cap d|}{|c|} \leq \sum_{d \in D} \sum_{c \in C} \frac{|c \cap d|}{|c|} = |C|.$$

And since $f_C(d) \leq 1$ for all clusters d, we also have that

$$\sum_{d \in D} f_C(d) \leq |D|.$$

Lemma 7 *For a clique graph of C it is the case that $w_d/v_d \leq f_C(d)$.*

Proof Given a cluster d and a clique graph G of C with weight $k > 0$, the volume of d is

$$v_d = \sum_{c \in C} k|c \cap d||c|,$$

and the within cluster weight is

$$w_d = \sum_{c \in C} k|c \cap d|^2.$$

Therefore

$$w_d \leq \sum_{c \in C} k|c \cap d||c| f_C(d) = v_d f_C(d).$$

And hence $w_d/v_d \leq f_C(d)$. ■

Lemma 8 *Let G be the clique graph of a clustering C with weight k, and let $0 < \beta < 1$ be a constant. Then $\sum_{d \in D}(w_d/v_d - \beta) = (1-\beta)|C|$ if $D = C$, while $\sum_{d \in D}(w_d/v_d - \beta) < (1-\beta)|C| - \epsilon$ if $D \neq C$, where $\epsilon = \min(\beta, 1-\beta, 1/|V|)/2$.*

Proof Suppose that $D = C$, then for every cluster $c \in C$, $w_c = v_c = k|c|^2$, and so

$$\sum_{c \in C}\Big(\frac{w_d}{v_d} - \beta\Big) = (1-\beta)|C|.$$

Otherwise, $D \neq C$. Assume that $\sum_{d \in D}(w_d/v_d - \beta) \geq (1-\beta)|C| - \min(\beta, 1/|V|)/2$. By Lemma 7,

$$\begin{aligned}
&|C| - \beta(|C|+1) \\
<&|C| - \beta|C| - \epsilon \\
\leq &\sum_{d \in D}(\frac{w_d}{v_d} - \beta) \\
\leq &\sum_{d \in D}(f_C(d) - \beta) \\
\leq &|C| - \beta|D|.
\end{aligned}$$

Since $\beta > 0$, this implies that $|D| < |C| + 1$.

Additionally, since $f_C(d) \leq 1$ for all clusters $d \in D$,

$$\begin{aligned}
&(1-\beta)(|C|-1) \\
<&(1-\beta)|C| - \epsilon \\
\leq &\sum_{d \in D} (f_C(d) - \beta) \\
\leq &(1-\beta)|D|
\end{aligned}$$

Since $\beta < 1$, this implies that $|D| > |C| - 1$. Hence $|D| = |C|$.

Suppose that $f_C(d) < 1$ for some $d \in D$, which implies that $|c \cap d| < |c|$. Because edges are discrete, this can only happen when $|c \cap d| \leq |c| - 1$ for all clusters c. And the size of clusters is bounded by $|c| \leq |V|$. Hence $f_C(d) \leq (|V| - 1)/|V| = 1 - 1/|V|$. And since for all other clusters d', $f_C(d') \leq 1$, we then have

$$\begin{aligned}
&\sum_{d \in D} (f_C(d) - \beta) \\
\leq &(1-\beta)|D| - 1/|V| \\
<&(1-\beta)|C| - \epsilon \\
\leq &\sum_{d \in D} (w_d/v_d - \beta) \\
\leq &\sum_{d \in D} (f_C(d) - \beta),
\end{aligned}$$

which is a contradiction. Hence, it must be the case that $f_C(d) = 1$ for all clusters $d \in D$. By the definition of f_C this means that for every d there is a cluster $c \in C$ such that $|c \cap d| = |c|$, and therefore $c \subseteq d$. Since the clusters are disjoint and $|D| = |C|$, this implies that $D = C$. Which is a contradiction, so $\sum_{d \in D}(w_d/v_d - \beta) < (1-\beta)|C| - \epsilon$. ■

When $M = 0$, the adaptive scale modularity reduces to $w_d/(\gamma v_d) - |D|/\gamma^2$, and the above lemma is enough to prove richness. For non-zero values of M, we can get 'close enough' by choosing large enough edge weights. This is formalized in the following lemma.

Lemma 9 *Let d be a cluster in a clustering of a clique graph of C with weight k. Then*

$$\frac{w_d}{v_d} - \beta - \beta M/k \leq q(d)/\beta \leq \frac{w_d}{v_d} - \beta + 2\beta^2 M/k,$$

where

$$q(d) = \frac{w_d}{M + v_d/\beta} - \Big(\frac{v_d}{M + v_d/\beta}\Big)^2$$

denotes the contribution of d to the M-adaptive modularity.

Proof Since clusters are non-empty, and in a clique graph $E(i,i) = k$, it follows that $v_d \geq w_d \geq k$. So

$$\begin{aligned}
& q(d)/\beta \\
=& \frac{\beta M w_d + v_d w_d - \beta v_d^2}{(\beta M + v_d)^2} \\
=& \frac{w_d}{v_d} - \beta + \frac{\beta^2 M(\beta M + 2v_d) - \beta^2 M^2 w_d/v_d - \beta M w_d}{(\beta M + v_d)^2} \\
\leq& \frac{w_d}{v_d} - \beta + \frac{\beta^2 M(\beta M + 2v_d)}{(\beta M + v_d)^2} \\
\leq& \frac{w_d}{v_d} - \beta + \frac{2\beta^2 M(\beta M + 2v_d)}{(\beta M + v_d)(\beta M + 2v_d)} \\
=& \frac{w_d}{v_d} - \beta + \frac{2\beta^2 M}{\beta M + v_d} \\
\leq& \frac{w_d}{v_d} - \beta + \frac{2\beta^2 M}{k}.
\end{aligned}$$

And since $w_d \leq v_d$,

$$\begin{aligned}
& q(d)/\beta \\
=& \frac{w_d}{v_d} - \beta + \frac{\beta^2 M(\beta M + 2v_d) - \beta^2 M^2 w_d/v_d - \beta M w_d}{(\beta M + v_d)^2} \\
\geq& \frac{w_d}{v_d} - \beta - \frac{\beta^2 M^2 + \beta M v_d}{(\beta M + v_d)^2} \\
=& \frac{w_d}{v_d} - \beta - \frac{\beta M}{\beta M + v_d} \\
\geq& \frac{w_d}{v_d} - \beta - \frac{\beta M}{k}.
\end{aligned}$$

■

Combining these lemmas yields the proof of the general theorem:

Proof Given a clustering C. Define $\beta = 1/\gamma$. If $\gamma > 1$ then $0 < \beta < 1$. Pick $k > 3|V|\beta^2 M/\epsilon$ where ϵ is defined as in Lemma 8.

Let G be the clique graph of C with weight k. Let $D \neq C$ be a clustering of G. Then by Lemmas 8 and 9,

$$\begin{aligned}
&Q_{M,\gamma}(G,D)/\beta \\
=&\sum_{d\in D} q(d) \\
\leq&\sum_{d\in D} (w_d/v_d - \beta + 2\beta^3 M/k) \\
\leq&(1-\beta)|C| + 2|D|\beta^3 M/k - \epsilon \\
\leq&(1-\beta)|C| + 2|V|\beta^2 M/k - \epsilon \\
<&(1-\beta)|C| - |V|\beta^2 M/k \\
\leq&(1-\beta)|C| - |C|\beta^2 M/k \\
=&\sum_{c\in C} (w_c/v_c - \beta + \beta^2 M/k) \\
\leq&Q_{M,\gamma}(C)/\beta.
\end{aligned}$$

Hence the quality is maximal for C. Since there is a clique graph and k for every clustering, adaptive scale modularity is rich. ■

Appendix C. Proof of Theorem 6 (Adaptive Scale Modularity is Monotonic)

Proof

Given a constants $M > 0$ and $\gamma \geq 2$, a graph G and a clustering C of G. Let $c \in C$ be any cluster. Writing the volume of c as $v_c = w_c + b_c$, the contribution of this cluster to the quality of G is $q(w_c, b_c)$ where

$$q(w,b) = \frac{w}{M+\gamma w+\gamma b} - \Big(\frac{w+b}{M+\gamma w+\gamma b}\Big)^2.$$

The partial derivatives of q are

$$\begin{aligned}
\frac{\partial q(w,b)}{\partial w} &= \frac{M^2 + (\gamma-2)M(w+b) + \gamma b(M+\gamma w+\gamma b)}{(M+\gamma w+\gamma b)^3} \geq 0 \\
\frac{\partial q(w,b)}{\partial b} &= -\frac{\gamma w M + (w+b)(M+\gamma^2 w)}{(M+\gamma w+\gamma b)^3} \leq 0.
\end{aligned}$$

This means that q is a monotonically non-decreasing function in w and a non-increasing function in b.

For any graph G' that is a C-consistent change of G, it holds that $w'_c \geq w_c$ and $b'_c \leq b_c$. So $q(w'_c, b'_c) \geq q(w_c, b_c)$. And therefore $Q_{M,\gamma}(G', C) \geq Q_{M,\gamma}(G, C)$. So adaptive scale modularity is monotonic. ■

References

Margareta Ackerman and Shai Ben-David. Measures of clustering quality: A working set of axioms for clustering. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *NIPS*, pages 121–128. Curran Associates, Inc., 2008.

Margareta Ackerman and Shai Ben-David. A characterization of linkage-based hierarchical clustering. *Journal of Machine Learning Research*, 2013.

Margareta Ackerman, Shai Ben-David, and David Loker. Towards property-based classification of clustering paradigms. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *NIPS*, pages 10–18. Curran Associates, Inc., 2010a.

Margareta Ackerman, Shai Ben-David, and David Loker. Characterization of linkage-based clustering. In Adam Tauman Kalai and Mehryar Mohri, editors, *COLT*, pages 270–281. Omnipress, 2010b. ISBN 978-0-9822529-2-5.

Margareta Ackerman, Shai Ben-David, Simina Brânzei, and David Loker. Weighted clustering. In Jörg Hoffmann and Bart Selman, editors, *AAAI*. AAAI Press, 2012.

Margareta Ackerman, Shai Ben-David, David Loker, and Sivan Sabato. Clustering oligarchies. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 31 of *JMLR Workshop and Conference Proceedings*, pages 66–74, 2013.

Helio Almeida, Dorgival Guedes, Wagner Meira Jr., and Mohammed J. Zaki. Is there a best quality metric for graph clusters? In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6911 of *Lecture Notes in Computer Science*, pages 44–59. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-23779-9.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.*, 2008(10):P10008, 2008. ISSN 1742-5468. doi: 10.1088/1742-5468/2008/10/P10008. URL `http://dx.doi.org/10.1088/1742-5468/2008/10/P10008`.

Béla Bollobás. *The Evolution of Random Graphs – the Giant Component*, pages 130–159. Cambridge University Press, 2001. ISBN 9780521797221.

Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008. ISSN 1041-4347. doi: 10.1109/TKDE.2007.190689.

Sébastien Bubeck and Ulrike von Luxburg. Nearest neighbor clustering: A baseline method for consistent clustering with arbitrary objective functions. *J. Mach. Learn. Res.*, 10: 657–698, June 2009. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1577069.1577092`.

Gunnar Carlsson, Facundo Mémoli, Alejandro Ribeiro, and Santiago Segarra. Axiomatic construction of hierarchical clustering in asymmetric networks. *CoRR*, abs/1301.7724, 2013.

Thang N. Dinh and My T. Thai. Community detection in scale-free networks: Approximation algorithms for maximizing modularity. *IEEE Journal on Selected Areas in Communications*, 31(6):997–1006, 2013.

Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proc. Natl. Acad. Sci. USA*, 104(1):36–41, 2007. doi: 10.1073/pnas.0605965104.

Sreenivas Gollapudi and Aneesh Sharma. An axiomatic approach for result diversification. In *Proceedings of the 18th International Conference on World Wide Web*, pages 381–390, 2009.

Benjamin H. Good, Yves A. de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Phys. Rev. E*, 81(4):046106, April 2010. doi: 10.1103/PhysRevE.81.046106. URL `http://dx.doi.org/10.1103/PhysRevE.81.046106`.

Jon M. Kleinberg. An impossibility theorem for clustering. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 446–453. MIT Press, 2002. ISBN 0-262-02550-7.

Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Phys. Rev. E*, 84:066122, December 2011. doi: 10.1103/PhysRevE.84.066122. URL `http://dx.doi.org/10.1103/PhysRevE.84.066122`.

Marina Meila. Comparing clusterings: an axiomatic view. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 577–584. ACM, 2005.

Mark E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74(3):036104, July 2006. doi: 10.1103/PhysRevE.74.036104. URL `http://dx.doi.org/10.1103/PhysRevE.74.036104`.

Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb 2004. doi: 10.1103/PhysRevE.69.026113. URL `http://pre.aps.org/abstract/PRE/v69/i2/e026113`.

Jan Puzicha, Thomas Hofmann, and Joachim M. Buhmann. A theory of proximity based clustering: Structure detection by optimization. *Pattern Recognition*, 33:617–634, 1999.

Jörg Reichardt and Stefan Bornholdt. Detecting fuzzy community structures in complex networks with a Potts model. *Phys. Rev. Lett.*, 93:218701, 2004. doi: 10.1103/PhysRevLett.93.218701.

Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006.

Jörg Reichardt and Stefan Bornholdt. Partitioning and modularity of graphs with arbitrary degree distribution. *Phys. Rev. E*, 76:015102, Jul 2007. doi: 10.1103/PhysRevE.76.015102. URL `http://link.aps.org/doi/10.1103/PhysRevE.76.015102`.

Jianhua Ruan and Weixiong Zhang. Identifying network communities with a high resolution. *Phys. Rev. E*, 77:016104, Jan 2008. doi: 10.1103/PhysRevE.77.016104. URL http://link.aps.org/doi/10.1103/PhysRevE.77.016104.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. volume 22, pages 888–905, Washington, DC, USA, August 2000. IEEE Computer Society. doi: 10.1109/34.868688. URL http://dx.doi.org/10.1109/34.868688.

Vincent A. Traag, Paul Van Dooren, and Yurii E. Nesterov. Narrow scope for resolution-limit-free community detection. *Phys. Rev. E*, 84:016114, Jul 2011. doi: 10.1103/PhysRevE.84.016114. URL http://link.aps.org/doi/10.1103/PhysRevE.84.016114.

Vincent A. Traag, Gautier Krings, and Paul Van Dooren. Significant scales in community structure. *Submitted*, Jun 2013. URL http://arxiv.org/abs/1306.3398.

Twan van Laarhoven and Elena Marchiori. Graph clustering with local search optimization: The resolution bias of the objective function matters most. *Phys. Rev. E*, 87:012812, Jan 2013. doi: 10.1103/PhysRevE.87.012812. URL http://link.aps.org/doi/10.1103/PhysRevE.87.012812.

Reza Bosagh Zadeh and Shai Ben-David. A uniqueness theorem for clustering. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 639–646, Arlington, Virginia, United States, 2009. AUAI Press. ISBN 978-0-9749039-5-8. URL http://dl.acm.org/citation.cfm?id=1795114.1795189.

Journal of Machine Learning Research 15 (2014) 217-252 Submitted 10/11; Revised 5/13; Published 1/13

Convex vs Non-Convex Estimators for Regression and Sparse Estimation: the Mean Squared Error Properties of ARD and GLasso

Aleksandr Aravkin SARAVKIN@US.IBM.COM
IBM T.J. Watson Research Center
1101 Kitchawan Rd, 10598
Yorktown Heights, NY, USA

James V. Burke BURKE@MATH.WASHINGTON.EDU
Department of Mathematics, Box 354350
University of Washington
Seattle, WA, 98195-4350 USA

Alessandro Chiuso CHIUSO@DEI.UNIPD.IT
Gianluigi Pillonetto GIAPI@DEI.UNIPD.IT
Department of Information Engineering
Via Gradenigo 6/A
University of Padova
Padova, Italy

Editor: Francis Bach

Abstract

We study a simple linear regression problem for grouped variables; we are interested in methods which jointly perform estimation and *variable selection*, that is, that automatically set to zero groups of variables in the regression vector. The Group Lasso (GLasso), a well known approach used to tackle this problem which is also a special case of Multiple Kernel Learning (MKL), boils down to solving convex optimization problems. On the other hand, a Bayesian approach commonly known as Sparse Bayesian Learning (SBL), a version of which is the well known Automatic Relevance Determination (ARD), lead to non-convex problems. In this paper we discuss the relation between ARD (and a penalized version which we call PARD) and Glasso, and study their asymptotic properties in terms of the Mean Squared Error in estimating the unknown parameter. The theoretical arguments developed here are independent of the correctness of the prior models and clarify the advantages of PARD over GLasso.

Keywords: Lasso, Group Lasso, Multiple Kernel Learning, Bayesian regularization, marginal likelihood

1. Introduction

We consider sparse estimation in a linear regression model where the explanatory factors $\theta \in \mathbb{R}^m$ are naturally grouped so that θ is partitioned as $\theta = [{\theta^{(1)}}^\top \quad {\theta^{(2)}}^\top \quad \dots \quad {\theta^{(p)}}^\top]^\top$. In this setting we assume that θ is group (or block) sparse in the sense that many of the constituent vectors $\theta^{(i)}$ are zero or have a negligible influence on the output $y \in \mathbb{R}^n$. In

addition, we assume that the number of *unknowns* m is large, possibly larger than the size of the available data n. Interest in general sparsity estimation and optimization has attracted the interest of many researchers in statistics, machine learning, and signal processing with numerous applications in feature selection, compressed sensing, and selective shrinkage (Hastie and Tibshirani, 1990; Tibshirani, 1996; Donoho, 2006; Candes and Tao, 2007). The motivation for our study of the group sparsity problem comes from the "dynamic Bayesian network" scenario identification problem (Chiuso and Pillonetto, 2012, 2010b,a). In a dynamic network scenario, the *explanatory variables* are the past histories of different input signals, with the groups $\theta^{(i)}$ representing the impulse responses[1] describing the relationship between the i-th input and the output y. This application informs our view of the group sparsity problem as well as our measures of success for a particular estimation procedure.

Several approaches have been put forward in the literature for joint estimation and variable selection problems. We cite the well known Lasso (Tibshirani, 1996), Least Angle Regression (LAR) (Efron et al., 2004), their group versions Group Lasso (GLasso) and Group Least Angle Regression (GLAR) (Yuan and Lin, 2006), Multiple Kernel Learning (MKL) (Bach et al., 2004; Evgeniou et al., 2005; Pillonetto et al.). Methods based on hierarchical Bayesian models have also been considered, including Automatic Relevance Determination (ARD) (Mackay, 1994), the Relevance Vector Machine (RVM) (Tipping, 2001), and the exponential hyperprior (Chiuso and Pillonetto, 2010b, 2012). The Bayesian approach considered by Chiuso and Pillonetto (2010b, 2012) is intimately related to that of Mackay (1994) and Tipping (2001); in fact the exponential hyperprior algorithm proposed by Chiuso and Pillonetto (2010b, 2012) is a penalized version of ARD (PARD) in which the prior on the groups $\theta^{(i)}$ is adapted to the structural properties of dynamical systems. A variational approach based on the golden standard spike and slab prior, also called two-groups prior (Efron, 2008), has been also recently proposed by Titsias and Lzaro-Gredilla (2011).

An interesting series of papers (Wipf and Rao, 2007; Wipf and Nagarajan, 2007; Wipf et al., 2011) provide a nice link between penalized regression problems like Lasso, also called type-I methods, and Bayesian methods (like RVM, Tipping, 2001 and ARD, Mackay, 1994) with hierarchical hyperpriors where the *hyperparameters* are estimated via maximizing the marginal likelihood and then inserted in the Bayesian model following the Empirical Bayes paradigm (Maritz and Lwin, 1989); these latter methods are also known as type-II methods (Berger, 1985). Note that this Empirical Bayes paradigm has also been recently used in the context of System Identification (Pillonetto and De Nicolao, 2010; Pillonetto et al., 2011; Chen et al., 2011).

Wipf and Nagarajan (2007) and Wipf et al. (2011) argue that type-II methods have advantages over type-I methods; some of these advantages are related to the fact that, under suitable assumptions, the former can be written in the form of type-I with the addition of a non-separable penalty term (a function $g(x_1, .., x_n)$ is non-separable if it cannot be written as $g(x_1, .., x_n) = \sum_{i=1}^{n} h(x_i)$). The analysis of Wipf et al. (2011) also suggests that in the low noise regime the type-II approach results in a "tighter" approximation to the ℓ_0 norm. This is supported by experimental evidence showing that these Bayesian approaches perform

1. Impulse responses may, in principle, be infinite dimensional.

well in practice. Our experience is that the approach based on the marginal likelihood is particularly robust w.r.t. noise regardless of the "correctness" of the Bayesian prior.

Motivated by the strong performance of the exponential hyperprior approach introduced in the dynamic network identification scenario (Chiuso and Pillonetto, 2010b, 2012), we provide some new insights clarifying the above issues. The main contributions are as follows:

(i) We first provide some motivating examples which illustrate the superiority of PARD (and also of ARD) over GLasso both in terms of selection (i.e., detecting block of zeros in θ) as well as in estimation (i.e., reconstructing the non zero blocks).

(ii) Theoretical findings explaining the reasons underlying the superiority of PARD over GLasso are then provided. In particular, all the methods are compared in terms of optimality (KKT) conditions, and tradeoffs between sparsity and shrinkage are studied.

(iii) We then consider a non-Bayesian point of view, in which the estimation error is measured in terms of the Mean Squared Error, in the vein of Stein-estimators (James and Stein, 1961; Efron and Morris, 1973; Stein, 1981). The properties of Empirical Bayes estimators, which form the basis of the computational schemes, are studied in terms of their Mean Square Error properties; this is first established in the simplest case of orthogonal regressors and then extended to more general cases allowing for the regressors to be realizations from (possibly correlated) stochastic processes. This, of course, is of paramount importance for the system identification scenario studied by Chiuso and Pillonetto (2010b, 2012).

    Our analysis avoids assumptions on the correctness of the priors which define the stochastic model and clarifies why PARD is likely to provide sparser and more accurate estimates in comparison with GLasso (MKL). As a consequence of this analysis, our study clarifies the asymptotic properties of ARD.

Before we proceed with these results, we need to establish a common framework for these estimators (GLasso/MKL and PARD); this mostly uses results from the literature, which are recalled without proof in order to make the paper as self contained as possible.

The paper is organized as follows. In Section 2 we provide the problem statement while in Section 3 PARD and GLasso (MKL) are introduced in a Bayesian framework. Section 4 illustrates the advantages of PARD over GLasso using a simple example and two Monte Carlo studies. In Section 5 the Mean Squared Error properties of the Empirical Bayes estimators are studied, including their asymptotic behavior. Some conclusions end the paper while the Appendix gathers the proofs of the main results.

2. Problem Statement

We consider a linear model $y = G\theta + v$ where the explanatory factors G used to predict y are grouped (and non-overlapping). As such we partition θ into p sub-vectors $\theta^{(i)}$, $i = 1, \dots, p$, so that

$$\theta = [\theta^{(1)\top} \quad \theta^{(2)\top} \quad \dots \quad \theta^{(p)\top}]^\top.$$

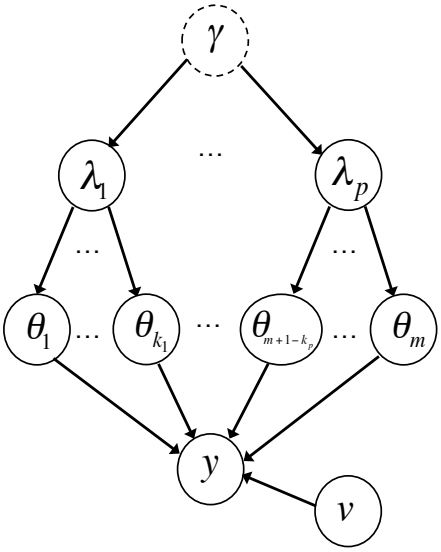


Figure 1: Bayesian networks describing the stochastic model for group sparse estimation

For $i = 1, \ldots, p$, assume that the sub-vector $\theta^{(i)}$ has dimension k_i so that $m = \sum_{i=1}^{p} k_i$. Next, conformally partition the matrix $G = [G^{(1)}, \ldots, G^{(p)}]$ to obtain the measurement model

$$y = G\theta + v = \sum_{i=1}^{p} G^{(i)}\theta^{(i)} + v. \tag{1}$$

In what follows, we assume that θ is *block sparse* in the sense that many of the blocks $\theta^{(i)}$ are zero, that is, with all of their components equal to zero, or have a negligible effect on y.

Our problem is to estimate θ from y while also detecting the null blocks of $\theta^{(i)}$.

3. Estimators Considered

The purpose of this Section is to place the estimators we consider (GLasso/MKL and PARD) in a common framework that unifies the analysis. The content of the section is a collection of results taken from the literature which are stated without proof; the readers are referred to previous works for details which are not relevant to our paper's goal.

3.1 Bayesian Model for Sparse Estimation

Figure 1 provides a hierarchical representation of a probability density function useful for establishing a connection between the various estimators considered in this paper. In particular, in the Bayesian network of Figure 1, nodes and arrows are either dotted or solid depending on whether the quantities/relationships are deterministic or stochastic, respectively. Here, λ denotes a vector whose components $\{\lambda_i\}_{i=1}^{p}$ are independent and identically distributed exponential random variables with probability density

$$p_\gamma(\lambda_i) = \gamma e^{-\gamma\lambda_i}\chi(\lambda_i)$$

where γ is a positive scalar and

$$\chi(t) = \begin{cases} 1, & t \geq 0 \\ 0, & \text{elsewhere.} \end{cases}$$

In addition, let $\mathcal{N}(\mu, \Sigma)$ be the Gaussian density of mean μ and covariance Σ while, given a generic k, we use I_k to denote the $k \times k$ identity matrix. Then, conditional on λ, the blocks $\theta^{(i)}$ of the vector θ are all mutually independent and each block is zero-mean Gaussian with covariance $\lambda_i I_{k_i}$, $i = 1, .., p$, that is,

$$\theta^{(i)}|\lambda_i \sim \mathcal{N}(0, \lambda_i I_{k_i}).$$

The measurement noise is also Gaussian, that is,

$$v \sim \mathcal{N}(0, \sigma^2 I_n).$$

3.2 Penalized ARD (PARD)

We introduce a sparse estimator, denoted by PARD in the sequel, cast in the framework of the Type II Bayesian estimators and consisting of a penalized version of ARD (Mackay, 1994; Tipping, 2001; Wipf and Nagarajan, 2007). It is derived from the Bayesian network depicted in Figure 1 as follows. First, the marginal density of λ is optimized, that is, we compute

$$\hat{\lambda} = \arg\max_{\lambda \in \mathbb{R}_+^p} \int_{\mathbb{R}^m} p(\theta, \lambda|y) d\theta.$$

Then, using an empirical Bayes approach, we obtain $\mathbb{E}[\theta|y, \lambda = \hat{\lambda}]$, that is, the minimum variance estimate of θ with λ taken as known and set to its estimate. The structure of the estimator is detailed in the following proposition (whose proof is straightforward and therefore omitted).

Proposition 1 (PARD) *Define*

$$\begin{aligned} \Sigma_y(\lambda) &:= G\Lambda G^\top + \sigma^2 I, & (2) \\ \Lambda &:= \mathit{blockdiag}(\{\lambda_i I_{k_i}\}). & (3) \end{aligned}$$

Then, the estimator $\hat{\theta}_{PA}$ of θ obtained from PARD is given by

$$\begin{aligned} \hat{\lambda} &= \arg\min_{\lambda \in \mathbb{R}_+^p} \frac{1}{2} \log\det(\Sigma_y(\lambda)) + \frac{1}{2} y^\top \Sigma_y^{-1}(\lambda) y + \gamma \sum_{i=1}^p \lambda_i, & (4) \\ \hat{\theta}_{PA} &= \hat{\Lambda} G^\top (\Sigma_y(\hat{\lambda}))^{-1} y. & (5) \end{aligned}$$

where $\hat{\Lambda}$ is defined as in (3) with each λ_i replaced by the i-th component of $\hat{\lambda}$ in (4).

■

One can see from (4) and (5) that the proposed estimator reduces to ARD if $\gamma = 0$.[2] In this case, the special notation $\hat{\theta}_A$ is used to denote the resulting estimator, that is,

$$\hat{\lambda} = \arg\min_{\lambda \in \mathbb{R}^p_+} \frac{1}{2}\log\det(\Sigma_y(\lambda)) + \frac{1}{2}y^\top \Sigma_y^{-1}(\lambda) y, \tag{6}$$

$$\hat{\theta}_A = \hat{\Lambda} G^\top (\Sigma_y(\hat{\lambda}))^{-1} y \tag{7}$$

where Σ_y is defined in (2), and $\hat{\Lambda}$ is defined as in (3) with each λ_i replaced by the i-th component of the $\hat{\lambda}$ in (6).

Observe that the objective in (4) is not convex in λ. Letting the vector μ denote the dual vector for the constraint $\lambda \geq 0$, the Lagrangian is given by

$$L(\lambda, \mu) := \tfrac{1}{2}\log\det(\Sigma_y(\lambda)) + \tfrac{1}{2}y^\top \Sigma_y(\lambda)^{-1} y + \gamma \mathbf{1}^\top \lambda - \mu^\top \lambda.$$

Using the fact that

$$\begin{aligned} \partial_{\lambda_i} L(\lambda, \mu) &= \frac{1}{2}\mathrm{tr}\left(G^{(i)\top} \Sigma_y(\lambda)^{-1} G^{(i)}\right) \\ &- \frac{1}{2} y^\top \Sigma_y(\lambda)^{-1} G^{(i)} G^{(i)\top} \Sigma_y(\lambda)^{-1} y + \gamma \ - \mu_i, \end{aligned}$$

we obtain the following KKT conditions for (4).

Proposition 2 (KKT for PARD) *The necessary conditions for λ to be a solution of (4) are*

$$\begin{aligned} &\Sigma_y = \sigma^2 I + \textstyle\sum_{i=1}^p \lambda_i G^{(i)} G^{(i)\top}, \\ &W \Sigma_y = I, \\ &tr\left(G^{(i)\top} W G^{(i)}\right) - \|G^{(i)\top} W y\|_2^2 + 2\gamma - 2\mu_i = 0, \quad i = 1, \ldots, p, \\ &\mu_i \lambda_i = 0, \quad i = 1, \ldots, p, \\ &0 \leq \mu, \ \lambda. \end{aligned} \tag{8}$$

3.3 Group Lasso (GLasso) and Multiple Kernel Learning (MKL)

A leading approach for the block sparsity problem is the Group Lasso (GLasso) (Yuan and Lin, 2006) which determines the estimate of θ as the solution of the following convex problem

$$\hat{\theta}_{GL} = \arg\min_{\theta \in \mathbb{R}^m} \frac{(y - G\theta)^\top (y - G\theta)}{2\sigma^2} + \gamma_{GL} \sum_{i=1}^p \|\theta^{(i)}\|, \tag{9}$$

where $\|\cdot\|$ denotes the classical Euclidean norm. Now, let ϕ be the Gaussian vector with independent components of unit variance such that

$$\theta_i = \sqrt{\lambda_i}\ \phi_i. \tag{10}$$

We partition ϕ conformally with θ, that is,

$$\phi = \left[\phi^{(1)\top} \quad \phi^{(2)\top} \quad \ldots \quad \phi^{(p)\top}\right]^\top. \tag{11}$$

2. Strictly speaking, what is called ARD in this paper corresponds to a group version of the original estimator discussed in Mackay (1994). A perfect correspondence is obtained when the dimension of each block is equal to one, that is, $k_i = 1\ \forall i$.

Then, interestingly, GLasso can be derived from the same Bayesian model in Figure 1 underlying PARD considering ϕ and λ as unknown variables and computing their maximum a posteriori (MAP) estimates. This is illustrated in the following proposition which is just a particular instance of the known relationship between regularization on kernel weights and block-norm based regularization. In particular, it establishes the known equivalence between GLasso and Multiple Kernel Learning (MKL) when linear models of the form (1) are considered, see the more general Theorem 1 of Tomioka and Suzuki (2011) for other details.

Proposition 3 (GLasso and its equivalence with MKL) *Consider the joint density of ϕ and λ conditional on y induced by the Bayesian network in Figure 1. Let $\hat{\lambda}$ and $\hat{\phi}$ denote, respectively, the maximum a posteriori estimates of λ and ϕ (obtained by optimizing their joint density). Then, for every γ_{GL} in (9) there exists γ such that the following equalities hold*

$$\hat{\lambda} = \arg\min_{\lambda \in \mathbb{R}^p_+} \frac{y^\top (\Sigma_y(\lambda))^{-1} y}{2} + \gamma \sum_{i=1}^{p} \lambda_i, \tag{12}$$

$$\begin{aligned} \hat{\phi}^{(i)} &= \sqrt{\hat{\lambda}_i} G^{(i)\top} (\Sigma_y(\hat{\lambda}))^{-1} y, \\ \hat{\theta}^{(i)}_{GL} &= \sqrt{\hat{\lambda}_i} \hat{\phi}^{(i)}. \end{aligned} \tag{13}$$

We warn the reader that MKL is more general than GLasso since it also embodies estimation in infinite dimensional models; yet in this paper we use interchangeably the nomenclature GLasso and MKL since they are equivalent for the considered model class.

Comparing Propositions 1 and 3, one can see that the sole difference between PARD and GLasso relies on the estimator for λ. In particular, notice that the objectives (12) and (4) differ only in the term $\frac{1}{2}\log\det(\Sigma_y)$ appearing in the PARD objective (4). This is the component that makes problem (4) non-convex but also the term that forces PARD to favor sparser solutions than GLasso (MKL), making the marginal density of λ more concentrated around zero. On the other hand, (12) is a convex optimization problem whose associated KKT conditions are reported in the following proposition.

Proposition 4 (KKT for GLasso and MKL) *The necessary and sufficient conditions for λ to be a solution of (12) are*

$$\begin{aligned} &K(\lambda) = \textstyle\sum_{i=1}^{p} \lambda_i G^{(i)} G^{(i)\top}, \\ &\Sigma_y = K(\lambda) + \sigma^2 I, \\ &W\Sigma_y = I, \\ &-\|G^{(i)\top} W y\|_2^2 + 2\gamma - 2\mu_i = 0, \quad i = 1, \ldots, p, \\ &\mu_i \lambda_i = 0, \quad i = 1, \ldots, p, \\ &0 \leq \mu,\ \lambda. \end{aligned} \tag{14}$$

■

Remark 5 (The LASSO case) *When all the blocks are one-dimensional, the estimator (9) reduces to Lasso and we denote the regularization parameter and the estimate by γ_L and $\hat{\theta}_L$, respectively. In this case, it is possible to obtain a derivation through marginalization. In fact, given the Bayesian network in Figure 1 with all the $k_i = 1$ and letting*

$$\hat{\theta} = \arg\max_{\theta \in \mathbb{R}^m} \int_{\mathbb{R}^m_+} p(\theta, \lambda | y) d\lambda,$$

it follows from Section 2 in Park and Casella (2008) that $\hat{\theta} = \hat{\theta}_L$ provided that $\gamma_L = \sqrt{2\gamma}$.

4. Comparing PARD And GLasso (MKL): Motivating Examples

In this section, we present a sparsity vs. shrinkage example, and a Monte Carlo simulation to demonstrate advantages of the PARD estimator over GLasso.

4.1 Sparsity vs. Shrinkage: A Simple Experiment

It is well known that the ℓ_1 penalty in Lasso tends to induce an excessive shrinkage of "large" coefficients in order to obtain sparsity. Several variations have been proposed in the literature in order to overcome this problem, including the so called *Smoothly-Clipped-Absolute-Deviation* (SCAD) estimator (Fan and Li, 2001) and re-weighted versions of ℓ_1 like the *adaptive Lasso* (Zou, 2006). We now study the tradeoffs between sparsity and shrinking for PARD. By way of introduction to the more general analysis in the next section, we first compare the sparsity conditions for PARD and GLasso (or, equivalently, MKL) in a simple, yet instructive, two group example. In this example, it is straightforward to show that PARD guarantees a more favorable tradeoff between sparsity and shrinkage, in the sense that it induces greater sparsity with the same shrinkage (or, equivalently, for a given level of sparsity it guarantees less shrinkage).

Consider two groups of dimension 1, that is,

$$y = G^{(1)}\theta^{(1)} + G^{(2)}\theta^{(2)} + v \quad y \in \mathbb{R}^2,\ \theta^{(1)},\ \theta^{(2)} \in \mathbb{R},$$

where $G^{(1)} = [1 \quad \delta]^\top$, $G^{(2)} = [0 \quad 1]^\top$, $v \sim \mathcal{N}(0, \sigma^2)$. Assume that the true parameter $\bar{\theta}$ satisfies: $\bar{\theta}^{(1)} = 0$, $\bar{\theta}^{(2)} = 1$. Our goal is to understand how the hyperparameter γ influences sparsity and the estimates of $\theta^{(1)}$ and $\theta^{(2)}$ using PARD and GLasso. In particular, we would like to determine which values of γ guarantee that $\hat{\theta}^{(1)} = 0$ and how the estimator $\hat{\theta}^{(2)}$ varies with γ. These questions can be answered by using the KKT conditions obtained in Propositions 2 and 4.

Let $y := [y_1\ y_2]^\top$. By (8), the necessary conditions for $\hat{\lambda}_1^{PA} = 0$ and $\hat{\lambda}_2^{PA} \geq 0$ to be the hyperparameter estimators for the PARD estimator (for fixed $\gamma = \gamma_{PA}$) are

$$\begin{aligned} &2\gamma_{PA} \geq \left[\tfrac{y_1}{\sigma^2} + \tfrac{\delta y_2}{\sigma^2 + \hat{\lambda}_2^{PA}}\right]^2 - \left[\tfrac{1}{\sigma^2} + \tfrac{\delta}{\sigma^2 + \hat{\lambda}_2^{PA}}\right] \quad \text{and} \\ &\hat{\lambda}_2^{PA} = \max\left\{\tfrac{-1+\sqrt{1+8\gamma_{PA}y_2^2}}{4\gamma_{PA}} - \sigma^2,\ 0\right\}. \end{aligned} \tag{15}$$

Similarly, by (14), the same conditions for $\hat{\lambda}_1^{GL} = 0$ and $\hat{\lambda}_2^{GL} \geq 0$ to be the estimators obtained using GLasso read as (for fixed $\gamma = \gamma_{GL}$):

$$\begin{aligned} 2\gamma_{GL} &\geq \left[\frac{y_1}{\sigma^2} + \frac{\delta y_2}{\sigma^2 + \hat{\lambda}_2^{GL}}\right]^2 \quad \text{and} \\ \hat{\lambda}_2^{GL} &= \max\left\{\frac{|y_2|}{\sqrt{2\gamma_{GL}}} - \sigma^2, 0\right\}. \end{aligned} \tag{16}$$

Note that it is always the case that the lower bound for γ_{GL} is strictly greater than the lower bound for γ_{PA} and that $\hat{\lambda}_2^{PA} \leq \hat{\lambda}_2^{GL}$ when $\gamma_{PA} = \gamma_{GL}$, where the inequality is strict whenever $\hat{\lambda}_2^{GL} > 0$. The corresponding estimators for $\hat{\theta}^{(1)}$ and $\hat{\theta}^{(2)}$ are

$$\hat{\theta}_{PA}^{(1)} = \hat{\theta}_{GL}^{(1)} = 0,$$

$$\hat{\theta}_{PA}^{(2)} = \frac{\hat{\lambda}_2^{PA} y_2}{\sigma^2 + \hat{\lambda}_2^{PA}} \qquad \text{and} \qquad \hat{\theta}_{GL}^{(2)} = \frac{\hat{\lambda}_2^{GL} y_2}{\sigma^2 + \hat{\lambda}_2^{GL}} \ .$$

Hence, $|\hat{\theta}_{PA}^{(2)}| < |\hat{\theta}_{GL}^{(2)}|$ whenever $y_2 \neq 0$ and $\hat{\lambda}_2^{GL} > 0$. However, it is clear that the lower bounds on γ in (15) and (16) indicate that γ_{GL} needs to be larger than γ_{PA} in order to set $\hat{\lambda}_1^{GL} = 0$ (and hence $\hat{\theta}_{GL}^{(1)} = 0$). Of course, having a larger γ tends to yield smaller $\hat{\lambda}_2$ and hence more shrinking on $\hat{\theta}^{(2)}$. This is illustrated in Figure 2 where we report the estimators $\hat{\theta}_{PA}^{(2)}$ (solid) and $\hat{\theta}_{GL}^{(2)}$ (dotted) for $\sigma^2 = 0.005$, $\delta = 0.5$. The estimators are arbitrarily set to zero for the values of γ which do not yield $\hat{\theta}^{(1)} = 0$. In particular from (15) and (16) we find that PARD sets $\hat{\theta}_{PA}^{(1)} = 0$ for $\gamma_{PA} > 5$ while GLasso sets $\hat{\theta}_{GL}^{(1)} = 0$ for $\gamma_{GL} > 20$. In addition, it is clear that MKL tends to yield greater shrinkage on $\hat{\theta}_{GL}^{(2)}$ (recall that $\bar{\theta}^{(2)} = 1$).

4.2 Monte Carlo Studies

We consider two Monte Carlo studies of 1000 runs. For each run a data set of size $n = 100$ is generated using the linear model (1) with $p = 10$ groups, each composed of $k_i = 4$ parameters. For each run, 5 of the groups $\theta^{(i)}$ are set to zero, one is always taken different from zero while each of the remaining 4 groups $\theta^{(i)}$ are set to zero with probability 0.5. The components of every block $\theta^{(i)}$ not set to zero are independent realizations from a uniform distribution on $[-a_i, a_i]$ where a_i is an independent realization (one for each block) from a uniform distribution on $[0, 100]$. The value of σ^2 is the variance of the noiseless output divided by 25. The noise variance is estimated at each run as the sum of the residuals from the least squares estimate divided by $n - m$. The two experiments differ in the way the columns of G are generated at each run.

1. In the first experiment, the entries of G are independent realizations of zero mean unit variance Gaussian noise.

2. In the second experiment the columns of G are correlated, being defined at every run by

$$\begin{aligned} &G_{i,j} = G_{i,j-1} + 0.2 v_{i,j-1}, \quad i = 1, .., n, \quad j = 2, .., m, \\ &v_{i,j} \sim \mathcal{N}(0, 1) \end{aligned}$$

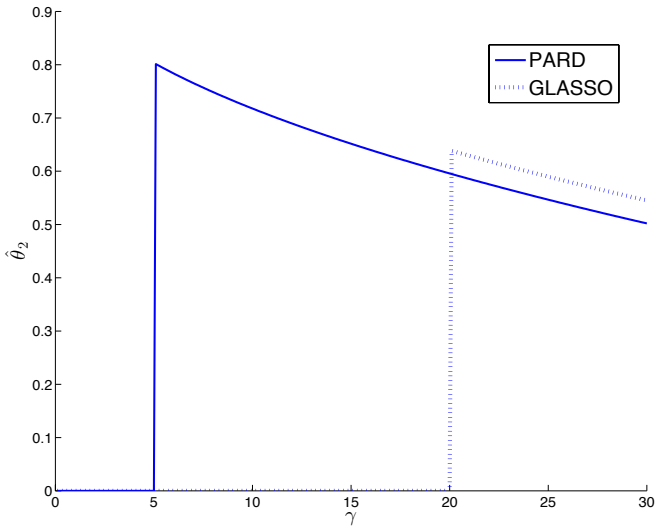


Figure 2: Estimators $\hat{\theta}^{(2)}$ as a function of γ. The curves are plotted only for the values of γ which yield also $\hat{\theta}^{(1)} = 0$ (different for PARD ($\gamma_{PA} > 5$) and MKL ($\gamma_{GL} > 20$)).

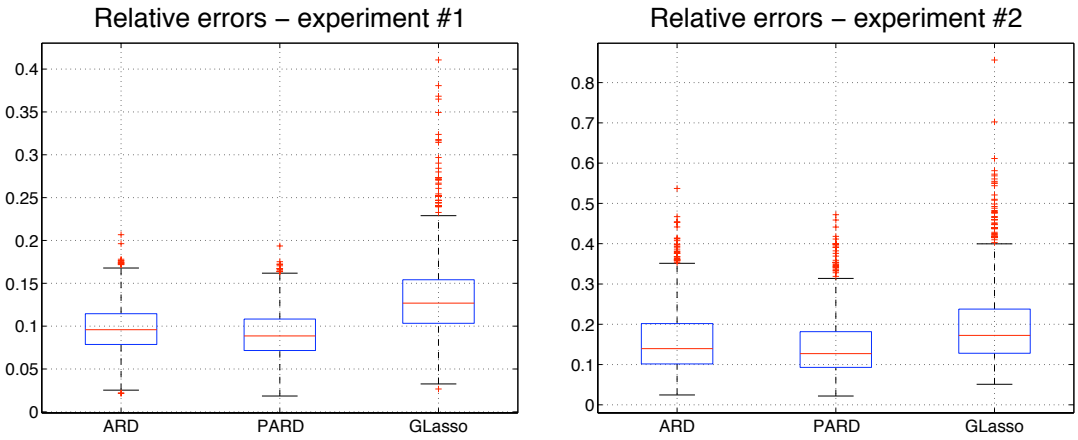


Figure 3: Boxplot of the relative errors in the reconstruction of θ obtained by the 2 non-convex estimators ARD and PARD and by the convex estimator GLasso (MKL) after the 1000 Monte Carlo runs in Experiment #1 (left panel) and #2 (right panel).

where $v_{i,j}$ are i.i.d. (as i and j vary) zero mean unit variance Gaussian and $G_{i,1}$ are i.i.d. zero mean unit variance Gaussian random variables. Note that correlated inputs renders the estimation problem more challenging.

Define $\hat{\kappa} \in \mathbb{R}$ as the optimizer of the ARD objective (6) under the constraint $\kappa = \lambda_1 = \ldots = \lambda_p$. Then, we define the following 3 estimators.

- **ARD**. The estimate θ_A is obtained by (6,7) using $\lambda_1 = \ldots = \lambda_p = \hat{\kappa}$ as starting point to solve (7) .
- **PARD**. The estimate θ_{PA} is obtained by (4,5) using cross validation to determine the regularization parameter γ. In particular, data are split into a training and validation set of equal size and the grid used by the cross validation procedure to select γ contains 30 elements logarithmically distributed between $10^{-2} \times \hat{\kappa}^{-1}$ and $10^{4} \times \hat{\kappa}^{-1}$. For each value of γ, (6) is solved using $\lambda_1 = \lambda_2 = \ldots = \lambda_p = \hat{\kappa}$ as starting point. Finally, θ_{PA} is obtained using the full data set fixing the regularization parameter to its estimate.
- **GLasso (MKL)**. The estimate θ_{GL} is obtained by (12-13) using the same cross validation strategy adopted by PARD to determine γ.

The three estimators described above are compared using the two performance indexes listed below:

1. Relative error: this is computed at each run as

$$\frac{\|\hat{\theta} - \theta\|}{\|\theta\|}$$

   where $\hat{\theta}$ is the estimator of θ.

2. Percentage of the blocks equal to zero correctly set to zero by the estimator after the 1000 runs.

The left and right panel of Figure 3 displays the boxplots of the 1000 relative errors obtained by the three estimators in the first and second experiment, respectively. The average relative error is also reported in Table 1. It is apparent that the performance of PARD and ARD is similar and that both of these non convex estimators outperform GLasso. Interestingy, in both of the experiments ARD and PARD return a reconstruction error smaller that that achieved by GLasso in more than 900 out of the 1000 runs.

In Table 2 we report the sparsity index. One can see that PARD exhibits the best performance, setting almost 75% of the blocks correctly to zero in the first and second experiment, respectively, while the performance of ARD is close to 67%. In contrast, GLasso (MKL) correctly set to zero no more than 40% of the blocks in each experiment.

Remark 6 (Projected Quasi-Newton Method) *We now comment on the optimization of (4). The same arguments reported below also apply to the objectives (6) and (12) which are just simplified versions of (4).*

	ARD	PARD	GLasso
Experiment #1	0.097	0.090	0.138
Experiment #2	0.151	0.144	0.197

Table 1: Comparison with MKL/GLasso (section 4.2). Average relative errors obtained by the three estimators.

	ARD	PARD	GLasso
Experiment #1	66.7%	74.5%	35.5%
Experiment #2	66.6%	74.6%	39.7%

Table 2: Comparison with MKL/GLasso (section 4.2). Percentage of the $\theta^{(i)}$ equal to zero correctly set to zero by the four estimators.

We notice that (4) is a differentiable function of λ. The computation of its derivative requires a one time evaluation of the matrices $G^{(i)}G^{(i)^\top}$, $i = 1, \ldots, p$. However, for each new value of λ, the inverse of the matrix $\Sigma_y(\lambda)$ also needs to be computed. Hence, the evaluation of the objective and its derivative may be costly since it requires computing the inverse of a possibly large matrix as well as large matrix products. On the other hand, the dimension of the parameter vector λ can be small, and projection onto the feasible set is trivial. We experimented with several methods available in the Matlab package `minConf` *to optimize (4). In these experiments, the fastest method was the limited memory projected quasi-Newton algorithm detailed in Schmidt et al. (2009). It uses L-BFGS updates to build a diagonal plus low-rank quadratic approximation to the function, and then uses the Projected Quasi-Newton Method to minimize a quadratic approximation subject to the original constraints to obtain a search direction. A backtracking line search is applied to this direction terminating at a step-size satisfying a Armijo-like sufficient decrease condition. The efficiency of the method derives in part from the simplicity of the projections onto the feasible region. We have also implemented the re-weighted method described by Wipf and Nagarajan (2007). In all the numerical experiments described above, we have assessed that it returns results virtually identical to those achieved by our method, with a similar computational effort. It is worth recalling that both the projected quasi-Newton method and the re-weighted approach guarantee convergence only to a stationary point of the objective.*

4.3 Concluding Remarks

The results in this section suggest that, when using GLasso, a suitable regularization parameter γ which does not induce oversmoothing (large bias) in $\hat{\theta}$ is not sufficiently large to induce "enough" sparsity. This drawback does not affect the nonconvex estimators. In addition, PARD and ARD seem to have the additional advantage of selecting the regularization parameters leading to more favorable Mean Squared Error (MSE) properties for the

reconstruction of the non zero blocks. The rest of the paper will be devoted to derivation of theoretical arguments supporting the intuition gained from these examples.

5. Mean Squared Error Properties of PARD and GLasso (MKL)

In this Section we evaluate the performance of an estimator $\hat{\theta}$ using its MSE, that is, the expected quadratic loss

$$\operatorname{tr}\left[\mathbb{E}\left[\left(\hat{\theta}-\theta\right)\left(\hat{\theta}-\theta\right)^{\top}\,\middle|\,\lambda,\theta=\bar{\theta}\right]\right],$$

where $\bar{\theta}$ is the true but unknown value of θ. When we speak about "Bayes estimators" we think of estimators of the form $\hat{\theta}(\lambda) := \mathbb{E}\left[\theta \,|\, y, \lambda\right]$ computed using the probabilistic model Figure 1 with γ fixed.

5.1 Properties Using "Orthogonal" Regressors

We first derive the MSE formulas under the simplifying assumption of *orthogonal* regressors ($G^{\top}G = nI$) and show that the Empirical Bayes estimator converges to an optimal estimator in terms of its MSE. This fact has close connections to the so called *Stein estimators* (James and Stein, 1961; Stein, 1981; Efron and Morris, 1973). The same optimality properties are attained, asymptotically, when the columns of G are realizations of uncorrelated processes having the same variance. This is of interest in the system identification scenario considered by Chiuso and Pillonetto (2010a,b, 2012) since it arises when one performs identification with i.i.d. white noises as inputs. We then consider the more general case of correlated regressors (see Section 5.2) and show that essentially the same result holds for a weighted version of the MSE.

In this section, it is convenient to introduce the following notation:

$$\mathbb{E}_v[\,\cdot\,] := \mathbb{E}[\,\cdot\,|\,\lambda,\,\theta=\bar{\theta}] \quad \text{and} \quad \operatorname{Var}_v[\,\cdot\,] := \mathbb{E}[\,\cdot\,|\,\lambda,\,\theta=\bar{\theta}].$$

We now report an expression for the MSE of the Bayes estimators $\hat{\theta}(\lambda) := \mathbb{E}\left[\theta \,|\, y, \lambda\right]$ (the proof follows from standard calculations and is therefore omitted).

Proposition 7 *Consider the model* (1) *under the probabilistic model described in Figure 1(b). The Mean Squared Error of the Bayes estimator* $\hat{\theta}(\lambda) := \mathbb{E}\left[\theta|y,\lambda\right]$ *given* λ *and* $\theta = \bar{\theta}$ *is*

$$\begin{aligned} MSE(\lambda) &= tr\left[\mathbb{E}_v\left[(\hat{\theta}(\lambda)-\theta)(\hat{\theta}(\lambda)-\theta)^{\top}\right]\right] \\ &= tr\left[\sigma^2\left(G^{\top}G+\sigma^2\Lambda^{-1}\right)^{-1}\left(G^{\top}G+\sigma^2\Lambda^{-1}\bar{\theta}\bar{\theta}^{\top}\Lambda^{-1}\right)\left(G^{\top}G+\sigma^2\Lambda^{-1}\right)^{-1}\right] \quad (17) \\ &= tr\left[\sigma^2\left(\Lambda G^{\top}G+\sigma^2\right)^{-1}\left(\Lambda G^{\top}G\Lambda+\sigma^2\bar{\theta}\bar{\theta}^{\top}\right)\left(G^{\top}G\Lambda+\sigma^2\right)^{-1}\right]. \end{aligned}$$

We can now minimize the expression for $MSE(\lambda)$ given in (17) with respect to λ to obtain the optimal minimum mean squared error estimator. In the case where $G^{\top}G = nI$ this computation is straightforward and is recorded in the following proposition.

Corollary 8 *Assume that $G^\top G = nI$ in Proposition 7. Then MSE(λ) is globally minimized by choosing*

$$\lambda_i = \lambda_i^{opt} := \frac{\|\bar{\theta}^{(i)}\|^2}{k_i}, \quad i = 1, \dots, p.$$

Next consider the Maximum a Posteriori estimator of λ again under the simplifying assumption $G^\top G = nI$. Note that, under the noninformative prior ($\gamma = 0$), this Maximum a Posteriori estimator reduces to the standard Maximum (marginal) Likelihood approach to estimating the prior distribution of θ. Consequently, we continue to call the resulting procedure Empirical Bayes (a.k.a. Type-II Maximum Likelihood (Berger, 1985)).

Proposition 9 *Consider model* (1) *under the probabilistic model described in Figure 1(b), and assume that $G^\top G = nI$. Then the estimator of λ_i obtained by maximizing the marginal posterior $\mathbf{p}(\lambda|y)$,*

$$\{\hat{\lambda}_1(\gamma), ..., \hat{\lambda}_p(\gamma)\} := \arg\max_{\lambda \in \mathbb{R}_+^p} \mathbf{p}(\lambda|y) = \arg\max_{\lambda \in \mathbb{R}_+^p} \int \mathbf{p}(y, \theta|\lambda)\mathbf{p}_\gamma(\lambda)\, d\theta,$$

is given by

$$\hat{\lambda}_i(\gamma) = \max\left(0, \frac{1}{4\gamma}\left[\sqrt{k_i^2 + 8\gamma\|\hat{\theta}_{LS}^{(i)}\|^2} - \left(k_i + \frac{4\sigma^2\gamma}{n}\right)\right]\right), \tag{18}$$

where

$$\hat{\theta}_{LS}^{(i)} = \frac{1}{n}\left(G^{(i)}\right)^\top y$$

is the Least Squares estimator of the $i-$th block $\theta^{(i)}$. As $\gamma \to 0$ ($\gamma = 0$ corresponds to an improper flat prior) the expression (18) *yields:*

$$\lim_{\gamma \to 0} \hat{\lambda}_i(\gamma) = \max\left(0, \frac{\|\hat{\theta}_{LS}^{(i)}\|^2}{k_i} - \frac{\sigma^2}{n}\right).$$

In addition, the probability $\mathbb{P}[\hat{\lambda}_i(\gamma) = 0 \,|\, \theta = \bar{\theta}]$ of setting $\hat{\lambda}_i = 0$ is given by

$$\mathbb{P}[\hat{\lambda}_i(\gamma) = 0 \,|\, \theta = \bar{\theta}] = \mathbb{P}\left[\chi^2\left(k_i, \|\bar{\theta}^{(i)}\|^2 \frac{n}{\sigma^2}\right) \le \left(k_i + 2\gamma\frac{\sigma^2}{n}\right)\right], \tag{19}$$

where $\chi^2(d, \mu)$ denotes a noncentral χ^2 random variable with d degrees of freedom and noncentrality parameter μ.

Note that the expression of $\hat{\lambda}_i(\gamma)$ in Proposition 9 has the form of a "saturation". In particular, for $\gamma = 0$, we have

$$\hat{\lambda}_i(0) = \max(0, \hat{\lambda}_i^*), \quad \text{where} \quad \hat{\lambda}_i^* := \frac{\|\hat{\theta}_{LS}^{(i)}\|^2}{k_i} - \frac{\sigma^2}{n}. \tag{20}$$

The following proposition shows that the "unsaturated" estimator $\hat{\lambda}_i^*$ is an unbiased and consistent estimator of λ_i^{opt} which minimizes the Mean Squared Error while $\hat{\lambda}_i(0)$ is only asymptotically unbiased and consistent.

Corollary 10 *Under the assumption $G^\top G = nI$, the estimator of $\hat{\lambda}^* := \{\lambda_1^*, .., \lambda_p^*\}$ in* (20) *is an unbiased and mean square consistent estimator of λ^{opt} which minimizes the Mean Squared Error, while $\hat{\lambda}(0) := \{\lambda_1(0), .., \lambda_p(0)\}$ is asymptotically unbiased and consistent, that is:*

$$\mathbb{E}[\hat{\lambda}_i^* \,|\, \theta = \bar{\theta}] = \lambda_i^{opt} \quad \lim_{n\to\infty} \mathbb{E}[\hat{\lambda}_i(0) \,|\, \theta = \bar{\theta}] = \lambda_i^{opt}$$

and

$$\lim_{n\to\infty} \hat{\lambda}_i^* \overset{m.s.}{=} \lambda_i^{opt} \quad \lim_{n\to\infty} \hat{\lambda}_i(0) \overset{m.s.}{=} \lambda_i^{opt} \tag{21}$$

where $\overset{m.s.}{=}$ denotes convergence in mean square.

Remark 11 *Note that if $\bar{\theta}^{(i)} = 0$, the optimal value λ_i^{opt} is zero. Hence* (21) *shows that asymptotically $\hat{\lambda}_i(0)$ converges to zero. However, in this case, it is easy to see from* (19) *that*

$$\lim_{n\to\infty} \mathbb{P}[\hat{\lambda}_i(0) = 0 \,|\, \theta = \bar{\theta}] < 1.$$

There is in fact no contradiction between these two statements because one can easily show that for all $\epsilon > 0$,

$$\mathbb{P}[\hat{\lambda}_i(0) \in [0, \epsilon) \,|\, \theta = \bar{\theta}] \overset{n\to\infty}{\longrightarrow} 1.$$

In order to guarantee that $\lim_{n\to\infty} \mathbb{P}[\hat{\lambda}_i(\gamma) = 0 \,|\, \theta = \bar{\theta}] = 1$ one must chose $\gamma = \gamma_n$ so that $2\frac{\sigma^2}{n}\gamma_n \to \infty$, with γ_n growing faster than n. This is in line with the well known requirements for Lasso to be model selection consistent. In fact, recalling remark 5, the link between γ and the regularization parameter γ_L for Lasso is given by $\gamma_L = \sqrt{2\gamma}$. The condition $n^{-1}\gamma_n \to \infty$ translates into $n^{-1/2}\gamma_{Ln} \to \infty$, a well known condition for Lasso to be model selection consistent (Zhao and Yu, 2006; Bach, 2008).

The results obtained so far suggest that the Empirical Bayes resulting from PARD has desirable properties with respect to the MSE of the estimators. One wonders whether the same favorable properties are inherited by MKL or, equivalently, by GLasso. The next proposition shows that this is not the case. In fact, for $\bar{\theta}^{(i)} \neq 0$, MKL does not yield consistent estimators for λ_i^{opt}; in addition, for $\theta^{(i)} = 0$, the probability of setting $\hat{\lambda}_i(\gamma)$ to zero (see Equation (24)) is much smaller than that obtained using PARD (see Equation (19)); this is illustrated in Figure 4 (top). Also note that, as illustrated in Figure 4 (bottom), when the true $\bar{\theta}$ is equal to zero, MKL tends to give much larger values of $\hat{\lambda}$ than those given by PARD. This results in larger values of $\|\hat{\theta}\|$ (see Figure 4).

Proposition 12 *Consider model* (1) *under the probabilistic model described in Figure 1(b), and assume $G^\top G = nI$. Then the estimator of λ_i obtained by maximizing the joint posterior $\mathbf{p}(\lambda, \phi|y)$ (see Equations* (10) *and* (11)*),*

$$\{\hat{\lambda}(\gamma), ..., \hat{\lambda}_p(\gamma)\} := \arg \max_{\lambda \in \mathbb{R}_+^p, \phi \in \mathbb{R}_+^m} \mathbf{p}(\lambda, \phi|y),$$

is given by

$$\hat{\lambda}_i(\gamma) = \max\left(0, \frac{\|\hat{\theta}_{LS}^{(i)}\|}{\sqrt{2\gamma}} - \frac{\sigma^2}{n}\right), \tag{22}$$

where

$$\hat{\theta}_{LS}^{(i)} = \frac{1}{n}\left(G^{(i)}\right)^{\top} y$$

is the Least Squares estimator of the $i-th$ block $\theta^{(i)}$ for $i = 1, \ldots, p$. For $n \to \infty$ the estimator $\hat{\lambda}_i(\gamma)$ satisfies

$$\lim_{n\to\infty} \hat{\lambda}_i(\gamma) \stackrel{m.s.}{=} \frac{\|\bar{\theta}^{(i)}\|}{\sqrt{2\gamma}} \ . \tag{23}$$

In addition, the probability $\mathbb{P}[\hat{\lambda}_i(\gamma) = 0 \,|\, \theta = \bar{\theta}]$ of setting $\hat{\lambda}_i(\gamma) = 0$ is given by

$$\mathbb{P}_\theta[\hat{\lambda}_i(\gamma) = 0 \,|\, \theta = \bar{\theta}] = \mathbb{P}\left[\chi^2\left(k_i, \|\bar{\theta}^{(i)}\|^2 \frac{n}{\sigma^2}\right) \leq 2\gamma \frac{\sigma^2}{n}\right] \ . \tag{24}$$

Note that the limit of the MKL estimators $\hat{\lambda}_i(\gamma)$ as $n \to \infty$ depends on γ. Therefore, using MKL (GLasso), one cannot hope to get consistent estimators of λ_i^{opt}. Indeed, for $\|\bar{\theta}^{(i)}\|^2 \neq 0$, consistency of $\hat{\lambda}_i(\gamma)$ requires $\gamma \to \frac{k_i^2}{2\|\bar{\theta}^{(i)}\|^2}$, which is a circular requirement.

5.2 Asymptotic Properties Using General Regressors

In this subsection, we replace the deterministic matrix G with $G_n(\omega)$, where $G_n(\omega)$ represents an $n \times m$ matrix defined on the complete probability space $(\Omega, \mathcal{B}, \mathbb{P})$ with ω a generic element of Ω and $\mathcal{B}$ the sigma field of Borel regular measures. In particular, the rows of G_n are independent[3] realizations from a zero-mean random vector with positive definite covariance Ψ.

As in the previous part of this section, λ and θ are seen as parameters, and the true value of θ is $\bar{\theta}$. Hence, all the randomness present in the next formulas comes only from G_n and the measurement noise.

We make the following (mild) assumptions on G_n. Recalling model (1), assume that $G^{\top}G/n$ is bounded and bounded away from zero in probability, so that there exist constants $\infty > c_{max} \geq c_{min} > 0$ with

$$\lim_{n\to\infty} P[c_{min} I \leq G^{\top}G/n \leq c_{max} I] = 1 \ , \tag{25}$$

so, as n increases, the probability that a particular realization G satisfies

$$c_{min} I \leq G^{\top}G/n \leq c_{max} I \tag{26}$$

increases to 1.

In the following lemma, whose proof is in the Appendix, we introduce a change of variables that is key for our understanding of the asymptotic properties of PARD under these more general regressors.

Lemma 13 *Fix $i \in \{1, \ldots, p\}$ and consider the decomposition*

$$\begin{array}{rcl} y & = & G^{(i)}\theta^{(i)} + \sum_{j=1, j\neq i}^{p} G^{(j)}\theta^{(j)} + v \\ & = & G^{(i)}\theta^{(i)} + \bar{v} \end{array} \tag{27}$$

3. The independence assumption can be removed and replaced by mixing conditions.

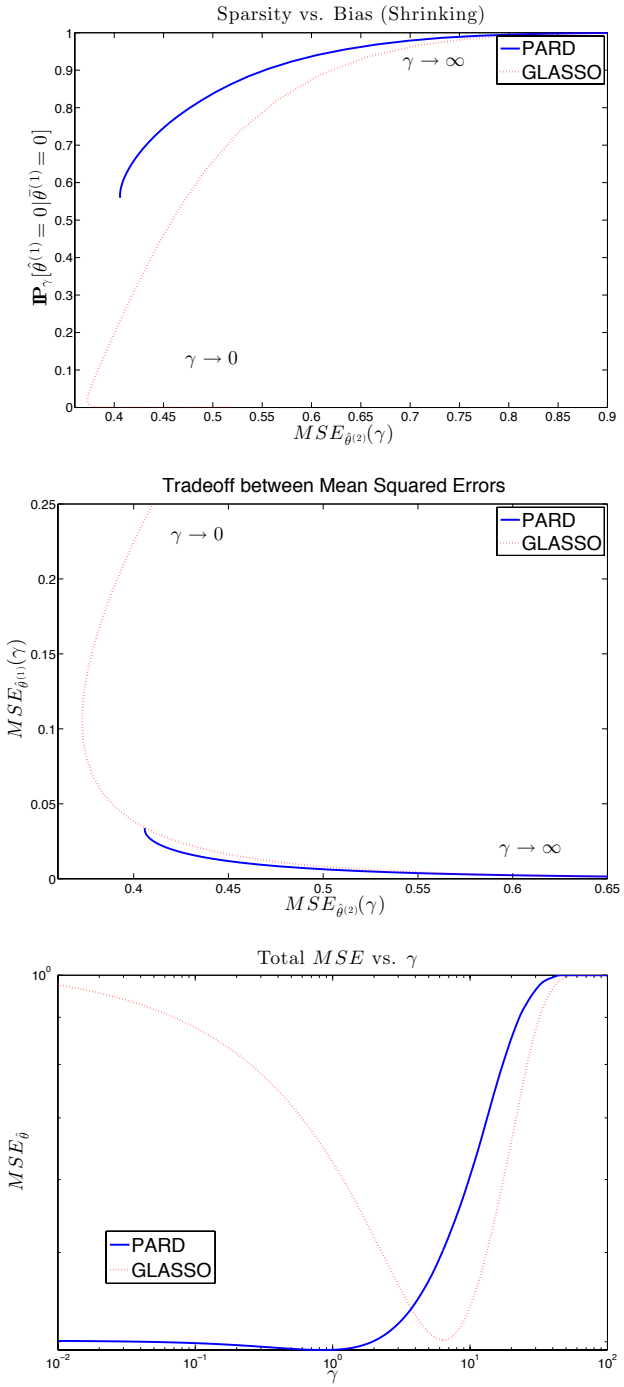


Figure 4: In this example we have two blocks ($p = 2$) of dimension $k_1 = k_2 = 10$ with $\bar{\theta}^{(1)} = 0$ and all the components of the true $\bar{\theta}^{(2)} \in \mathbb{R}^{10}$ set to one. The matrix G is the identity, so that the output dimension ($y \in \mathbb{R}^n$) is $n = 20$; the noise variance equals 0.5. Top: probability of setting $\hat{\theta}^{(1)}$ to zero vs Mean Squared Error in $\hat{\theta}^{(2)}$. Center: Mean Squared Error in $\hat{\theta}^{(1)}$ vs. Mean Squared Error in $\hat{\theta}^{(2)}$; both curves are parametrized in $\gamma \in [0, +\infty)$. Bottom: Total Mean Squared Error (on $\hat{\theta}$) as a function of γ.

of the linear measurement model (1) *and assume* (26) *holds. Define*

$$\Sigma_{\bar{v}}^{(i)} := \sum_{j=1, j\neq i}^{p} G^{(j)} \left(G^{(j)}\right)^{\top} \lambda_j + \sigma^2 I.$$

Consider now the singular value decomposition

$$\frac{\Sigma_{\bar{v}}^{(i)^{-1/2}} G^{(i)}}{\sqrt{n}} = U_n^{(i)} D_n^{(i)} \left(V_n^{(i)}\right)^{\top}, \tag{28}$$

where each $D_n^{(i)} = diag(d_{k,n}^{(i)})$ *is* $k_i \times k_i$ *diagonal matrix. Then* (27) *can be transformed into the equivalent linear model*

$$z_n^{(i)} \quad = \quad D_n^{(i)} \beta_n^{(i)} + \epsilon_n^{(i)}, \tag{29}$$

where

$$\begin{gathered} z_n^{(i)} := \left(U_n^{(i)}\right)^{\top} \frac{\Sigma_{\bar{v}}^{(i)^{-1/2}} y}{\sqrt{n}} = (z_{k,n}^{(i)}), \qquad \beta_n^{(i)} := \left(V_n^{(i)}\right)^{\top} \theta^{(i)} = (\beta_{k,n}^{(i)}), \\ \epsilon_n^{(i)} := \left(U_n^{(i)}\right)^{\top} \frac{\Sigma_{\bar{v}}^{(i)^{-1/2}} \bar{v}}{\sqrt{n}} = (\epsilon_{k,n}^{(i)}), \end{gathered} \tag{30}$$

and $D_n^{(i)}$ *is uniformly (in* n*) bounded and bounded away from zero.*

Below, the dependence of $\Sigma_y(\lambda)$ on G_n, and hence on n, is omitted to simplify the notation. Furthermore, $\longrightarrow_p$ denotes convergence in probability.

Theorem 14 *For known* γ *and conditional on* $\theta = \bar{\theta}$*, define*

$$\hat{\lambda}^n = \arg \min_{\lambda \in \mathcal{C} \bigcap \mathbb{R}_+^p} \frac{1}{2} \log \det(\Sigma_y(\lambda)) + \frac{1}{2} y^{\top} \Sigma_y^{-1}(\lambda) y + \gamma \sum_{i=1}^{p} \lambda_i, \tag{31}$$

where $\mathcal{C}$ *is any* p*-dimensional ball with radius strictly larger than* $\max_i \frac{\|\bar{\theta}^{(i)}\|^2}{k_i}$*. Suppose that the hypotheses of Lemma 13 hold. Consider the estimator (31) along its* $i-th$ *component* λ_i *that, in view of* (29)*, is given by:*

$$\hat{\lambda}_i^n = \arg \min_{\lambda \in \mathbb{R}_+} \frac{1}{2} \sum_{k=1}^{k_i} \left[\frac{\eta_{k,n}^2 + v_{k,n}}{\lambda + w_{k,n}} + \log(\lambda + w_{k,n}) \right] + \gamma \lambda \, , \tag{32}$$

where $\eta_{k,n} := \beta_{k,n}^{(i)}$*,* $w_{k,n} := 1/(n(d_{k,n}^{(i)})^2)$ *and* $v_{k,n} = 2 \frac{\epsilon_{k,n}^{(i)}}{d_{k,n}^{(i)}} \beta_{k,n}^{(i)} + \left(\frac{\epsilon_{k,n}^{(i)}}{d_{k,n}^{(i)}} \right)^2$*. Let*

$$\bar{\lambda}_i^{\gamma} := \frac{-k_i + \sqrt{k_i^2 + 8\gamma \|\bar{\theta}^{(i)}\|^2}}{4\gamma} \, , \quad \bar{\lambda}_i = \frac{\|\bar{\theta}^{(i)}\|^2}{k_i} \, .$$

We have the following results:

1. $\bar{\lambda}_i^{\gamma} \leq \bar{\lambda}_i$ *for all* $\gamma > 0$*, and* $\lim_{\gamma \to 0^+} \bar{\lambda}_i^{\gamma} = \bar{\lambda}_i$ *.*

2. *If* $\|\bar{\theta}^{(i)}\| > 0$ *and* $\gamma > 0$, *we have* $\hat{\lambda}_i^n \longrightarrow_p \bar{\lambda}_i^\gamma$.

3. *If* $\|\bar{\theta}^{(i)}\| > 0$ *and* $\gamma = 0$, *we have* $\hat{\lambda}_i^n \longrightarrow_p \bar{\lambda}_i$.

4. *if* $\bar{\theta}^{(i)} = 0$, *we have* $\hat{\lambda}_i^\gamma \longrightarrow_p 0$ *for any value* $\gamma \geq 0$.

We now show that, when $\gamma = 0$, the above result relates to the problem of minimizing the MSE of the i-th block with respect to λ_i, with all the other components of λ coming from $\hat{\lambda}^n$. For any index i, we define

$$I_1^{(i)} := \left\{ j \,:\, j \neq i \text{ and } \bar{\theta}^{(j)} \neq 0 \right\}, \quad I_0^{(i)} := \left\{ j \,:\, j \neq i \text{ and } \bar{\theta}^{(j)} = 0 \right\}. \tag{33}$$

If $\hat{\theta}_n^{(i)}(\lambda)$ denotes the i-th component of the PARD estimate of θ defined in (5), our aim is to optimize the objective

$$MSE_n(\lambda_i) := \operatorname{tr}\left[\mathbb{E}_v\left[(\hat{\theta}_n^{(i)}(\lambda) - \bar{\theta}^{(i)})(\hat{\theta}_n^{(i)}(\lambda) - \bar{\theta}^{(i)})^\top\right]\right] \quad \text{with} \quad \lambda_j = \bar{\lambda}_j^n \quad \text{for} \quad j \neq i$$

where is $\bar{\lambda}_j^n$ is any sequence satisfying condition

$$\lim_{n\to\infty} f_n = +\infty \quad \text{where} \quad f_n := \min_{j \in I_1^{(i)}} n\lambda_j^n, \tag{34}$$

(condition (34) appears again in the Appendix as (47)). Note that, in particular, $\bar{\lambda}_j^n = \hat{\lambda}_j^n$ in (31) satisfy (34) in probability.

Lemma 13 shows that we can consider the transformed linear model associated with the i-th block, that is,

$$z_{k,n}^{(i)} \;=\; d_{k,n}^{(i)}\beta_{k,n}^{(i)} + \epsilon_{k,n}^{(i)}, \quad k = 1, \ldots, k_i, \tag{35}$$

where all the three variables on the RHS depend on $\bar{\lambda}_j^n$ for $j \neq i$. In particular, the vector $\beta_n^{(i)}$ consists of an orthonormal transformation of $\theta^{(i)}$ while the $d_{k,n}^{(i)}$ are all bounded below in probability. In addition, by letting

$$\mathbb{E}_v\left[\epsilon_{k,n}^{(i)}\right] = m_{k,n}, \quad \mathbb{E}_v\left[(\epsilon_{k,n}^{(i)} - m_{k,n})^2\right] = \sigma_{k,n}^2,$$

we also know from Lemma 20 (see Equations (48) and (49)) that, provided $\bar{\lambda}_j^n$ ($j \neq i$) satisfy condition (34), both $m_{k,n}$ and $\sigma_{k,n}^2$ tend to zero (in probability) as n goes to ∞. Then, after simple computations, one finds that the MSE relative to $\beta_n^{(i)}$ is the following random variable whose statistics depend on n:

$$MSE_n(\lambda_i) = \sum_{k=1}^{k_i} \frac{\beta_{k,n}^2 + n\lambda_i^2 d_{k,n}^2(m_{k,n}^2 + \sigma_{k,n}^2) - 2\lambda_i d_{k,n} m_{k,n}\beta_{k,n}}{(1 + n\lambda_i d_{k,n}^2)^2} \quad \text{with } \lambda_j = \bar{\lambda}_j^n \text{ for } j \neq i.$$

Above, except for λ_i, the dependence on the block number i was omitted to improve readability.

Now, let $\breve{\lambda}_i^n$ denote the minimizer of the following weighted version of the $MSE_n(\lambda_i)$:

$$\breve{\lambda}_i^n = \arg\min_{\lambda \in \mathbb{R}_+} \sum_{k=1}^{k_i} d_{k,n}^4 \frac{\beta_{k,n}^2 + n\lambda_i^2 d_{k,n}^2(m_{k,n}^2 + \sigma_{k,n}^2) - 2\lambda_i d_{k,n} m_{k,n}\beta_{k,n}}{(1 + n\lambda_i d_{k,n}^2)^2}.$$

Then, the following result holds.

Proposition 15 *For $\gamma = 0$ and conditional on $\theta = \bar{\theta}$, the following convergences in probability hold*

$$\lim_{n\to\infty} \breve{\lambda}_i^n = \frac{\|\bar{\theta}^{(i)}\|^2}{k_i} = \lim_{n\to\infty} \hat{\lambda}_i^n \ , \quad i = 1, 2, \ldots, p. \tag{36}$$

The proof follows arguments similar to those used in last part of the proof of Theorem 14, see also proof of Theorem 6 in Aravkin et al. (2012), and is therefore omitted.

We can summarize the two main findings reported in this subsection as follows. As the number of measurements go to infinity:

1. regardless of the value of γ (provided γ does not depend on n; in such a case suitable conditions on the rate are necessary, see also Remark 11), the proposed estimator will correctly set to zero only those λ_i associated with null blocks;

2. when $\gamma = 0$, results 2 and 3 of Theorem 14 provide the asymptotic properties of ARD, showing that the estimate of λ_i will converge to the energy of the i-th block (divided by its dimension).

   This same value also represents the asymptotic minimizer of a weighted version of the MSE relative to the i-th block. In particular, the weights change with n, as they are defined by the singular values $d_{k,n}^{(i)}$ (raised at fourth power) that depend on the trajectories of the other components of λ (see (28)). This roughly corresponds to giving more emphasis to components of θ which excite directions in the output space where the signal to noise ratio is high; this indicates some connection with reduced rank regression where one only seeks to approximate the most important (relative to noise level) directions in output space.

Remark 16 (Consistency of $\hat{\theta}_{PA}$) *It is a simple check to show that, under the assumptions of Theorem 14, the empirical Bayes estimator $\hat{\theta}_{PA}(\hat{\lambda}_n)$ in (5) is a consistent estimator of $\bar{\theta}$. Indeed, Theorem 14 shows much more than this, implying that for $\gamma = 0$, $\hat{\theta}_{PA}(\hat{\lambda}_n)$ possesses some desirable asymptotic properties in terms on Mean Squared Error, see also Remark 17.*

5.3 Marginal Likelihood and Weighted MSE: Perturbation Analysis

We now provide some additional insights on point 2 above, investigating why the weights $d_{k,n}^4$ may lead to an effective strategy for hyperparameter estimation.
For our purposes, just to simplify the notation, let us consider the case of a single m-dimensional block. In this way, λ becomes a scalar and the noise $\epsilon_{k,n}$ in (35) is zero-mean of variance $1/n$.

Under the stated assumptions, the MSE weighted by $d_{k,n}^{\alpha}$, with α an integer, becomes

$$\sum_{k=1}^{m} d_{k,n}^{\alpha} \frac{n^{-1}\beta_{k,n}^2 + \lambda^2 d_{k,n}^2}{(n^{-1} + \lambda d_{k,n}^2)^2},$$

whose partial derivative with respect to λ, apart from the scale factor $2/n$, is

$$F_{\alpha}(\lambda) = \sum_{k=1}^{m} d_{k,n}^{\alpha+2} \frac{\lambda - \beta_{k,n}^2}{(n^{-1} + \lambda d_{k,n}^2)^3}.$$

Let $\beta_k = \lim_{n\to\infty} \beta_{k,n}$ and $d_k = \lim_{n\to\infty} d_{k,n}$.[4] When n tends to infinity, arguments similar to those introduced in the last part of the proof of Theorem 14 show that, in probability, the zero of F_α becomes

$$\breve{\lambda}(\alpha) = \frac{\sum_{k=1}^m d_k^{\alpha-4}\beta_k^2}{\sum_{k=1}^m d_k^{\alpha-4}}.$$

Notice that the formula above is a generalization of the first equality in (36) that was obtained by setting $\alpha = 4$. However, for practical purposes, the above expressions are not useful since the true values of $\beta_{k,n}$ and β_k depend on the unknown $\bar{\theta}$. One can then consider a noisy version of F_α obtained by replacing $\beta_{k,n}$ with its least squares estimate, that is,

$$\tilde{F}_\alpha(\lambda) = \sum_{k=1}^m d_{k,n}^{\alpha+2} \frac{\lambda - \left(\beta_{k,n} + \frac{v_{k,n}}{\sqrt{n}d_{k,n}}\right)^2}{(n^{-1} + \lambda d_{k,n}^2)^3},$$

where the random variable $v_{k,n}$ is of unit variance. For large n, considering small additive perturbations around the model $z_k = d_k\beta_k$, it is easy to show that the minimizer tends to the following perturbed version of $\breve{\lambda}$:

$$\breve{\lambda}(\alpha) + 2\frac{\sum_{k=1}^m d_k^{\alpha-5}\beta_k v_{k,n}}{\sqrt{n}\sum_{k=1}^m d_k^{\alpha-4}}. \tag{37}$$

We must now choose the value of α that should enter the above formula. This is far from trivial since the optimal value (minimizing MSE) depends on the unknown β_k. On one hand, it would seem advantageous to have α close to zero. In fact, $\alpha = 0$ relates $\breve{\lambda}$ to the minimization of the MSE on θ while $\alpha = 2$ minimizes the MSE on the output prediction, see the discussion in Section 4 of Aravkin et al. (2012). On the other hand, a larger value for α could help in controlling the additive perturbation term in (37) possibly reducing its sensitivity to small values of d_k. For instance, the choice $\alpha = 0$ introduces in the numerator of (37) the term β_k/d_k^5. This can destabilize the convergence towards $\breve{\lambda}$, leading to poor estimates of the regularization parameters, as, for example, described via simulation studies in Section 5 of Aravkin et al. (2012). In this regard, the choice $\alpha = 4$ appears interesting: it sets $\breve{\lambda}$ to the energy of the block divided by m, removing the dependence of the denominator in (37) on d_k. In particular, it reduces (37) to

$$\frac{\|\beta\|^2}{m} + \frac{2}{m}\sum_{k=1}^m \frac{\beta_k v_{k,n}}{\sqrt{n}d_k} = \sum_{k=1}^m \frac{\beta_k^2}{m}\left(1 + 2\frac{v_{k,n}}{\beta_k\sqrt{n}d_k}\right). \tag{38}$$

It is thus apparent that $\alpha = 4$ makes the perturbation on $\frac{\beta_k^2}{m}$ dependent on $\frac{v_{k,n}}{\beta_k\sqrt{n}d_k}$, that is, on the relative reconstruction error on β_k. This appears a reasonable choice to account for the ill-conditioning possibly affecting least-squares.

Interestingly, for large n, this same philosophy is followed by the marginal likelihood procedure for hyperparameter estimation up to first-order approximations. In fact, under

4. We are assuming that both of the limits exist. This holds under conditions ensuring that the SVD decomposition leading to (35) is unique, for example, see the discussion in Section 4 of Bauer (2005), and combining the convergence of sample covariances with a perturbation result for the Singular Value Decomposition of symmetric matrices (such as Theorem 1 in Bauer, 2005, see also Chatelin, 1983).

the stated assumptions, apart from constants, the expression for twice the negative log of the marginal likelihood is

$$\sum_{k=1}^{m} \log(n^{-1} + \lambda d_{k,n}^2) + \frac{z_{k,n}^2}{n^{-1} + \lambda d_{k,n}^2},$$

whose partial derivative w.r.t. λ is

$$\sum_{k=1}^{m} \frac{\lambda d_{k,n}^4 + n^{-1} d_{k,n}^2 - z_{k,n}^2 d_{k,n}^2}{(n^{-1} + \lambda d_{k,n}^2)^2}.$$

As before, we consider small perturbations around $z_k = d_k \beta_k$ to find that a critical point occurs at

$$\sum_{k=1}^{m} \frac{\beta_k^2}{m} \left(1 + 2\frac{v_{k,n}}{\beta_k \sqrt{n} d_k}\right),$$

which is exactly the same minimizer reported in (38).

5.4 Concluding Remarks and Connections to Subsection 4.2

We can now give an interpretation of the results depicted in Figure 3 in view of the theoretical analyses developed in this section.

When the regressors are orthogonal, which corresponds, asymptotically, to the case of white noise defining the entries of G, the results in subsection 5.1 (e.g., Corollary 10) show that ARD has a clear advantage over GLasso (MKL) in terms of MSE. This explains the outcomes from the first numerical experiment of Section 4.2 which are depicted in the left panel of Figure 3.

For the case of general regressors, subsection 5.2 provides insights regarding the properties of ARD, including its consistency. Ideally, a regularized estimator should adopt those hyperparameters contained in λ that minimize the MSE objective, but this objective depends on $\bar{\theta}$, which is what we aim to estimate. One could then consider a noisy version of the MSE function, for example, obtained replacing $\bar{\theta}$ with its least squares estimate. The problem is that this new objective can be very sensitive to the noise, leading to poor regularizers as, for example, described by simulation studies in Section 5 of Aravkin et al. (2012). On an asymptotic basis, ARD circumvents this problem using particular weights which introduce a bias in the MSE objective but make it more stable, that is, less sensitive to noise. This results in a form of regularization introduced through hyperparameter estimation. We believe that this peculiarity is key to understanding not only the results in Figure 3 but also the success of ARD in several applications documented in the literature.

Remark 17 [PARD: penalized version of ARD] *Note that, when one considers sparsity inducing performance, the use of a penalized version of ARD, for example, given by PARD, clearly may help in setting more blocks to zero, see Figure 4 (top). In comparison with GLasso, the important point here is that the non convex nature of PARD permits sparsity promotion without adopting too large a value of γ. This makes PARD a slightly perturbed version of ARD. Hence, PARD is able to induce more sparsity than ARD while*

maintaining similar performance in the reconstruction of the non null blocks. This is illustrated by the Monte Carlo results in Section 4.2. To better understand the role of γ, consider the orthogonal case discussed in Section 5.1, for sake of clarity. Recall the observation in Remark 11 that model selection consistency requires $\gamma = \gamma_n$. It is easy to show that the oracle property (Zou, 2006) holds provided $\frac{\gamma_n}{n} \to \infty$ and $\frac{\gamma_n}{n^2} \to 0$. However, large γ's tend to introduce excessive shrinkage, for example, see Figure 4 (center). It is well known (Leeb and Pötscher, 2005) that shrinkage estimators that possess the oracle property have unbounded normalized risk (normalized Mean Squared Error for quadratic loss), meaning that they are certainly not optimal in terms of Mean Squared Error. To summarize, the asymptotic properties suggest that to obtain the oracle properties γ_n should go to infinity at a suitable rate with n while γ should be set equal to zero to optimize the mean squared error. However, for finite data length n, the optimal mean squared error properties as a function of γ are found for a finite but nonzero γ. This fact, also illustrated in Figure 4 (bottom), is not in contrast w.r.t. Corollary 10: γ may induce a useful bias in the marginal likelihood estimator of the λ_i which can reduce the variance. This also explains the experimental results showing that PARD performs slightly better than ARD.

6. Conclusions

We have presented a comparative study of some methods for sparse estimation: GLasso (equivalently, MKL), ARD and its penalized version PARD, which is cast in the framework of the Type II Bayesian estimators. They derive from the same Bayesian model, yet in a different way. The peculiarities of PARD can be summarized as follows:

- in comparison with GLasso, PARD derives from a marginalized joint density with the resulting estimator involving optimization of a non-convex objective;
- the non-convex nature allows PARD to achieve higher levels of sparsity than GLasso without introducing too much regularization in the estimation process, thus providing a better tradeoff between sparsity and shrinking.
- the MSE analysis reported in this paper reveals the superior performance of PARD in the reconstruction of the parameter groups different from zero. Remarkably, our analysis elucidates this issue showing the robustness of the empirical Bayes procedure, based on marginal likelihood optimization, independently of the correctness of the priors which define the stochastic model underlying PARD. As a consequence of our analysis, the asymptotic properties of ARD have also been illuminated.

Many variations of PARD are possible, adopting different prior models for λ. In this paper, the exponential prior is used to compare different estimators that can be derived from the same Bayesian model underlying GLasso. In this way, it is shown that the same stochastic framework can give rise to an estimator derived from a posterior marginalization that has significant advantages over another estimator derived from posterior optimization.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement no 257462 HYCON2

Network of excellence, by the MIUR FIRB project RBFR12M3AC - Learning meets time: a new computational approach to learning in dynamic systems.

Appendix A. Proofs

In this Appendix, we present proofs of the main results in the paper.

A.1 Proof of Proposition 9

Under the simplifying assumption $G^\top G = nI$, one can use

$$\Sigma_y(\lambda)^{-1} = \sigma^{-2}\left[I - G(\sigma^2\Lambda^{-1} + G^TG)^{-1}G^\top\right]$$

which derives from the matrix inversion lemma to obtain

$$G^{(i)\top}\Sigma_y(\lambda)^{-1} = \frac{1}{n\lambda_i + \sigma^2}G^{(i)\top},$$

and so

$$\mathrm{tr}\left(G^{(i)\top}\Sigma_y^{-1}G^{(i)}\right) = \frac{nk_i}{n\lambda_i+\sigma^2} \quad \text{and} \quad \|G^{(i)\top}\Sigma_y^{-1}y\|_2^2 = \left(\frac{n}{n\lambda_i+\sigma^2}\right)^2\|\hat\theta_{LS}^{(i)}\|^2 .$$

Inserting these expressions into (8) with $\mu_i = 0$ yields a quadratic equation in λ_i which always has two real solutions. One is always negative while the other, given by

$$\frac{1}{4\gamma}\left[\sqrt{k_i^2 + 8\gamma\|\hat\theta_{LS}^{(i)}\|^2} - \left(k_i + \frac{4\sigma^2\gamma}{n}\right)\right]$$

is non-negative provided

$$\frac{\|\hat\theta_{LS}^{(i)}\|^2}{k_i} \geq \frac{\sigma^2}{n}\left[1 + \frac{2\gamma\sigma^2}{nk_i}\right] . \tag{39}$$

This concludes the proof of (18). The limiting behavior for $\gamma \to 0$ can be easily verified, yielding

$$\hat\lambda_i(0) = \max\left(0, \frac{\|\hat\theta_{LS}^{(i)}\|^2}{k_i} - \frac{\sigma^2}{n}\right) \quad i = 1, .., p.$$

Also note that $\hat\theta_{LS}^{(i)} = \frac{1}{n}\left(G^{(i)}\right)^\top y$ and $\left(G^{(i)}\right)^\top G^{(i)} = nI_{k_i}$ while $\left(G^{(i)}\right)^\top G^{(j)} = 0$, $\forall j \neq i$. This implies that $\hat\theta_{LS}^{(i)} \sim \mathcal{N}(\bar\theta^{(i)}, \frac{\sigma^2}{n}I_{k_i})$. Therefore

$$\|\hat\theta_{LS}^{(i)}\|^2\frac{n}{\sigma^2} \sim \chi^2(d,\mu) \quad d = k_i, \quad \mu = \|\bar\theta^{(i)}\|^2\frac{n}{\sigma^2}.$$

This, together with (39), proves also (19).

A.2 Proof of Proposition 10

In the proof of Proposition 9 it was shown that $\|\hat{\theta}_{LS}^{(i)}\|^2 \frac{n}{\sigma^2}$ follows a noncentral χ^2 distribution with k_i degrees of freedom and noncentrality parameter $\|\bar{\theta}^{(i)}\|^2 \frac{n}{\sigma^2}$. Hence, it is a simple calculation to show that

$$\mathbb{E}[\hat{\lambda}_i^* \,|\, \theta = \bar{\theta}] = \frac{\|\bar{\theta}^{(i)}\|^2}{k_i} \quad \mathrm{Var}[\hat{\lambda}_i^* \,|\, \theta = \bar{\theta}] = \frac{2\sigma^4}{k_i n^2} + \frac{4\|\bar{\theta}^{(i)}\|^2 \sigma^2}{k_i^2 n} \ .$$

By Corollary 8, the first of these equations shows that $\mathbb{E}[\hat{\lambda}_i^* \,|\, \theta = \bar{\theta}] = \lambda_i^{opt}$. In addition, since $\mathrm{Var}\{\hat{\lambda}_i^*\}$ goes to zero as $n \to \infty$, $\hat{\lambda}_i^*$ converges in mean square (and hence in probability) to λ_i^{opt}.

As for the analysis of $\hat{\lambda}_i(0)$, observe that

$$\mathbb{E}[\hat{\lambda}_i(0) \,|\, \theta = \bar{\theta}] = \mathbb{E}[\hat{\lambda}_i^* \,|\, \theta = \bar{\theta}] - \int_0^{k_i \frac{\sigma^2}{n}} \left(\frac{\|\hat{\theta}_{LS}^{(i)}\|^2}{k_i} - \frac{\sigma^2}{n} \right) dP(\|\hat{\theta}_{LS}^{(i)}\|^2 \,|\, \theta = \bar{\theta})$$

where $dP(\|\hat{\theta}_{LS}^{(i)}\|^2 \,|\, \theta = \bar{\theta})$ is the measure induced by $\|\hat{\theta}_{LS}^{(i)}\|^2$. The second term in this expression can be bounded by

$$-\int_0^{k_i \frac{\sigma^2}{n}} \left(\frac{\|\hat{\theta}_{LS}^{(i)}\|^2}{k_i} - \frac{\sigma^2}{n} \right) dP(\|\hat{\theta}_{LS}^{(i)}\|^2 \,|\, \theta = \bar{\theta}) \leq \frac{\sigma^2}{n} \int_0^{k_i \frac{\sigma^2}{n}} dP(\|\hat{\theta}_{LS}^{(i)}\|^2 \,|\, \theta = \bar{\theta}),$$

where the last term on the right hand side goes to zero as $n \to \infty$. This proves that $\hat{\lambda}_i(0)$ is asymptotically unbiased. As for consistency, it is sufficient to observe that $\mathrm{Var}[\hat{\lambda}_i(0) \,|\, \theta = \bar{\theta}] \leq \mathrm{Var}[\hat{\lambda}_i^* \,|\, \theta = \bar{\theta}]$ since "saturation" reduces variance. Consequently, $\hat{\lambda}_i(0)$ converges in mean square to its mean, which asymptotically is λ_i^{opt} as shown above. This concludes the proof.

A.3 Proof of Proposition 12

Following the same arguments as in the proof of Proposition 9, under the assumption $G^\top G = nI$ we have that

$$\|G^{(i)\top} \Sigma_y^{-1} y\|_2^2 = \left(\frac{n}{n\lambda_i + \sigma^2} \right)^2 \|\hat{\theta}_{LS}^{(i)}\|^2.$$

Inserting this expression into (14) with $\mu_i = 0$, one obtains a quadratic equation in λ_i which has always two real solutions. One is always negative while the other, given by

$$\frac{\|\hat{\theta}_{LS}^{(i)}\|}{\sqrt{2\gamma}} - \frac{\sigma^2}{n}.$$

is non-negative provided

$$\|\hat{\theta}_{LS}^{(i)}\|^2 \geq \frac{2\gamma\sigma^4}{n^2} \ . \tag{40}$$

This concludes the proof of (22).

The limiting behavior for $n \to \infty$ in Equation (23) is easily verified with arguments similar to those in the proof of Proposition 10. As in the proof of Proposition 9, $\|\hat{\theta}_{LS}^{(i)}\|^2 \frac{n}{\sigma^2}$ follows a noncentral $\chi^2(d, \mu)$ distribution with $d = k_i$ and $\mu = \|\bar{\theta}^{(i)}\|^2 \frac{n}{\sigma^2}$, so that from (40) the probability of setting $\hat{\lambda}_i(\gamma)$ to zero is as given in (24).

A.4 Proof of Lemma 13:

Let us consider the Singular Value Decomposition (SVD)

$$\frac{\sum_{j=1, j\neq i}^{p} G^{(j)} \left(G^{(j)}\right)^\top \lambda_j}{n} = PSP^\top, \tag{41}$$

where, by the assumption (26), using $\frac{\sum_{j=1,j\neq i}^{p} G^{(j)}\left(G^{(j)}\right)^\top \lambda_j}{n} \geq \frac{\sum_{j=1,j\neq i,\lambda_j\neq 0}^{p} G^{(j)}\left(G^{(j)}\right)^\top}{n} \min\{\lambda_j, j : \lambda_j \neq 0\}$ and Lemma 19 the minimum singular value $\sigma_{min}(S)$ of S in (41) satisfies

$$\sigma_{min}(S) \geq c_{min} \min\{\lambda_j, j : \lambda_j \neq 0\}. \tag{42}$$

Then the SVD of $\Sigma_{\bar{v}}^{(i)} = \sum_{j=1, j\neq i}^{p} G^{(j)} \left(G^{(j)}\right)^\top \lambda_j + \sigma^2 I$ satisfies

$$\Sigma_{\bar{v}}^{(i)\,-1} = \begin{bmatrix} P & P_\perp \end{bmatrix} \begin{bmatrix} (nS + \sigma^2)^{-1} & 0 \\ 0 & \sigma^{-2} I \end{bmatrix} \begin{bmatrix} P^\top \\ P_\perp^\top \end{bmatrix}$$

so that $\left\| \Sigma_{\bar{v}}^{(i)\,-1} \right\| = \sigma^{-2}$.

Note now that

$$D_n^{(i)} = \left(U_n^{(i)}\right)^\top \frac{\Sigma_{\bar{v}}^{(i)\,-1/2} G^{(i)}}{\sqrt{n}} V_n^{(i)}$$

and therefore, using Lemma 19,

$$\|D_n^{(i)}\| \leq \left\| \Sigma_{\bar{v}}^{(i)\,-1/2} \right\| \sqrt{c_{max}} = \sigma^{-1} \sqrt{c_{max}}$$

proving that $D_n^{(i)}$ is bounded. In addition, again using Lemma 19, condition (26) implies that $\forall a, b$ (of suitable dimensions) s.t. $\|a\| = \|b\| = 1$, $a^\top \frac{P_\perp^\top G^{(i)}}{\sqrt{n}} b \geq k$, $k = \sqrt{1 - \cos^2(\theta_{min})} \geq \frac{c_{min}}{c_{max}} > 0$. This, using (28), guarantees that

$$\begin{aligned} D_n^{(i)} &= \left(U_n^{(i)}\right)^\top \frac{\Sigma_{\bar{v}}^{(i)\,-1/2} G^{(i)}}{\sqrt{n}} V_n^{(i)} = \left(U_n^{(i)}\right)^\top \left(P(nS + \sigma^2)^{-1/2} P^\top + P_\perp \sigma^{-1} P_\perp^\top\right) \frac{G^{(i)}}{\sqrt{n}} \\ &\geq \left(U_n^{(i)}\right)^\top \left(P_\perp \sigma^{-1} P_\perp^\top\right) \frac{G^{(i)}}{\sqrt{n}} \\ &\geq k\sigma^{-1} I \end{aligned}$$

and therefore $D_n^{(i)}$ is bounded away from zero. It is then a matter of simple calculations to show that with the definitions (30) then (27) can be rewritten in the equivalent form (29). □

A.5 Preliminary Lemmas

This part of the Appendix contains some preliminary lemmas which will be used in the proof of Theorem 14. This first focuses on the estimator (31). We show that when the hypotheses of Lemma 13 hold, the estimate (31) satisfies the key assumptions of the forthcoming Lemma 20. We begin with a detailed study of the objective (4).

Let

$$I_1 := \left\{ j \,:\, \bar{\theta}^{(j)} \neq 0 \right\}, \quad I_0 := \left\{ j \,:\, \bar{\theta}^{(j)} = 0 \right\}.$$

Note that these are analogous to $I_1^{(i)}$ and $I_0^{(i)}$ defined in (33), but do not depend on any specific index i. We now state the following lemma.

Lemma 18 *Writing the objective in* (4) *in expanded form gives*

$$g_n(\lambda) = \log \sigma^2 + \underbrace{\frac{1}{2n} \log\det(\sigma^{-2}\Sigma_y(\lambda))}_{S_1} + \underbrace{\frac{1}{2n} \sum_{j \in I_1} \frac{\|\hat{\theta}^{(j)}(\lambda)\|^2}{k_j \lambda_j}}_{S_2} + \underbrace{\frac{1}{2n} \sum_{j \in I_0} \frac{\|\hat{\theta}^{(j)}(\lambda)\|^2}{k_j \lambda_j}}_{S_3}$$
$$+ \underbrace{\frac{1}{n}\gamma\|\lambda\|_1}_{S_4} + \underbrace{\frac{1}{2n\sigma^2}\|y - \sum_j G^j \hat{\theta}^{(j)}(\lambda)\|^2}_{S_5},$$

where $\hat{\theta}(\lambda) = \Lambda G^T \Sigma_y^{-1} y$ *(see* (5)*),* k_j *is the size of the* j*th block, and dependence on* n *has been suppressed. For any minimizing sequence* λ^n*, we have the following results:*

1. $\hat{\theta}_n \to_p \bar{\theta}$.
2. $S_1, S_2, S_3, S_4 \to_p 0$.
3. $S_5 \to_p \frac{1}{2}$.
4. $n\lambda_j^n \to_p \infty$ *for all* $j \in I_1$.

Proof First, note that $0 \leq S_i$ for $i \in \{1, 2, 3, 4\}$. Next,

$$\begin{aligned} S_5 &= \frac{1}{2n\sigma^2}\|y - \sum_j G^j \bar{\theta}^{(j)}(\lambda) + \sum_j G^j \left(\bar{\theta}^{(j)}(\lambda) - \hat{\theta}^{(j)}(\lambda) \right) \|^2 \\ &= \frac{1}{2n\sigma^2}\|\nu + \sum_j G^j \left(\bar{\theta}^{(j)}(\lambda) - \hat{\theta}^{(j)}(\lambda) \right) \|^2 \\ &= \frac{1}{2n\sigma^2}\|\nu\|^2 + \frac{1}{2n\sigma^2}\nu^T \sum_j G^j \left(\bar{\theta}^{(j)}(\lambda) - \hat{\theta}^{(j)}(\lambda) \right) + \frac{1}{2n\sigma^2}\| \sum_j G^j \left(\bar{\theta}^{(j)}(\lambda) - \hat{\theta}^{(j)}(\lambda) \right) \|^2. \end{aligned} \tag{43}$$

The first term converges in probability to $\frac{1}{2}$. Since ν is independent of all G^j, the middle term converges in probability to 0. The third term is the bias incurred unless $\hat{\theta} = \bar{\theta}$. These facts imply that, $\forall \epsilon > 0$,

$$\lim_{n \to \infty} P\left[S_5(\lambda(n)) > \frac{1}{2} - \epsilon \right] = 1\,. \tag{44}$$

Next, consider the particular sequence $\bar{\lambda}_j^n = \frac{\|\bar{\theta}_j\|^2}{k_j}$. For this sequence, it is immediately clear that $S_i \to_p 0$ for $i \in \{2,3,4\}$. To show $S_1 \to_p 0$, note that $\sum \lambda_i G_i G_i^T \leq \max\{\lambda_i\} \sum G_i G_i^T$, and that the nonzero eigenvalues of GG^T are the same as those of $G^T G$. Therefore, we have

$$S_1 \leq \frac{1}{2n} \sum_{i=1}^{m} \log(1 + n\sigma^{-2} \max\{\lambda\} c_{max}) = O_P\left(\frac{\log(n)}{n}\right) \to_p 0 \, .$$

Finally $S_5 \to_p \frac{1}{2}$ by (43), so in fact, $\forall \epsilon > 0$,

$$\lim_{n\to\infty} P\left[\left| g_n(\bar{\lambda}(n)) - \frac{1}{2} - \log(\sigma^2) \right| < \epsilon \right] = 1 \, . \tag{45}$$

Since (45) holds for the deterministic sequence $\bar{\lambda}_n$, any minimizing sequence $\hat{\lambda}_n$ must satisfy, $\forall \epsilon > 0$,

$$\lim_{n\to\infty} P\left[g_n(\hat{\lambda}(n)) < \frac{1}{2} + \log(\sigma^2) + \epsilon \right] = 1$$

which, together with (44), implies (45)

Claims $1, 2, 3$ follow immediately. To prove claim 4, suppose that for a particular minimizing sequence $\check{\lambda}(n)$, we have $n\check{\lambda}_j^n \not\to_p \infty$ for $j \in I_1$. We can therefore find a subsequence where $n\check{\lambda}_j^n \leq K$, and since $S_2(\check{\lambda}(n)) \to_p 0$, we must have $\|\hat{\theta}^{(j)}(\check{\lambda})\| \to_p 0$. But then there is a nonzero bias term in (43), since in particular $\bar{\theta}^{(j)}(\lambda) - \hat{\theta}^{(j)}(\lambda) = \bar{\theta}^{(j)}(\lambda) \neq 0$, which contradicts the fact that $\check{\lambda}(n)$ was a minimizing sequence. ■

We now state and prove a technical Lemma which will be needed in the proof of Lemma 20.

Lemma 19 *Assume* (26) *holds; then the following conditions hold*

(i) Consider $I = [I(1), \ldots, I(p_I)]$ of size p_I to be any subset of the indices $[1, \ldots, p]$, so $p \geq p_I$ and define

$$G^{(I)} = \left[G^{(I(1))} \ldots G^{(I(p_I))} \right] \, ,$$

obtained by taking the subset of blocks of columns of G indexed by I. Then

$$c_{min} I \leq \frac{(G^{(I)})^T G^{(I)}}{n} \leq c_{max} I \, . \tag{46}$$

(ii) Let I^c be the complementary set of I in $[1, \ldots, p]$, so that $I^c \cap I = \emptyset$ and $I \cup I^c = [1, \ldots, p]$. Then the minimal angle θ_{min} between the spaces

$$\mathcal{G}^I := col\, span\{G^{(i)}/\sqrt{n}, \;\; i \in I\} \quad \text{and} \quad \mathcal{G}^{I^c} := col\, span\{G^{(j)}/\sqrt{n} : j \in I^c\}$$

satisfies:

$$\theta_{min} \geq \operatorname{acos}\left(\sqrt{1 - \frac{c_{min}}{c_{max}}} \right) > 0.$$

Proof Result (46) is a direct consequence of Horn and Johnson (1994), see Corollary 3.1.3. As far as condition (ii) is concerned we can proceed as follows: let U_I and U_{I^c} be orthonormal matrices whose columns span $\mathcal{G}^I$ and $\mathcal{G}^{I^c}$, so that there exist matrices T_I and T_{I^c} so that

$$G^{(I)}/\sqrt{n} = U_I T_I,$$
$$G^{(I^c)}/\sqrt{n} = U_{I^c} T_{I^c}$$

where $G^{(I^c)}$ is defined analogously to $G^{(I)}$. The minimal angle between $\mathcal{G}^I$ and $\mathcal{G}^{I^c}$ satisfies

$$\cos(\theta_{min}) = \left\| U_I^\top U_{I^c} \right\|.$$

Now observe that, up to a permutation of the columns which is irrelevant, $G/\sqrt{n} = [U_I T_I \quad U_{I^c} T_{I^c}]$, so that

$$U_I^\top G/\sqrt{n} = [T_I \quad U_I^\top U_{I^c}\ T_{I^c}] = [I \quad U_I^\top U_{I^c}] \begin{bmatrix} T_I & 0 \\ 0 & T_{I^c} \end{bmatrix}.$$

Denoting with $\sigma_{min}(A)$ and $\sigma_{max}(A)$ the minimum and maximum singular values of a matrix A, it is a straightforward calculation to verify that the following chain of inequalities holds:

$$\begin{aligned} c_{min} = \sigma_{min}(G^\top G/n) \leq \sigma^2_{min}\left(U_I^\top G/\sqrt{n}\right) &= \sigma^2_{min}\left([I \quad U_I^\top U_{I^c}] \begin{bmatrix} T_I & 0 \\ 0 & T_{I^c} \end{bmatrix}\right) \\ &\leq \sigma^2_{min}\left([I \quad U_I^\top U_{I^c}]\right) \sigma^2_{max}\left(\begin{bmatrix} T_I & 0 \\ 0 & T_{I^c} \end{bmatrix}\right) \\ &= \sigma^2_{min}\left([I \quad U_I^\top U_{I^c}]\right) \max\left(\sigma^2_{max}(T_I), \sigma^2_{max}(T_{I^c})\right) \\ &\leq \sigma^2_{min}\left([I \quad U_I^\top U_{I^c}]\right) c_{max}. \end{aligned}$$

Observe now that $\sigma^2_{min}\left([I \quad U_I^\top U_{I^c}]\right) = 1 - \cos^2(\theta_{min})$ so that

$$c_{min} \leq (1 - \cos^2(\theta_{min}))c_{max}$$

and, therefore,

$$\cos^2(\theta_{min}) \leq 1 - \frac{c_{min}}{c_{max}}$$

from which the thesis follows. ■

Lemma 20 *Assume that the spectrum of G satisfies* (25)*. For any index i, let $I_1^{(i)}$ and $I_0^{(i)}$ be as in* (33)*. Finally, assume $a\lambda_j^n$, which may depend on n, are bounded and satisfy:*

$$\lim_{n\to\infty} f_n = +\infty \quad \text{where} \quad f_n := \min_{j \in I_1^{(i)}} n\lambda_j^n. \tag{47}$$

Then, conditioned on θ, $\epsilon_n^{(i)}$ in (30) *and* (29) *can be decomposed as*

$$\epsilon_n^{(i)} = m_{\epsilon_n}(\theta) + v_{\epsilon_n}.$$

The following conditions hold:

$$\mathbb{E}_v\left[\epsilon_n^{(i)}\right] = m_{\epsilon_n}(\theta) = O_P\left(\frac{1}{\sqrt{f_n}}\right) \qquad v_{\epsilon_n} = O_P\left(\frac{1}{\sqrt{n}}\right) \tag{48}$$

so that $\epsilon_n^{(i)}|\theta$ *converges to zero in probability (as* $n \to \infty$*). In addition*

$$Var_v\{\epsilon_n^{(i)}\} = \mathbb{E}_v\left[v_{\epsilon_n} v_{\epsilon_n}^\top\right] = O_P\left(\frac{1}{n}\right). \tag{49}$$

If in addition[5]

$$n^{1/2}\frac{\left(G^{(i)}\right)^\top G^{(j)}}{n} = O_P(1) \;;\; j \in I_1^{(i)} \tag{50}$$

then

$$m_{\epsilon_n}(\theta) = O_P\left(\frac{1}{\sqrt{n f_n}}\right). \tag{51}$$

Proof Consider the Singular Value Decomposition

$$\bar{P}_1 \bar{S}_1 \bar{P}_1^\top := \frac{1}{n}\sum_{j\in I_1} G^{(j)}\left(G^{(j)}\right)^\top \lambda_j^n. \tag{52}$$

Using (47), there exist $\bar{n}$ so that, $\forall\ n > \bar{n}$ we have $0 < \lambda_j^n \leq M < \infty$, $j \in I_1^{(i)}$. Otherwise, we could find a subsequence n_k so that $\lambda_j^{n_k} = 0$ and hence $n_k \lambda_j^{n_k} = 0$, contradicting (47). Therefore, the matrix $\bar{P}_1$ in (52) is an orthonormal basis for the space $\mathcal{G}_1 := col\ span\{G^{(j)}/\sqrt{n} : j \in I_1^{(i)}\}$. Let also $T^{(j)}$ be such that $G^{(j)}/\sqrt{n} = \bar{P}_1 T^{(j)}$, $j \in I_1^{(i)}$. Note that by assumption (25) and lemma 19

$$\|T^{(j)}\| = O_P(1) \quad \forall\, j \in I_1^{(i)}. \tag{53}$$

Consider now the Singular Value Decomposition

$$\begin{aligned}\begin{bmatrix} P_1 & P_0 \end{bmatrix}\begin{bmatrix} S_1 & 0 \\ 0 & S_0 \end{bmatrix}\begin{bmatrix} P_1^\top \\ P_0^\top \end{bmatrix} &:= \underbrace{\frac{1}{n}\sum_{j\in I_1} G^{(j)}\left(G^{(j)}\right)^\top \lambda_j^n}_{\bar{P}_1\bar{S}_1\bar{P}_1^\top} + \underbrace{\frac{1}{n}\sum_{j\in I_0} G^{(j)}\left(G^{(j)}\right)^\top \lambda_j^n}_{\Delta.} \\ &= \bar{P}_1\bar{S}_1\bar{P}_1^\top + \Delta.\end{aligned} \tag{54}$$

For future reference note that $\exists T_{\bar{P}_1} \;:\; \bar{P}_1 = \begin{bmatrix} P_1 & P_0 \end{bmatrix} T_{\bar{P}_1}$. Now, from (28) we have that

$$\frac{\Sigma_{\bar{v}}^{(i)\,-1} G^{(i)}}{\sqrt{n}} V_n^{(i)}\left(D_n^{(i)}\right)^{-1} = \Sigma_{\bar{v}}^{(i)\,-1/2} U_n^{(i)}. \tag{55}$$

Using (55) and defining

$$P := \begin{bmatrix} P_1 & P_0 \end{bmatrix} \quad S := \begin{bmatrix} S_1 & 0 \\ 0 & S_0 \end{bmatrix},$$

5. This is equivalent to say that the columns of $G^{(j)}$, $j = 1, .., k$, $j \neq i$ are asymptotically orthogonal to the columns of $G^{(i)}$.

Equation (30) can be rewritten as:

$$\begin{aligned}
\epsilon_n^{(i)} &= \left(U_n^{(i)}\right)^\top \frac{{\Sigma_{\bar{v}}^{(i)}}^{-1/2}\bar{v}}{\sqrt{n}} \\
&= \left(D_n^{(i)}\right)^{-1}\left(V_n^{(i)}\right)^\top \frac{\left(G^{(i)}\right)^\top}{\sqrt{n}}{\Sigma_{\bar{v}}^{(i)}}^{-1}\frac{\bar{v}}{\sqrt{n}} \\
&= \left(D_n^{(i)}\right)^{-1}\left(V_n^{(i)}\right)^\top \frac{\left(G^{(i)}\right)^\top}{\sqrt{n}}\left[\begin{array}{cc} P & P_\perp \end{array}\right]\left[\begin{array}{cc} (nS+\sigma^2 I)^{-1} & 0 \\ 0 & \sigma^{-2}I \end{array}\right]\left[\begin{array}{c} P^\top \\ P_\perp^\top \end{array}\right]\frac{\bar{v}}{\sqrt{n}} \\
&= \left(D_n^{(i)}\right)^{-1}\left(V_n^{(i)}\right)^\top \frac{\left(G^{(i)}\right)^\top}{\sqrt{n}}\left[\begin{array}{cc} P & P_\perp \end{array}\right]\left[\begin{array}{cc} (nS+\sigma^2 I)^{-1} & 0 \\ 0 & \sigma^{-2}I \end{array}\right]\left[\begin{array}{c} P^\top \\ P_\perp^\top \end{array}\right] \\
&\times \left[\sum_{j\in I_1^{(i)}}\frac{G^{(j)}}{\sqrt{n}}\theta^{(j)}+\frac{v}{\sqrt{n}}\right] \\
&= \underbrace{\left(D_n^{(i)}\right)^{-1}\left(V_n^{(i)}\right)^\top \frac{\left(G^{(i)}\right)^\top}{\sqrt{n}}P(nS+\sigma^2 I)^{-1}\left[\begin{array}{c} P_1^\top P_1 \\ P_0^\top P_1 \end{array}\right]\sum_{j\in I_1}T^{(j)}\theta^{(j)}}_{m_{\epsilon_n}(\theta)} + \\
&\quad \underbrace{+\left(D_n^{(i)}\right)^{-1}\left(V_n^{(i)}\right)^\top \frac{\left(G^{(i)}\right)^\top}{\sqrt{n}}\left[\begin{array}{cc} P & P_\perp \end{array}\right]\left[\begin{array}{cc} (nS+\sigma^2 I)^{-1} & 0 \\ 0 & \sigma^{-2}I \end{array}\right]\frac{v_{\bar{P}}}{\sqrt{n}}}_{v_{\epsilon_n}}
\end{aligned}$$

where the last equation defines $m_{\epsilon_n}(\theta)$ and v_{ϵ_n}, the noise

$$v_{\bar{P}} := \left[\begin{array}{c} P^\top \\ P_\perp^\top \end{array}\right] v$$

is still a zero mean Gaussian noise with variance $\sigma^2 I$ and $\frac{G^{(j)}}{\sqrt{n}} = \bar{P}_1 T^{(j)}$ provided $j \neq i$. Note that m_{ϵ_n} does not depend on v and that $\mathbb{E}_v v_{\epsilon_n} = 0$. Therefore $m_{\epsilon_n}(\theta)$ is the mean (when only noise v is averaged out) of ϵ_n. As far as the asymptotic behavior of $m_{\epsilon_n}(\theta)$ is concerned, it is convenient to first observe that

$$(nS+\sigma^2 I)^{-1}\left[\begin{array}{c} P_1^\top \bar{P}_1 \\ P_0^\top \bar{P}_1 \end{array}\right] = \left[\begin{array}{c} (nS_1+\sigma^2 I)^{-1}P_1^\top \bar{P}_1 \\ (nS_0+\sigma^2 I)^{-1}P_0^\top \bar{P}_1 \end{array}\right]$$

and that the second term on the right hand side can be rewritten as

$$(nS_0+\sigma^2 I)^{-1}P_0^\top \bar{P}_1 = \left[\begin{array}{c} (n[S_0]_{1,1}+\sigma^2)^{-1}P_{0,1}^\top \bar{P}_1 \\ (n[S_0]_{22}+\sigma^2)^{-1}P_{0,2}^\top \bar{P}_1 \\ \vdots \\ (n[S_0]_{m-k,m-k}+\sigma^2)^{-1}P_{0,m-k}^\top \bar{P}_1 \end{array}\right] \tag{56}$$

where $[S_0]_{ii}$ is the $i-th$ diagonal element of S_0 and $P_{0,i}$ is the $i-th$ column of P_0. Now, using Equation (54) one obtains that

$$\begin{aligned}
n[S_0]_{ii} = P_{0,i}^\top P n S P^\top P_{0,i} &= P_{0,i}^\top\left(\bar{P}_1 n\bar{S}_1\bar{P}_1^\top + n\Delta\right)P_{0,i} \\
&\geq P_{0,i}^\top \bar{P}_1 n\bar{S}_1\bar{P}_1^\top P_{0,i} \\
&\geq \sigma_{min}(n\bar{S}_1)P_{0,i}^\top\bar{P}_1\bar{P}_1^\top P_{0,i} \\
&= \sigma_{min}(n\bar{S}_1)\|P_{0,i}^\top\bar{P}_1\|^2.
\end{aligned}$$

An argument similar to that used in (42) shows that

$$\sigma_{min}(n\bar{S}_1) \geq c_{min}\min\{n\lambda_j^n,\ j \in I_1^{(i)}\} = c_{min} f_n \tag{57}$$

also holds true; denoting $\|P_{0,i}^\top \bar{P}_1\| = g_n$, the generic term on the right hand side of (56) satisfies

$$\begin{array}{rcl} \|(n[S_0]_{ii} + \sigma^2)^{-1} P_{0,i}^\top \bar{P}_1\| & \leq & \frac{\|P_{0,i}^\top \bar{P}_1\|}{n\sigma_{min}(\bar{S}_1)\|P_{0,i}^\top \bar{P}_1\|^2 + \sigma^2} \\ & \leq & k\ \min(g_n, (f_n g_n)^{-1}) \\ & = & \frac{k}{\sqrt{f_n}} \min(\sqrt{f_n} g_n, (\sqrt{f_n} g_n)^{-1}) \\ & \leq & \frac{k}{\sqrt{f_n}} \end{array} \tag{58}$$

for some positive constant k. Now, using Lemma 13, $D_n^{(i)}$ is bounded and bounded away from zero in probability, so that $\|D_n^{(i)}\| = O_P(1)$ and $\| \left(D_n^{(i)}\right)^{-1} \| = O_P(1)$. In addition, $V_n^{(i)}$ is an orthonormal matrix and $\|\frac{G^{(i)}}{\sqrt{n}}\| = O_P(1)$. Last, using (57) and (25), we have $\|(nS_1 + \sigma^2 I)^{-1}\| = O_P(1/n)$. Combining these conditions with (53) and (58), we obtain the first expression in (48). As far as the asymptotics on v_{ϵ_n} are concerned, it suffices to observe that

$$w_n^\top v_{\bar{P}}/\sqrt{n} = O_P(1/\sqrt{n}) \ \text{ if : } \ \|w_n\| = O_P(1).$$

The variance (w.r.t. noise v) $Var_v\{\epsilon_n\} = \mathbb{E}_v\left[v_{\epsilon_n} v_{\epsilon_n}^\top\right]$ satisfies

$$Var_v\{\epsilon_n\} = \frac{\sigma^2}{n} \left(U_n^{(i)}\right)^\top \Sigma_{\bar{v}}^{(i)^{-1}} \left(U_n^{(i)}\right)$$

so that, using the condition $\left\| \Sigma_{\bar{v}}^{(i)^{-1}} \right\| = \sigma^{-2}$ derived in Lemma 13, and the fact that $U_n^{(i)}$ has orthonormal columns, the condition $Var_v\{\epsilon_n\} = O_P\left(\frac{1}{n}\right)$ in (49) follows immediately.

If, in addition, (50) holds then (53) becomes

$$\|T^{(j)}\| = O_P(1/\sqrt{n}) \quad j = 1, ..., k; \quad j \neq k$$

so that an extra $\sqrt{n}$ appears in the denominator in the expression of $m_\epsilon(\theta)$ yielding (51). This concludes the proof. ■

Before we proceed, we review a useful characterization of convergence. While it can be stated for many types of convergence, we present it specifically for convergence in probability, since this is the version we will use.

Lemma 21 *The sequence a^n converges in probability to a (written $a^n \to_p a$) if and only if every subsequence $a^{n(j)}$ of a^n has a further subsequence $a^{n(j(k))}$ with $a^{n(j(k))} \to_p a$.*

Proof If $a^n \to_p a$, this means that for any $\epsilon > 0$, $\delta > 0$ there exists some $n_{\epsilon,\delta}$ such that for all $n \geq n_{\epsilon,\delta}$, we have $P(|a^n - a| > \epsilon) \leq \delta$. Clearly, if $a^n \to_p a$, then $a^{n(j)} \to_p a$ for every subsequence $a^{n(j)}$ of a^n. We prove the other direction by contrapositive.

Assume that $a^n \not\to_p a$. That means precisely that there exist some $\epsilon > 0, \delta > 0$ and a subsequence $a^{n(j)}$ so that $P(|a - a^{n(j)}| > \epsilon) \geq \delta$. Therefore the subsequence $a^{n(j)}$ cannot have further subsequences that converge to a in probability, since every term of $a^{n(j)}$ stays ϵ-far away from a with positive probability δ. ■

Lemma 21 plays a major role in the proof of the main result.

A.6 Proof of Theorem 14

Since the hypotheses of Lemma 13 hold, we know $w_{k,n} \to 0$ in (32). Then Lemma 18 guarantees that condition (47) holds true (in probability) so that Lemma 20 applies, and therefore $v_{k,n} \to_p 0$ in (32). We now give the proofs of results 1-4 in Theorem 14.

1. The reader can quickly check that $\frac{d}{d\gamma}\bar{\lambda}_1^\gamma < 0$, so $\bar{\lambda}_1^\gamma$ is decreasing in γ. The limit calculation follows immediately from L'Hopital's rule yielding $\lim_{\gamma\to 0^+} \lambda_1^\gamma = \bar{\lambda}_1$.

2. We use the convergence characterization given in Lemma 21. Pick any subsequence $\hat{\lambda}_1^{n(j)}$ of $\hat{\lambda}_1^n$. Since $\{V_{n(j)}\}$ is bounded, by Bolzano-Weierstrass it must have a convergent subsequence $V_{n(j(k))} \to V$, where V satisfies $V^TV = I$ by continuity of the 2-norm. The first-order optimality conditions for $\hat{\lambda}_1^n > 0$ are given by

$$0 = f_1(\lambda, w, v, \eta) = \frac{1}{2}\sum_{k=1}^{k_1} \frac{-\eta_k^2 - v_k}{(\lambda + w_k)^2} + \frac{1}{\lambda + w_k} + \gamma \,, \tag{59}$$

   and we have $f_1(\lambda, 0, 0, V^T\bar{\theta}^{(1)}) = 0$ if and only if $\lambda = \bar{\lambda}_1^\gamma$. Taking the derivative we find

$$\frac{d}{d\lambda} f_1(\lambda, 0, 0, V^T\bar{\theta}^{(1)}) = \frac{\|\bar{\theta}^{(1)}\|^2}{\lambda^3} - \frac{k_1}{2\lambda^2} \,,$$

   which is nonzero at λ_1^γ for any γ, since the only zero is at $2\frac{\|\bar{\theta}^{(1)}\|^2}{k_1} = 2\bar{\lambda}_1 \geq 2\bar{\lambda}_1^\gamma$.

   Applying the Implicit Function Theorem to f at $\left(\lambda_1^\gamma, 0, 0, V^\top\bar{\theta}^{(1)}\right)$ yields the existence of neighborhoods $\mathcal{U}$ of $(0, 0, V^\top\bar{\theta}^{(1)})$ and $\mathcal{W}$ of λ_1^γ such that

$$f(\phi(w, v, \eta), w, v, \eta) = 0 \qquad \forall\, (w, v, \eta) \in \mathcal{U} \,.$$

   In particular, $\phi(0, 0, V^\top\bar{\theta}^{(1)}) = \lambda_1^\gamma$. Since $(w_{n(j(k))}, v_{n(j(k))}, \eta_{n(j(k))}) \to_p (0, 0, V^\top\bar{\theta}^{(1)})$, we have that for any $\delta > 0$ there exist some k_δ so that for all $n(j(k)) > n(j(k_\delta))$ we have $P((w_{n(j(k)}, v_{n(j(k))}, \eta_{n(j(k))}) \notin \mathcal{U}) \leq \delta$. For anything in $\mathcal{U}$, by continuity of ϕ we have

$$\hat{\lambda}_1^{n(j(k))} = \phi(w_{n(j(k))}, v_{n(j(k))}, \eta_{n(j(k))}) \to_p \phi(0, 0, V^\top\bar{\theta}^{(1)}) = \lambda_1^\gamma \,.$$

   These two facts imply that $\hat{\lambda}_1^{n(j(k))} \to_p \lambda_1^\gamma$. We have shown that every subsequence $\hat{\lambda}_1^{n(j)}$ has a further subsequence $\hat{\lambda}_1^{n(j(k))} \to_p \lambda_1^\gamma$, and therefore $\hat{\lambda}_1^n \to_p \lambda_1^\gamma$ by Lemma 21.

3. In this case, the only zero of (59) with $\gamma = 0$ is found at $\bar{\lambda}_1$, and the derivative of the optimality conditions is nonzero at this estimate, by the computations already given. The result follows by the implicit function theorem and subsequence argument, just as in the previous case.

4. Rewriting the derivative (59)

$$\frac{1}{2}\sum_{k=1}^{k_1} \frac{\lambda - v_k - \eta_k^2 + w_k}{(\lambda + w_k)^2} + \gamma \,,$$

   we observe that for any positive λ, the probability that the derivative is positive tends to one. Therefore the minimizer λ_1^γ converges to 0 in probability, regardless of the value of γ.

References

A. Aravkin, J. Burke, A. Chiuso, and G. Pillonetto. On the estimation of hyperparameters for empirical bayes estimators: Maximum marginal likelihood vs minimum MSE. In *Proc. IFAC Symposium on System Identification (SysId 2012)*, 2012.

F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the 21st International Conference on Machine Learning*, page 4148, 2004.

F.R. Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

D. Bauer. Asymptotic properties of subspace estimators. *Automatica*, 41:359–376, 2005.

J.O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer, second edition, 1985.

E. Candes and T. Tao. The Dantzig selector: statistical estimation when p is much larger than n. *Annals of Statistics*, 35:2313–2351, 2007.

F. Chatelin. *Spectral Approximation of Linear Operators*. Academic Press, NewYork, 1983.

T. Chen, H. Ohlsson, and L. Ljung. On the estimation of transfer functions, regularization and gaussian processes - revisited. In *IFAC World Congress 2011*, Milano, 2011.

A. Chiuso and G. Pillonetto. Nonparametric sparse estimators for identification of large scale linear systems. In *Proceedings of IEEE Conf. on Dec. and Control*, Atlanta, 2010a.

A. Chiuso and G. Pillonetto. Learning sparse dynamic linear systems using stable spline kernels and exponential hyperpriors. In *Proceedings of Neural Information Processing Symposium*, Vancouver, 2010b.

A. Chiuso and G. Pillonetto. A Bayesian approach to sparse dynamic network identification. *Automatica*, 48:1553–1565, 2012.

D. Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 52(4):1289–1306, 2006.

B. Efron. Microarrays, empirical Bayes and the two-groups model. *Statistical Science*, 23: 122, 2008.

B. Efron and C. Morris. Stein's estimation rule and its competitors–an empirical bayes approach. *Journal of the American Statistical Association*, 68(341):117–130, 1973.

B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, december 2001.

T. J. Hastie and R. J. Tibshirani. Generalized additive models. In *Monographs on Statistics and Applied Probability*, volume 43. Chapman and Hall, London, UK, 1990.

Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1994.

W. James and C. Stein. Estimation with quadratic loss. In *Proc. 4th Berkeley Sympos. Math. Statist. and Prob., Vol. I*, pages 361–379. Univ. California Press, Berkeley, Calif., 1961.

H. Leeb and B. Pötscher. Model selection and inference: Facts and fiction. *Econometric Theory*, 21:2159, 2005.

D.J.C. Mackay. Bayesian non-linear modelling for the prediction competition. *ASHRAE Trans.*, 100(2):3704–3716, 1994.

J. S. Maritz and T. Lwin. *Empirical Bayes Method*. Chapman and Hall, 1989.

T. Park and G. Casella. The Bayesian Lasso. *Journal of the American Statistical Association*, 103(482):681–686, June 2008.

G. Pillonetto and G. De Nicolao. A new kernel-based approach for linear system identification. *Automatica*, 46(1):81–93, 2010.

G. Pillonetto, F. Dinuzzo, and G. De Nicolao. Bayesian online multitask learning of Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2).

G. Pillonetto, A. Chiuso, and G. De Nicolao. Prediction error identification of linear systems: a nonparametric Gaussian regression approach. *Automatica*, 45(2):291–305, 2011.

M. Schmidt, E. Van Den Berg, M. P. Friedlander, and Kevin Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. In *Proc. of Conf. on Artificial Intelligence and Statistics*, pages 456–463, 2009.

C.M. Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9(6):1135–1151, 1981.

R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society, Series B.*, 58, 1996.

M. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

M.K. Titsias and M. Lzaro-Gredilla. Spike and slab variational inference for multi-task and multiple kernel learning. *Advances in Neural Information Processing Systems 25 (NIPS 2011)*, 2011.

R. Tomioka and T. Suzuki. Regularization strategies and empirical bayesian learning for MKL. *Journal of Machine Learning Research*, 2011.

D.P. Wipf and S. Nagarajan. A new view of automatic relevance determination. In *Proc. of NIPS*, 2007.

D.P. Wipf and B.D. Rao. An empirical bayesian strategy for solving the simultaneous sparse approximation problem. *IEEE Transactions on Signal Processing*, 55(7):3704–3716, 2007.

D.P. Wipf, B.D. Rao, and S. Nagarajan. Latent variable Bayesian models for promoting sparsity. *IEEE Transactions on Information Theory*, 57(9):6236–6255, 2011.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006.

P. Zhao and B. Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7:2541–2563, Nov. 2006.

H. Zou. The adaptive Lasso and it oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

Journal of Machine Learning Research 15 (2014) 253-282 Submitted 1/12; Revised 6/13; Published 1/14

Using Trajectory Data to Improve Bayesian Optimization for Reinforcement Learning

Aaron Wilson* AARON.WILSON@PARC.COM
PARC, a Xerox company
3333 Coyote Hill Road
Palo Alto, CA 94304 USA

Alan Fern AFERN@EECS.OREGONSTATE.EDU
Prasad Tadepalli TADEPALL@EECS.OREGONSTATE.EDU
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR 97331-5501, USA

Editor: Joelle Pineau

Abstract

Recently, Bayesian Optimization (BO) has been used to successfully optimize parametric policies in several challenging Reinforcement Learning (RL) applications. BO is attractive for this problem because it exploits Bayesian prior information about the expected return and exploits this knowledge to select new policies to execute. Effectively, the BO framework for policy search addresses the exploration-exploitation tradeoff. In this work, we show how to more effectively apply BO to RL by exploiting the sequential trajectory information generated by RL agents. Our contributions can be broken into two distinct, but mutually beneficial, parts. The first is a new Gaussian process (GP) kernel for measuring the similarity between policies using trajectory data generated from policy executions. This kernel can be used in order to improve posterior estimates of the expected return thereby improving the quality of exploration. The second contribution, is a new GP mean function which uses learned transition and reward functions to approximate the surface of the objective. We show that the model-based approach we develop can recover from model inaccuracies when good transition and reward models cannot be learned. We give empirical results in a standard set of RL benchmarks showing that both our model-based and model-free approaches can speed up learning compared to competing methods. Further, we show that our contributions can be combined to yield synergistic improvement in some domains.

Keywords: reinforcement learning, Bayesian, optimization, policy search, Markov decision process, MDP

1. Introduction

In the policy search setting, RL agents seek an optimal policy within a fixed set. The agent iteratively selects new policies, executes selected policies, and estimates each individual policy performance. Naturally, future policy selection decisions should benefit from the

∗. Work was performed at Oregon State University.

information generated by previous selections. A question arises regarding how the performance of untried policies can be estimated using this data, and how to use the estimates to direct the selection of a new policy. Ideally, the process of selecting new policies accounts for the agent's uncertainty in the performance estimates, and directs the agent to explore new parts of the policy space where uncertainty is high. Bayesian Optimization (BO) tackles this problem. It is a method of planning a sequence of queries from an unknown objective function for the purpose of seeking the maximum. In BO uncertainty in the objective function is encoded in a Bayesian prior distribution that estimates the performance of policies. Because the method is Bayesian uncertainty in the estimated values is explicitly encoded. After policy executions a posterior distribution over the objective is computed, and this posterior is used to guide the exploration process.

Success of BO relies on the quality of the objective function model. In this work, similar to past efforts applying BO to RL, we focus on GP models of the expected return (Rasmussen and Williams, 2005). The generalization performance of Gaussian process (GP) models, and hence the performance of the BO technique, is strongly impacted by the definition of both the GP mean function, and the kernel function encoding relatedness between points in the function space.

Prior work applying BO to the RL problem tackled difficult problems of gait optimization and vehicle control, but ignored the sequential nature of the decision process (Lizotte et al., 2007; Lizotte, 2008). Execution of a policy generates a trajectory represented by a sequence of action/observation/reward tuples. Previously, this information was reduced to a Monte-Carlo estimate of the expected return and all other information present in the observed trajectories was discarded. By taking advantage of the sequential process, we argue, and empirically demonstrate, that our methods dramatically improve the data efficiency of BO methods for RL. To take advantage of trajectory information we propose two complementary methods. First, we discuss a new kernel function which uses trajectory information to measure the relatedness of policies. Second, we discuss the incorporation of approximate domain models into the basic BO framework.

Our first contribution, is a notion of relatedness tailored for the RL context. Past work has used simple kernels to relate policy parameters. For instance, squared exponential kernels were used by Lizotte et al. (2007), Lizotte (2008) and Brochu et al. (2009). These kernels relate policies by differences in policy parameter values. We propose that policies are better related by their *behavior* rather than their parameters. We use a simple definition of policy behavior, and motivate an information-theoretic measure of policy similarity. Additionally, we show that the measure of similarity can be estimated without learning the transition and reward functions.

Our second contribution incorporates learned domain models (the transition and reward function) into the BO framework. Learned domain models are used to simulate Monte-Carlo policy roll-outs for the purpose of estimating policy returns. Crucially, we consider the setting where the simulator *is not* an accurate model of the true domain. The domain model class may have significant bias that prevents close approximation of the true transition and reward functions. Consequently, Monte-Carlo simulations of the environment, using the learned functions, can produce substantial errors that prevent the direct application of standard model-based RL algorithms. To overcome this problem, we propose using the GP model to correct for errors introduced by the poor domain model approximations, and show

empirically that our algorithm successfully uses the learned transition and reward models to quickly identify high quality policies.

In the following sections we discuss the general problem of BO, motivate our modeling efforts, and discuss how to incorporate our changes into BO algorithms. We conclude with a discussion of empirical evaluation of the new algorithms on five benchmark RL domains.

2. Reinforcement Learning and Bayesian Optimization

We study the reinforcement learning problem in the context of Markov decision processes (MDPs). MDPs are described by a tuple (S, A, P, P_0, R). We consider processes with continuous state spaces and discrete action spaces. Each state $s \in S$ is a vector of real values. Each action $a \in A$ represents a discrete choice available to the agent. The transition function P is a probability distribution $P(s_t|s_{t-1}, a_{t-1})$ that defines the probability of transitioning to state s_t, conditioned on the selected action a_{t-1}, and the current state s_{t-1}. Distribution $P_0(s_0)$ is the distribution over initial states. It defines the probability of the agent starting in state s_0. The reward function $R(s, a, s')$ returns a numeric value representing the immediate reward for the state, action, next state triplet. Finally, the agent selects actions according to a parametric policy π_θ. The policy is a stochastic mapping from states to actions $P_\pi(a|s, \theta)$ as a function of a vector of parameters $\theta \in \Re^k$.

We study episodic average reward RL. We define a trajectory to be a sequence of states and actions $\xi = (s_0, a_0, ..., a_{T-1}, s_T)$. Trajectories begin in an initial state s_0, and terminate after at most T steps. It follows that the probability of a trajectory is,

$$P(\xi|\theta) = P_0(s_0) \prod_{t=1}^{T} P(s_t|s_{t-1}, a_{t-1}) P_\pi(a_{t-1}|s_{t-1}, \theta).$$

This is the probability of observing a trajectory ξ given that the agent executes a policy with parameters θ. The value of a trajectory,

$$\bar{R}(\xi) = \sum_{t=0}^{T} R(s_t, a_t, s_{t+1}),$$

is simply the sum of rewards received. The variable T is assumed to have a maximum value ensuring that all trajectories have finite length.

We define our objective function to be the expected return,

$$\eta(\theta) = \int \bar{R}(\xi) P(\xi|\theta) d\xi.$$

The basic policy search problem is to identify the policy parameters that maximize this expectation,

$$\theta^* = \arg\max_\theta \eta(\theta).$$

In the RL problem setting the values of the expected return are not known to the agent. Likewise, the transition function, initial state distribution, and reward function are also unknown. This complicates the search for the maximum value.

3. Policy Search Using Bayesian Optimization

Bayesian optimization addresses the general problem of identifying the maximum of a real valued objective function,

$$\theta^* = \arg\max_{\theta} \eta(\theta).$$

This problem could be solved by optimizing the objective function directly. For instance, the Covariance Matrix Adaptation algorithm by Hansen (2006), which has already shown some promise in RL, directly searches the objective function by executing thousands of queries to identify the maximum. If each evaluation of the objective has a small cost, then thousands of evaluations could easily be performed. However, when individual evaluations of the objective incur high costs, algorithms which rely on many evaluations are inappropriate. Examples of domains with this property are easy to find. For example, the evaluation of airfoil design and engine components rely on running expensive finite element simulations of gas flow. These simulations have extreme time costs; evaluations of each design can take upwards of 24 hours. Other domains, familiar to RL researchers, include robot control. Running robots is time-consuming and increases the likelihood of physical failures. In cases like these it is best to minimize the number of objective function evaluations. This is the ideal setting for the application of BO. To reduce the number of function evaluations the BO approach uses a Bayesian prior model of the objective function and exploits this model to plan a sequence of objective function queries. Essentially, BO algorithms trade computational resources, expended to determine query points, for a reduced number of objective function evaluations.

Modeling the objective is a standard strategy in learning problems where the true function may be approximated by, for example, regression trees, neural networks, polynomials, and other structures that match properties of the target function. Using the parlance of RL, the Bayesian prior model of the objective function, sometimes called the surrogate function, can be viewed as a function approximator that supports Bayesian methods of analysis. However, where standard function approximators generalize across states the model of the objective function used in BO algorithms generalizes across policies. This is necessary to support intelligently querying the surrogate representation of the objective function.

The BO method plans a sequence of queries. The process proceeds as follows: (1) A query is selected. Queries are selected by optimizing a measure of improvement (to be defined below). Typically, the improvement measure incorporates an exploration strategy that directs search to poorly modeled regions of the solution space. (2) The query is evaluated by the true objective function. Real data is gathered from the system being optimized. Ideally, the computational resources expended in the previous step improves the quality of the observed data. (3) The system observes the performance at the query point and updates the posterior model of the objective function. (4) The process returns to step 1. Below we discuss the key components of BO algorithms including the improvement function and the prior model of the objective function (the central object of study in this paper).

3.1 Measure of Improvement

The selection criteria plays an important role in the quality of the exploration, and consequently the speed of identifying the optimal point. The basic problem of selection can be framed as identifying the point that minimizes the agent's expected risk,

$$\min_{\theta} \int \parallel \eta(\theta) - \eta(\theta^*) \parallel dP(\eta|D_{1:n}),$$

with respect to the posterior distribution (By minimizing the expected difference between $\eta(\theta)$ and the maximum $\eta(\theta^*) = \arg\max_\theta \eta(\theta)$ we maximize the value of $\eta(\theta)$). The data $D_{1:n}$ is a collection of pairs $D_{1:n} = \{\langle \theta_i, \eta(\theta_i)\rangle\}|_{i=1}^n$. Each pair is a previously selected policy point θ_i and the evaluated performance at that point $\eta(\theta_i)$. The posterior distribution $P(\eta|D_{1:n})$ encodes all of the agent's knowledge of the objective function. This risk functional is a natural foundation for a myopic iterative selection criteria,

$$\theta_{n+1} = \arg\min_{\theta} \int \parallel \eta(\theta) - \eta(\theta^*) \parallel dP(\eta|D_{1:n}).$$

Unfortunately, this selection criteria requires solving a computationally demanding minimax problem. A heuristic method of selection must be used.

A common heuristic called Maximum Expected Improvement (MEI) (Mockus, 1994) is the method of selection used in this work. The MEI heuristic compares new points to the point with highest observed return in the data set. We denote the value at this empirically maximal point to be η_{max}. Using this maximal value one can construct an *improvement* function,

$$I(\theta) = max\{0, \eta(\theta) - \eta_{max}\},$$

which is positive when $\eta(\theta)$ exceeds the current maximum and zero at all other points. The MEI criteria searches for the maximum of the expected improvement,

$$\theta_{n+1} = \arg\max_{\theta} E_{P(\eta|D_{1:n})}\left[I(\theta)\right].$$

Crucially, the expected improvement function exploits the posterior uncertainty. If the mean value at a new point is less than η_{max} the value of the Expected Improvement may still be greater than zero. Consider the case where the posterior distribution, $P(\eta(\theta)|D_{1:n})$, has probability mass on values exceeding η_{max}. In this case, the expected improvement will be positive. Therefore, the agent will explore until it is sufficiently certain that no other policy will improve on the best policy in the data set. Due to the empirical success of the MEI criterion it has become the standard choice in most work on BO.

When the posterior distribution is Gaussian then the expected improvement function has a convenient solution,

$$E_{P(\eta(\theta)|D_{1:n})}\left[I(\theta)\right] = \sigma(\theta)[\frac{\mu(\theta) - \eta_{max}}{\sigma(\theta)}\Phi(\frac{\mu(\theta) - \eta_{max}}{\sigma(\theta)}) + \phi(\frac{\mu(\theta) - \eta_{max}}{\sigma(\theta)})].$$

The functions $\mu(\theta)$ and $\sigma(\theta)$ are the mean and standard deviation of the Gaussian distribution. Function $\Phi(.)$ is the cumulative distribution function of the standard Gaussian distribution, and $\phi(.)$ is the probability distribution function. Please note that the expected improvement function is zero when the standard deviation is zero.

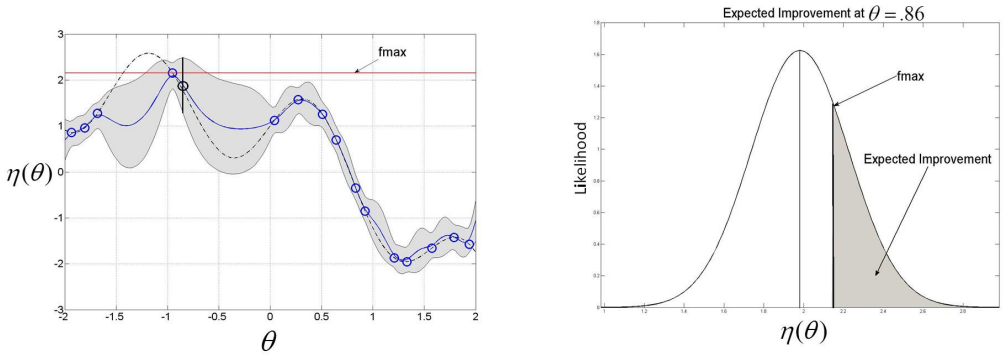


Figure 1: The expected improvement heuristic. The heuristic assesses the value of observing new points. On the left we consider the point circled (in black) at $\theta = -.86$. On the right we illustrate the expected improvement assuming that $P(\eta(\theta)|\theta = -.86)$ is Gaussian. To be considered an improvement the value of $\eta(\theta)$ must exceed the value of the current maximum $fmax$. The probability mass associated with the expected improvement is shaded. The expected improvement is proportional to the expected value of the indicated mass.

3.2 Objective Function Model

As a Bayesian method the performance of BO depends profoundly on the quality of the modeling effort. The specification of the prior distribution determines the nature of the posterior and hence the generalization performance of the surrogate representation. We elect to model the objective function using a Gaussian process (Rasmussen and Williams, 2005),

$$\eta(\theta) \sim GP(m(\theta), k(\theta, \theta')).$$

GP models are defined by a mean function $m(\theta)$ and a covariance function $k(\theta, \theta')$. The mean function specifies the expected value at a given point $m(\theta) = E[\eta(\theta)]$. Likewise, the covariance function estimates the covariance $k(\theta, \theta') = E[(\eta(\theta) - m(\theta))(\eta(\theta') - m(\theta'))]$ The kernel function encodes how correlated values of the objective are at points θ and θ'. Both of these functions encode knowledge of the underlying class of functions.

For the purpose of computing the improvement function described above, the posterior distribution at new points must be computed. In the GP model, this posterior has a simple form. Given the data $D_{1:n}$ the conditional posterior distribution is Gaussian with mean,

$$\mu(\eta(\theta_{n+1})|D_{1:n}) = m(\theta_{n+1}) - \mathbf{k}(\theta_{n+1}, \theta)\mathbf{K}(\theta, \theta)^{-1}(\mathbf{y} - \mathbf{m}),$$

where $\mathbf{m}$ is a vector of size n with elements $m(\theta_1), ..., m(\theta_n)$ and variance,

$$\sigma^2(\eta(\theta_{n+1})|D_{1:n}) = k(\theta_{n+1}, \theta_{n+1}) - \mathbf{k}(\theta_{n+1}, \theta)^t\mathbf{K}(\theta, \theta)^{-1}\mathbf{k}(\theta, \theta_{n+1}).$$

Define $\mathbf{y}$ to be the column vector of observed performances such that $y_i = \eta(\theta_i)$. Define $\mathbf{K}(\theta, \theta)$ to be the covariance matrix with elements $\mathbf{K}_{i,j} = k(\theta_i, \theta_j)$. Define $\mathbf{k}(\theta_{n+1}, \theta)$ to be

Algorithm 1 Bayesian Optimization Algorithm (BOA)

1: Let $D_{1:n} = \{(\theta_i, \hat{\eta}(\theta_i), \xi_i)\}|_{i=1}^{n}$.
2: **repeat**
3: Compute the matrix of covariances $\mathbf{K}$.
4: Select the next policy to evaluate: $\theta_{n+1} = \arg\max_\theta E_{P(\eta(\theta)|D)}[I(\theta)]$.
5: Execute the policy θ_{n+1} for E episodes.
6: Compute Monte-Carlo estimate of expected return $\hat{\eta}(\theta_{n+1}) = \frac{1}{E}\sum_{\xi\in\xi_{n+1}} \bar{R}(\xi)$
7: Update $D_{1:n+1} = D_{1:n} \cup (\theta_{n+1}, \hat{\eta}(\theta_{n+1}))$
8: **until** Convergence

the column vector of correlations such that the i^{th} element is $k(\theta_{n+1}, \theta_i)$ ($\mathbf{k}(\theta_{n+1}, \theta)^t$ is the transpose of this vector).

3.3 Bayesian Optimization for RL

Algorithm 1 outlines the basic loop of the BO routine discussed above. Line 1 assumes a batch of data of the form, $D_{1:n} = \{(\theta_i, \eta(\theta_i), \xi_i)\}|_{i=1}^{n}$. Hereafter, we will write D to indicate the collection of n data tuples. The Monte-Carlo estimate for policy θ_i is computed using a set of trajectories ξ_i sampled from the target system. Given data D, the surface of the expected return is modeled using the GP. The kernel matrix $\mathbf{K}$ is pre-computed in line 3 for reuse during maximization of the EI. Line 4 maximizes the EI. For this purpose, any appropriate optimization package can be used. To compute the expected improvement the GP posterior distribution $P(\eta(\theta_{n+1}|D))$ must be computed. This entails computing the mean function $m(\theta_{n+1})$, the vector of covariances $\mathbf{k}(\theta_{n+1}, \theta)$, and performing the required multiplications. Additional computational costs introduced into the mean and kernel functions will impact the computational cost of the optimization. Our modifications to the underlying model will increase these costs, but lead to more efficient search of the objective function space. The additional costs must be balanced against the cost of evaluating the objective function. Once selection is completed, new trajectories are generated from the selected policy (line 5), and an estimate of the expected return is recorded (lines 6 and 7).

4. Incorporating Trajectory Information into Bayesian Optimization for RL

We propose two complementary changes to the GP model of the expected return aimed at improving performance in RL. We define new covariance and mean functions specifically designed to exploit trajectory data. Section 4.1 details a new kernel function designed to compare policies in the RL context. The kernel uses a behavior-based measure of policy correlation. We motivate the use of this kernel and suggest a simple method for its estimation. Section 4.2 details our method of using a learned approximate Monte-Carlo simulator of policy performance. We detail how the outputs of this simulator are used to define a GP mean function and define a method for dealing with errors generated by the simulator.

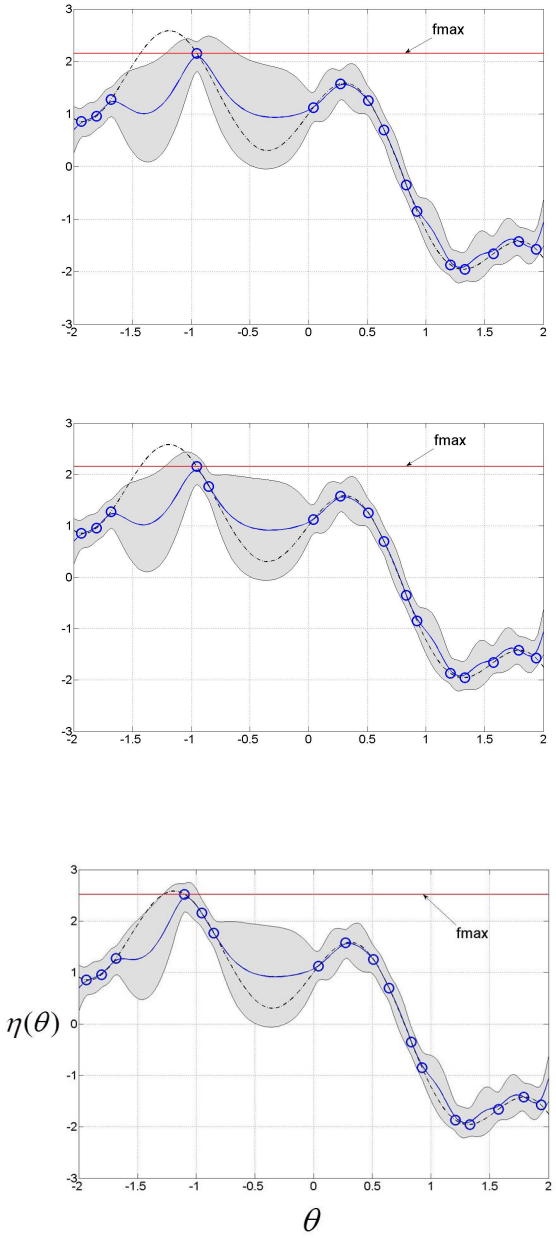


Figure 2: An illustration of Bayesian Optimization. The agent observes the objective function (dashed line) at a finite set of points (blue circles). Conditioned on the observations the agent maintains a model of the underlying objective function. The solid blue line depicts the mean of this model and the shaded regions illustrate the model uncertainty (2 standard deviations). Uncertainty is lower near densely sampled regions of θ space. The agent selects new data points for purposes of identifying the true maximum. As new observations are added the quality of the model improves and observations are focused near the maximal value.

4.1 Model-Free RL via Bayesian Optimization: A Behavior-Based Kernel (BBK)

In this section we design a kernel for the RL setting that leverages trajectory data to compute a domain independent measure of policy relatedness. Consider BO algorithms as a kind of space filling algorithm. Wherever sufficient uncertainty exists, the algorithm will aim to select a point to fill that space thereby reducing uncertainty in the vicinity of the selected point. The kernel function defines the volume to be filled. It is important for the development of BO algorithms for RL that kernel functions are robust to the parametrization of the policy space. Most kernel functions do not have this property. For instance, squared exponential kernels require that policies have a finite and fixed number of parameters. The kernel cannot compare non-parametric policies. In this case, individual policies can have distinct structures preventing their comparison using this form of kernel function. We seek a kernel function which is useful for comparing policies with distinct structural forms.

To construct an appropriate representation of uncertainty for the RL problem we propose relating policies by their behaviors. We define the behavior of a policy to be the associated trajectory density $P(\xi|\theta)$. Below, we discuss how to use the definition to construct a kernel function and how to estimate the kernel values without learning transition and reward functions.

To develop our kernel and demonstrate its relationship to the expected return we prove the following theorem:

Theorem 1 *For any θ_i, and θ_j, $Rmax \geq 0$*
$|\eta(\theta_i) - \eta(\theta_j)| \leq Rmax\sqrt{2}\left[\sqrt{KL(P(\xi|\theta_i)||P(\xi|\theta_j))} + \sqrt{KL(P(\xi|\theta_j)||P(\xi|\theta_i))}\right]$.

Proof: Below we establish the upper bound stated above. To begin we rewrite the absolute value of the difference in expected returns,

$$|\eta(\theta_i) - \eta(\theta_j)| = \left|\int \bar{R}(\xi)P(\xi|\theta_i)d\xi - \int \bar{R}(\xi)P(\xi|\theta_j)\,d\xi\right| = \left|\int \bar{R}(\xi)(P(\xi|\theta_i) - P(\xi|\theta_j))\,d\xi\right|.$$

By moving the absolute value into the integrand we upper bound the difference,

$$\left|\int \bar{R}(\xi)(P(\xi|\theta_i) - P(\xi|\theta_j))\,d\xi\right| \leq \int |\bar{R}(\xi)(P(\xi|\theta_i) - P(\xi|\theta_j))|\,d\xi.$$

We define a new quantity $Rmax$ bounding the trajectory reward from above. The trajectory reward is simply the sum of rewards at each trajectory step. $Rmax$ is defined to be the maximal trajectory value. Using the $Rmax$ quantity we construct a new bound,

$$\int |\bar{R}(\xi)(P(\xi|\theta_i) - P(\xi|\theta_j))|\,d\xi \leq Rmax \int |P(\xi|\theta_i) - P(\xi|\theta_j)|\,d\xi,$$

expressing the difference in returns as the product of a constant and a term depending only on the variational difference in the trajectory densities. An upper bound for the variational distance was developed by Pinsker (1964). The inequality states that

$\frac{1}{2}(V(P,Q))^2 \leq KL(P||Q)$ where V is the variational distance, $\int |(P(x) - Q(x))|dx$, and $KL(P,Q)$ is the Kullback Leibler divergence,

$$KL(P(\xi|\theta_i)||P(\xi|\theta_j)) = \int P(\xi|\theta_i) \log \left(\frac{P(\xi|\theta_i)}{P(\xi|\theta_j)} \right) d\xi.$$

We can use Pinsker's inequality to upper bound the variational distance,

$$Rmax \int |P(\xi|\theta_i) - P(\xi|\theta_j)| \, d\xi \leq Rmax \sqrt{2} \sqrt{KL(P(\xi|\theta_i)||P(\xi|\theta_j))}.$$

Finally, we use the fact that the variational distance is symmetric $\int |(P(x) - Q(x))|dx = \int |(Q(x) - P(x))|$,

$$\begin{aligned} |\eta(\theta_i) - \eta(\theta_j)| &\leq Rmax \sqrt{2} \sqrt{KL(P(\xi|\theta_i)||P(\xi|\theta_j))} \\ &\leq Rmax \sqrt{2} \left[\sqrt{KL(P(\xi|\theta_i)||P(\xi|\theta_j))} + \sqrt{KL(P(\xi|\theta_j)||P(\xi|\theta_i))} \right]. \end{aligned}$$

Hence, this simple bound relates the difference in returns of two policies to the trajectory density $\square$.

Importantly, the bound is a symmetric positive measure of the distance between policies. It bounds, from above, the absolute difference in expected value, and reaches zero only when the divergence is zero (the policies are the same). Additionally, computing the tighter variational bound, $Rmax \int |P(\xi|\theta_i) - P(\xi|\theta_j)| \, d\xi$, inherently requires knowledge of the domain transition models. Alternatively, the log term of the KL-divergence is a ratio of path probabilities. Given a sample of trajectories the ratio can be computed with no knowledge of the domain model. This characteristic is important when learned transition and reward functions are not available. Our goal is to incorporate the final measure of policy relatedness into the surrogate representation of the expected return. Unfortunately, the divergence function does not meet the standard requirements for a kernel (Moreno et al., 2004). To transform the bound into a valid kernel we first define a function,

$$D(\theta_i, \theta_j) = \sqrt{KL(P(\xi|\theta_i)||P(\xi|\theta_j))} + \sqrt{KL(P(\xi|\theta_j)||P(\xi|\theta_i))},$$

and define the covariance function to be the negative exponential of D,

$$K(\theta_i, \theta_j) = \exp(-\alpha \cdot D(\theta_i, \theta_j)).$$

The kernel has a single scalar parameter α controlling its width. This is precisely what we sought, a measure of policy similarity which depends on the action selection decisions. The kernel compares behaviors rather than parameters, making the measure robust to changes in policy parameterization.

4.2 Estimation of the Kernel Function Values

Below we discuss using estimates of the divergence values in place of the exact values for $D(\theta_i, \theta_j)$. Computing the exact KL-divergence requires access to a model of the decision process and is a computationally demanding process. No closed form solution is available.

The divergence must be estimated. In this work we elect to use a simple Monte-Carlo estimate of the divergence. The divergence between policy θ_i and θ_j is approximated by,

$$\hat{D}(\theta_i, \theta_j) = \sum_{\xi \in \xi_i} \log\left(\frac{P(\xi|\theta_i)}{P(\xi|\theta_j)}\right) + \sum_{\xi \in \xi_j} \log\left(\frac{P(\xi|\theta_j)}{P(\xi|\theta_i)}\right),$$

using a sparse sample of trajectories generated by each policy respectively (ξ_i represents the set of trajectories generated by policy θ_i). Because of the definition of the trajectory density, the term within the logarithm reduces to a ratio of action selection probabilities,

$$\begin{aligned}\log\left(\frac{P(\xi|\theta_i)}{P(\xi|\theta_j)}\right) &= \log\left(\frac{P_0(s_0)\prod_{t=1}^{T} P(s_t|s_{t-1},a_{t-1})P_\pi(a_{t-1}|\phi(s_{t-1}),\theta_i)}{P_0(s_0)\prod_{t=1}^{T} P(s_t|s_{t-1},a_{t-1})P_\pi(a_{t-1}|\phi(s_{t-1}),\theta_j)}\right) \\ &= \sum_{t=1}^{T} \log\left(\frac{P_\pi(a_t|s_t,\theta_i)}{P_\pi(a_t|s_t,\theta_j)}\right),\end{aligned}$$

and is easily estimated using trajectory data.

A problem arises when computing the Expected Improvement (Line 4 of the BOA). Computing the conditional mean and covariance for new points requires the evaluation of the kernel for policies which have no trajectories present in the data set. We elect to use an importance sampled estimate of the divergence, because we do not have access to learned transition and reward functions,

$$\hat{D}(\theta_{new}, \theta_j) = \sum_{\xi \in \xi_j} \frac{P(\xi|\theta_{new})}{P(\xi|\theta_j)} \log\left(\frac{P(\xi|\theta_{new})}{P(\xi|\theta_j)}\right) + \log\left(\frac{P(\xi|\theta_j)}{P(\xi|\theta_{new})}\right).$$

Though the variance of this estimate can be large, our empirical results show that errors in the divergence estimates, including the importance sampled estimates, do not negatively impact performance. Alternative methods of estimating f-divergences (KL-divergence being a specific case) have been proposed in the literature (Nguyen et al., 2007), and can be used for future implementations.

4.3 Model-Based RL via Bayesian Optimization

The behavior based kernel has some important limitations. First, due to the definition of the BBK the kernel can only compare stochastic policies. The KL divergence is meaningful when the conditional trajectory densities share the same support. Second, the upper bound used to construct the kernel function is loose. This can lead to excessive exploration when the kernel exaggerates the differences between policies. In this section, we introduce a distinct method of leveraging trajectory data that does not require stochastic policies and can leverage any appropriate kernel function (including the BBK).

Specifically, our Model-Based Bayesian Optimization Algorithm (MBOA) learns the initial state distribution, the transition function, and the reward function from the observed trajectory data. These learned functions are used to generate Monte-Carlo estimates of policy performance. To compute estimates of the expected return for policy θ we generate trajectory roll-outs. A roll-out is performed by sampling an initial state from the learned initial state distribution and then executing policy θ until termination. Simulated

trajectories are sampled from the approximate trajectory density,

$$\hat{P}(\xi|\theta) = \hat{P}_0(s_0)\prod_{t=1}^{T}\hat{P}(s_t|s_{t-1},a_{t-1})P_\pi(a_{t-1}|\phi(s_{t-1}),\theta).$$

We write $\hat{P}$ to indicate that the transition function and initial state distribution have been learned from the observed trajectory data. From a fixed sample of E simulated trajectories we compute a model-based Monte-Carlo estimate of the expected return,

$$m(\theta, D) = \hat{\eta}(\theta_i) = \frac{1}{E}\sum_{j=1}^{E}\hat{R}(\xi_j).$$

The function $\hat{R}$ indicates the learned reward function.

If the agent has learned accurate domain models an optimal policy can be learned by maximizing $\theta^* = \arg\max_\theta m(\theta, D)$. Unfortunately, in many domains it is difficult to specify and to learn accurate domain models. In the worst case, the domain model classes selected by the designer may not contain the true domain models. Moreover, the cost of sampling trajectories may become prohibitive as the complexity of the domain models increases. Therefore, we wish to allow the designer the flexibility of selecting a class of domain models that is simple, efficient, and possibly an inaccurate representation of the target system. In our work we propose a means of accurately estimating the expected return despite domain model errors.

To overcome problems stemming from the predictive errors we propose using $m(\theta, D)$ as the prior mean function for the GP model thereby modeling the deviations from this mean as a GP. The predictive distribution of the GP changes to be Gaussian with mean,

$$\mu(\eta(\theta_{n+1})|D) = m(\theta_{n+1}, D) + \mathbf{k}^t(\theta_{n+1},\theta)\mathbf{K}(\theta,\theta)^{-1}(\eta - \mathbf{m}(\theta, D)),$$

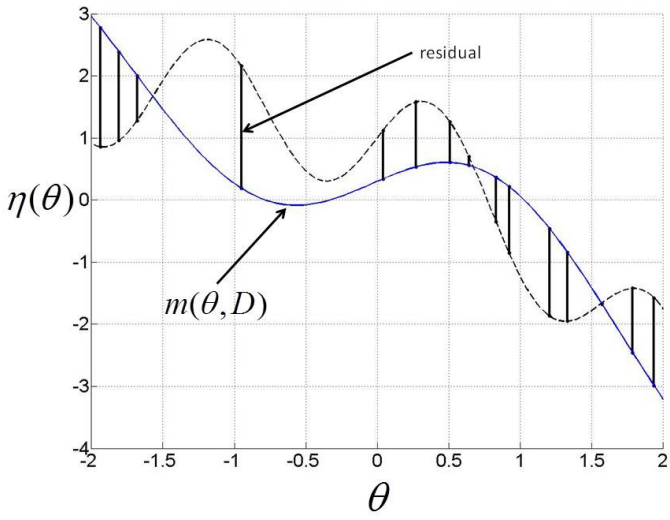
where $\mathbf{m}(\theta, D) = (m(\theta_1, D), ..., m(\theta_n, D))$ is a column vector of Monte-Carlo estimates with an element for each policy in D (This vector must be recomputed when new trajectories are added to the data). The variance remains unchanged. The new predictive mean is a sum of the Monte-Carlo approximation of the expected return and the GP's prediction of the residual. We illustrate the advantage of this model in Figure 3. As shown in the figure the model of the residuals directly compensates for errors introduced by the learned domain models.

In the case where the domain models cannot be effectively approximated, the model-based estimates of the expected return may badly skew the predictions. Consider the following degenerate case: $m(\theta, D_{1:n})$ underestimates the true mean for all policies resulting in zero expected improvement in the region of the optimal policy. In this case, the pessimistic estimates stifle the exploration of the policy space thereby preventing the discovery of the optimal solution. Our goal is to account for the domain model bias in a principled way.

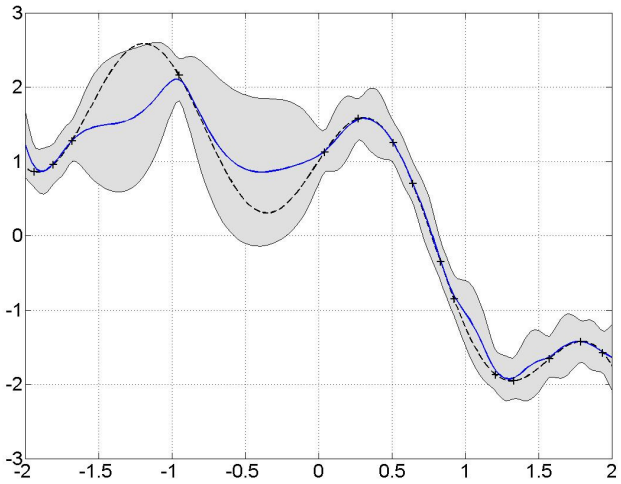
We propose a new model of the expected return,

$$\eta(\theta) = (1-\beta)\eta_1(\theta) + \beta\eta_2(\theta).$$

The new model is a convex combination of two functions governed by the parameter β. We model the function $\eta_1(\theta) \sim GP(0, k(\theta,\theta))$ with a zero mean GP, and we model the



(a)



(b)

Figure 3: (a) The relationship between the surface of $m(\theta, D)$ and the objective function. Both surfaces are observed at a fixed set of points. The values of the surfaces at these points compose the vectors η and $\mathbf{m}(\theta, D)$ respectively. The magnitude of the residuals are depicted as vertical bars. We build a GP model of these residual values. (b) The complete model combines the function $m(\theta, D)$ (as depicted above) with the GP model of the residuals. The solid blue line corresponds to the corrected mean of the model. The shaded area depicts two standard deviations from the mean.

function $\eta_2(\theta) \sim GP(m(\theta, D_{1:n}), k(\theta, \theta))$ with a GP distribution that uses the model-based mean $m(\theta, D)$ introduced above. The kernel of both GP priors is identical. The resulting distribution for $\eta(\theta) \sim GP(\beta m(\theta, D), c(\beta)k(\theta, \theta))$ is a GP with prior mean $\beta m(\theta, D)$ and covariance computed using the kernel function. This change impacts the predicted mean which now weights the model estimate by β,

$$\mu(\theta_{n+1}|D) = \beta m(\theta_{n+1}, D) + k(\theta_{n+1}, \theta)K(\theta, \theta)^{-1}(\eta(\theta) - \beta m(\theta, D)).$$

The variance changes to incorporate a factor $c(\beta)$.

We choose to optimize the β parameter using evidence maximization (Rasmussen and Williams, 2005). For a GP with mean βm and covariance matrix $\mathbf{K}$ the log likelihood of the data is,

$$P(\eta(\theta)|D) = -\frac{1}{2}(\eta(\theta) - \beta\mathbf{m}(\theta, D))^t K^{-1}(\eta(\theta) - \beta\mathbf{m}(\theta, D)) - \frac{1}{2}\log|K| - \frac{n}{2}\log(2\pi).$$

Taking the gradient of the log likelihood and solving for β results in a closed form solution,

$$\beta = \frac{\eta(\theta)^t K^{-1}\mathbf{m}(\theta, D))}{\mathbf{m}(\theta, D))^t K^{-1}\mathbf{m}(\theta, D))}.$$

Optimizing the value of β, prior to maximizing the expected improvement, allows the model to control the impact of the mean function on the predictions. This tradeoff is illustrated in Figure 4. Intuitively, the algorithm can return to the performance of the unmodified BOA when the domain models are poor (by setting β to zero).

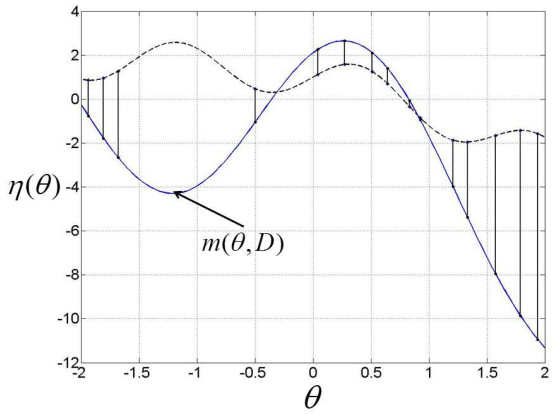
Algorithm 2 outlines the steps necessary to incorporate the new model into the BOA. MBOA takes advantage of trajectory data by building an approximate simulator of the expected return. Like the BBK the prior mean function used by the MBOA only requires policies to output actions. It is oblivious to the internal structure of policies. Additionally, by contrast to the BBK, MBOA can compare stochastic and deterministic policies.

Most work in model-based RL assumes the learned transition and reward functions are unbiased estimators of the true functions. This is difficult to guarantee in real-world tasks. Due to our limited knowledge transition and reward models frequently exhibit considerable model bias. MBOA aims to overcome sources of bias by combining a weighted mean function with a residual model. By construction MBOA can ignore the mean function if it produces systematic errors, due to model bias, and can benefit from the mean function where the estimates are accurate. In the results section we provide examples where MBOA benefits from the use of simple (biased) transition and reward models.

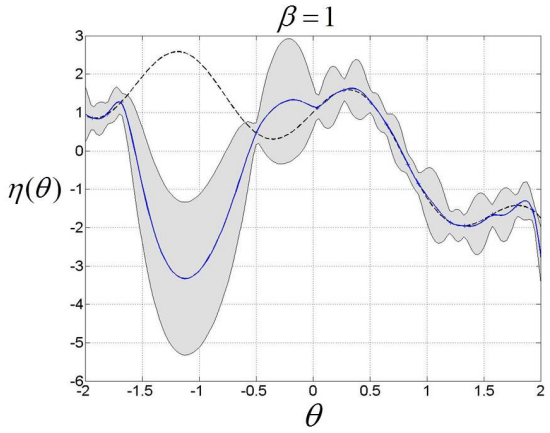
5. Experiment Results

We examine the performance of MBOA and BOA with the behavior based kernel (BBK) in 5 benchmark RL tasks including a mountain car task, a Cart-pole balancing task, a 3-link planar arm task, an acrobot swing up task, and a bicycle balancing task. Additional details about the Cart-pole, mountain car, and acrobot domains can be found in Sutton and Barto (1998). We use the bicycle simulator originally introduced by Randlov and Alstrom (1998).

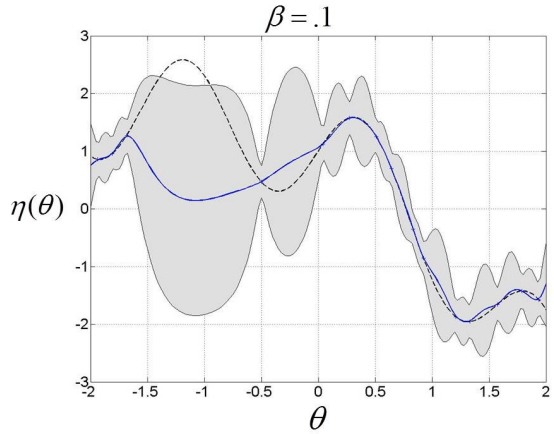
Comparisons are made between MBOA, BOA with the BBK, the DYNA-Q algorithm, PILCO (Deisenroth and Rasmussen, 2011), BOA with a squared exponential kernel (Lizotte, 2008), OLPOMDP (Baxter et al., 2001), Q-Learning with CMAC function approximation (Sutton and Barto, 1998), and LSPI (Lagoudakis et al., 2003).



(a) An illustration of the degenerate case where the mean function systematically underestimates the objective function (objective function marked by dashed line).



(b) The corrected model with control parameter $\beta = 1$. After correction the model underestimates the objective function.



(c) The corrected model with control parameter $\beta = .1$. By reducing the influence of the mean function the model more accurately estimates the true function.

Figure 4: Illustration of the impact of optimizing β.

Algorithm 2 Model-based Bayesian Optimization Algorithm (MBOA)

1: Let $D_{1:n} = \{\theta_i, \eta(\theta_i), \xi\}|_{i=1}^{n}$.
2: **repeat**
3: Learn the transition function, reward function, and initial state distribution from trajectory data.
4: Compute the vector $m(\theta, D_{1:n})$ using the approximate simulator.
5: Optimize β by maximizing the log likelihood.
6: Select the next point in the policy space to evaluate: $\theta_{n+1} = \arg\max_\theta E(I(\theta)|D_{1:n})$.
7: Execute the policy with parameters θ_{n+1} in the MDP.
8: Update $D_{1:n+1} = D_{1:n} \cup (\theta_{n+1}, \eta(\theta_{n+1}), \xi_{n+1})$
9: **until** Convergence.

5.1 Experiment Setup

We detail the special requirements necessary to implement each algorithm in this section.

For all experiments, except Cart-pole, the policy search algorithms search for parametric soft-max action selection policies,

$$P(a|s) = \frac{\exp(\theta_a \cdot f(s))}{\sum_{y \in A} \exp(\theta_y \cdot f(s))}.$$

The parameters θ_y are the set of policy parameters associated with action y. The function $f(s)$ computes features of the state.

In the Cart-pole experiments, a linear policy maps directly to the action,

$$a = \theta \cdot f(s) + \epsilon.$$

Epsilon is a small noise parameter used in algorithms requiring stochastic policies.

Below we discuss the implementations of each algorithm.

- **MBOA and BOA.** The GP model used in MBOA and BOA can accept any kernel function (including the BBK). Below we show results comparing these kernels. Unless stated otherwise the squared exponential kernel,

  $$k(\theta_i, \theta_j) = \exp(-\frac{1}{2}\rho(\theta_i - \theta_j)^t(\theta_i - \theta_j)),$$

  is used in our experiments. The width of this kernel is controlled by the scaling parameter ρ. The ρ parameter was tuned for each experiment, and the same value was used in both MBOA and BOA. The prior mean function of BOA is the zero function, and MBOA employs the model-based mean function discussed above.

  It is necessary to optimize the expected improvement for all of the BO algorithms. For this purpose we make use of two simple gradient-free black box optimization algorithms. The DIRECT algorithm detailed by Jones et al. (1993) is used for all tasks except bicycle. DIRECT is poorly suited for problems with more than 15 dimensions. In the bicycle riding domain the policy has 100 dimensions. In this case we use the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm detailed by Hansen (2006). Both DIRECT and CMA-ES require specifying upper and lower

bounds on the policy parameters. We specify an upper bound of 1 and a lower bound of -1 for each dimension of the policy. These bounds hold for all experiments reported below.

To implement the MBOA the designer must define a class of domain models. For the experiments reported below linear models were used. The models are of the form,

$$s_{t_i} = w_i \cdot \phi(s_{t-1}, a_{t-1})^t, r^t = w_r \cdot \phi(s_{t-1}, a_{t-1}, s_t)',$$

where $s_{t,i}$ is the i^{th} state variable at time t, w_i is the weight vector for the i^{th} state variable, and $\phi(s^{t-1}, a^{t-1}, s^t)'$ is a column vector of features computed from the states and actions (and next states in the case of the reward model). The features used in our experiments are found in Table 1. Parameters w_i and w_r are estimated from data using standard linear regression.

- **PILCO.** We use an implementation of PILCO provided by the authors Deisenroth and Rasmussen (2011). This implementation uses sparse GPs to learn the transition and reward functions of the MDP.

- **DYNA-Q.** We make two slight modifications to the DYNA-Q algorithm. First, we provide the algorithm with the same linear models employed by MBOA. These are models of continuous transition functions which DYNA-Q is not normally suited to handle. The problem arises during the sampling of previously visited states during internal reasoning. To perform this sampling we maintain a dictionary of past observations and sample visited states from it. These continuous states are discretized using a CMAC function approximator. The second change we make is to disallow internal reasoning until a trial is completed. To reduce the computational cost reasoning between steps is not allowed. After each trial DYNA-Q is allowed 200000 internal samples to update its Q-function. This was to ensure that during policy selection DYNA-Q was allowed computational resources comparable to the resources used by MBOA and BOA with the BBK.

- **Q-Learning with CMAC function approximation.** ϵ-Greedy exploration is used in the experiments (ϵ is annealed after each step). The discretization of the CMAC approximator was chosen to give robust convergence to good solutions.

- **OLPOMDP** OLPOMDP is a simple gradient based policy search algorithm (Baxter et al., 2001). The OLPOMDP implementations use the same policy space optimized by the BO algorithms.

- **LSPI.** LSPI results are reported in the cart-pole, acrobot, and bicycle tasks. We do not report mountain car or arm results because no tested combinations of basis functions and exploration strategies yielded good performance. We attempted radial basis, polynomial basis, and hybrid basis, but none of these achieved good results. Our exploration strategies used policies returned from the LSPI optimization with added noise (including fully random exploration).

Domain/Features	Transition Function	Reward Function	Policy	Description
Mountain Car	quadratic expansion: $(l, u, cos(l), cos(u), action)$	quadratic expansion: $(l, u, cos(l), cos(u), action)$	cubic expansion: l, u	Variable l denotes the location and u denotes the velocity of the car.
Acrobot	quadratic expansion: $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2, action)$	quadratic expansion: $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2, cos(\theta_1), cos(\theta_2), action)$	$(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$	(θ_1, θ_2) are the angles between the links and $(\dot{\theta}_1, \dot{\theta}_2)$ are the angular velocities.
Cart Pole	$(v, \dot{v}, \omega, \dot{\omega}, action, sin(\omega), cos(\omega), sin(\omega), 1/cos(\omega))$	$(v, \dot{v}, \omega, \dot{\omega}, action, sin(\omega), cos(\omega), sin(\omega), 1/cos(\omega),\ v > 4, v < -4,\ \omega > \frac{\pi}{4},\ \omega < -\frac{\pi}{4})$	$(v, v', \omega, \dot{\omega})$	(v, v') is the velocity and change in velocity of the cart, (ω, ω'), is the angle of the pole, and angular velocity of the pole.
3-link Planar Arm	$(\theta_1, \theta_2, \theta_3, x_t, y_t, action)$	$(\theta_1, \theta_2, \theta_3, (x_t - x_g)^2, (y_t - y_g)^2, action)$	$(x_t - x_g, y_t - y_g)$	$(\theta_1, \theta_2, \theta_3)$ are the link angles, (x_t, y_t) is the location of the arm tip, and (x_g, y_g) is the goal location.
Bicycle	$(\omega, \dot{\omega}, \nu, \dot{\nu}, x_f, x_r, y_f, y_r, action)$	$(\omega, \dot{\omega}, \nu, \dot{\nu}, x_f, x_r, y_f, y_r, action, \lvert\omega\rvert > \frac{\pi}{15}))$	See Lagoudakis et al. (2003) for policy features.	Variables (x_f, x_r, y_f, y_r) represent the locations of the front and rear tires respectively.

Table 1: Features for the transition function, reward function, and policy function.

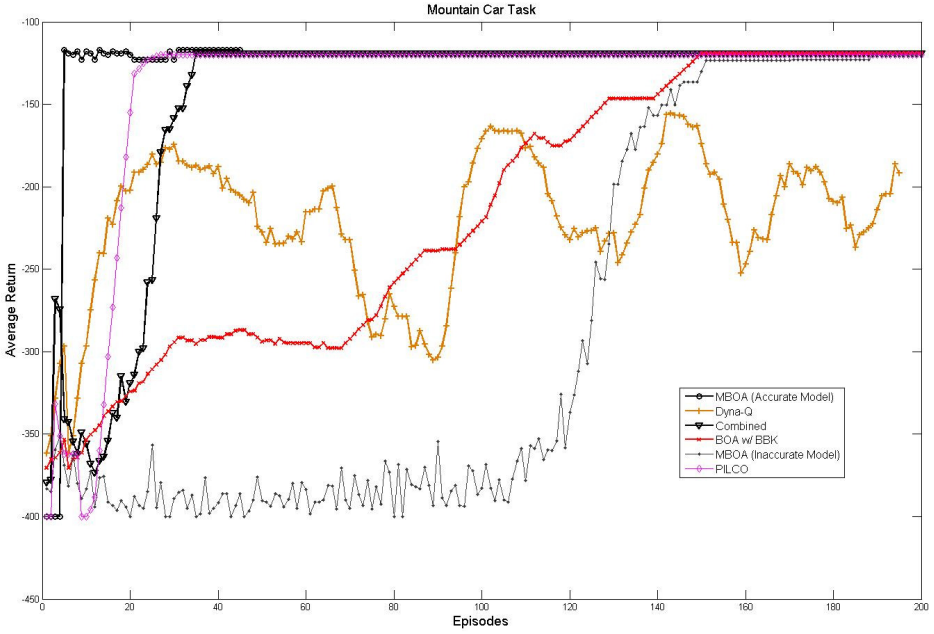


Figure 5: Mountain Car Task: We report the total return per episode averaged over 40 runs of each algorithm.

5.2 Mountain Car Task

In the mountain car domain the goal is to accelerate a car from a fixed position at the base of a hill to the hill's apex. Our implementation of the mountain car task uses eight features derived from the standard state variables (velocity and location of the car). The control policy selects from two actions (applying acceleration forward or to the rear). The policy has sixteen dimensions. The reward function penalizes the agent -1 for each step taken to reach the goal. Agents are allowed a maximum of 400 actions for each episode. Results for the Mountain Car task are shown in Figure 5.

The sparse reward signal makes Mountain Car an ideal experiment for illustrating the importance of directed exploration based on differences in policy behavior. Approaches based on random exploration and exploration weighted by returns are poorly suited to this kind of domain (we have excluded the other model-free methods from the graph because they fail to improve within 200 episodes). Visual inspection of the performance of BOA shows that many of the selected policies, which are unrelated according to the squared exponential kernel, produce similar action sequences when started from the initial state. This redundant search is completely avoided when generalization is controlled by the BBK.

Note that the performance of all kernel methods depends on the settings of the parameters. Optimization of the hyper parameters is not the focus of this work. However, the parameters of the BBK (and any other kernel used with MBOA) can be automatically optimized using any standard method of model selection for GP models (Rasmussen and Williams, 2005). We elect to simply set the parameter of the BBK, α, to 1 for all experiments.

When accurate domain models can be learned, the Monte-Carlo estimates of the expected return computed by the MBOA will accurately reflect the true objective. Therefore, if the domain models can be accurately estimated with few data points MBOA will rapidly identify the optimal policy. To illustrate this we hand-constructed a set of features for the linear domain models used in the mountain car task. Depicted in Figure 5 are the results for MBOA with high quality hand-constructed features. Once the domain models are estimated, only four episodes are needed to yield accurate domain models. Once accurate domain models are available MBOA immediately finds an optimal policy. Typically, the insight used to construct model features is not available in complex domains. To understand the performance of MBOA in settings with poor insight into the correct domain model class, we remove the hand constructed features from the linear models and examine the resulting performance. With poor models, the performance of the MBOA degrades. As shown in the figure, the performance of MBOA is no longer better than the performance of standard BOA with the model-free BBK. The BBK can be combined with the model-based mean function used in MBOA. We show the performance of this combination, using the BBK as the kernel function for the MBOA, labeled Combined, in Figure 5. The results are comparable to the case where MBOA has access to high quality domain models. The performance of PILCO is comparable to the combined algorithm. After 20 episodes the PILCO algorithm has identified a high quality policy for the task. However, PILCO uses all available transition data to train the sparse GP models of the transition and reward function. After 30 episodes the kernel matrices consume all available memory and force the algorithm to page to disk. Due to this problem we terminated the PILCO experiments after 30 episodes, and we report the value of the optimal policy found by PILCO after this point. By contrast, the cost of GP prediction for MBOA grows with the number of episodes, and MBOA exploits a less complex (linear) class of domain models. Consequently, it discovers the optimal policy using less experience and less computational resources.

The performance of MBOA should be contrasted with the performance of DYNA-Q. During DYNA-Q's internal simulations errors in the estimated domain models negatively impact the accuracy of the Q-value estimates. This leads to poor performance in the actual task. In contrast, MBOA is specifically constructed to mitigate the impact of inaccurate domain models. Eventually, even if the domain models have significant errors MBOA can

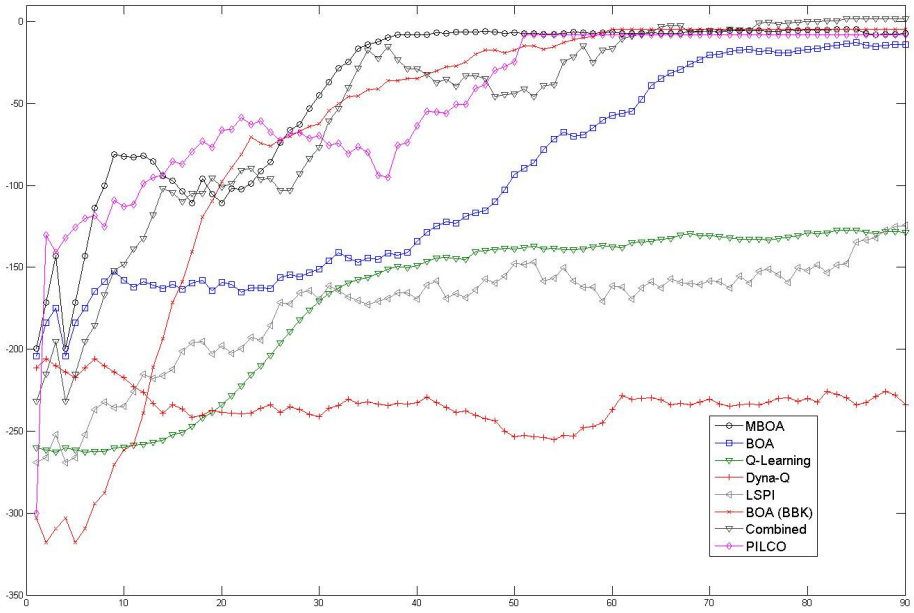


Figure 6: Acrobot Task: We report the total return per episode averaged over 40 runs of each algorithm.

still identify the optimal policy. Standard model-based RL algorithms, such as DYNA-Q, cannot learn optimal policies when confronted with systematic domain model errors.

5.3 Acrobot Task

In the acrobot domain, the goal is to swing the foot of the acrobot over a specified threshold by applying torque at the hip. Details of the implementation can be found in Sutton and Barto (1998). Four features are used for the soft-max policy resulting in twelve policy parameters. The acrobot results are shown in Figure 6.

We were unable to construct accurate linear domain models for this task. Consequently, the DYNA-Q algorithm is unable to find a good policy. MBOA is able to compensate for the domain model errors and exhibits the best performance (some policies are accurately simulated by the poor models). The performance of the BBK is less pronounced in this domain. The policy class generates a varied set of behaviors such that small changes in the policy parameters lead to very different action sequences. Therefore, more behaviors must be searched before good policies are identified. Even so, the behavior based kernel does outperform BOA with squared exponential kernel and it is competitive with the performance of MBOA. In contrast to the mountain car task, combining the BBK with MBOA does not yield improved performance. The performance is comparable to MBOA. The performance of PILCO is similar to the model-based BOAs. However, PILCO suffered from the same problems discussed above. Due to these memory issues we were forced to terminate the

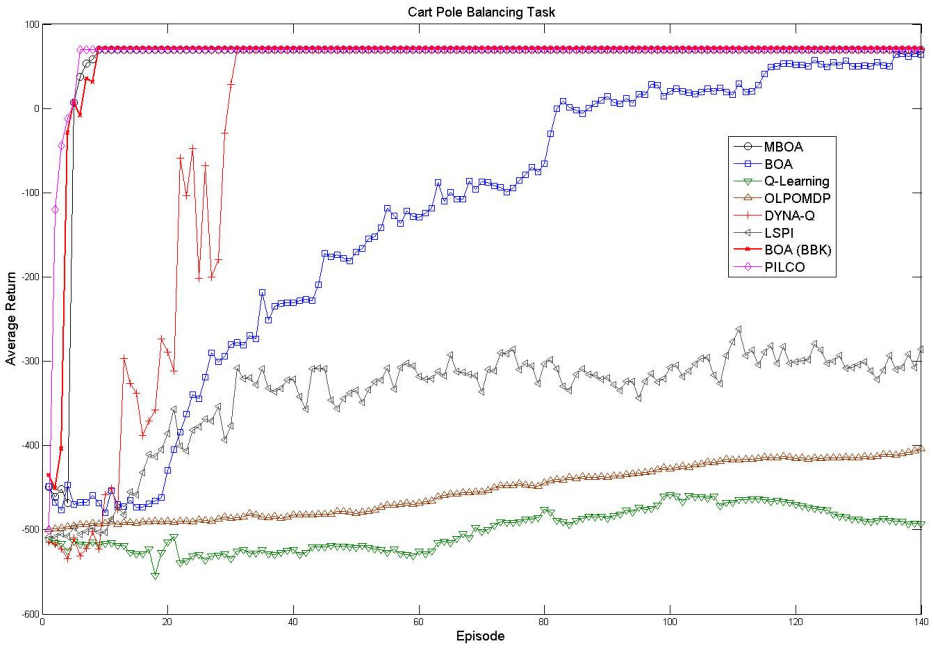


Figure 7: Cart Pole Task: We report the total return per episode averaged over 40 runs of each algorithm.

PILCO results after 50 episodes. Thereafter we report the performance of the best policy found by PILCO.

5.4 Cart-pole Task

In the Cart-pole domain, the agent attempts to balance a pole fixed to a movable cart. The policy selects the magnitude of the change in velocity (positive or negative) applied to the base of the cart. The policy is a function of four features (v, v', ω, ω'), the velocity, change in velocity, angle of the pole, and angular velocity of the pole. The reward function gives a positive reward for each successful step. A penalty is added to this reward when pole angles deviate from vertical, and when the location of the cart deviates from the center. A large positive reward is received after 1000 steps if the pole has been kept upright and the cart has remained within a fixed boundary. Episodes terminate early if the pole falls or the cart leaves its specified boundary.

Figure 7 shows the results for the Cart Pole task. Clearly, the BBK outperforms all of the model-free competitors including BOA with the squared exponential kernel. Its performance is comparable to MBOA despite being fully model-free. In this task, the linear domain models are highly accurate. MBOA effectively employs the models to rapidly identify the optimal policy. By comparison, all other methods, except BOA(BBK), must accumulate more data to identify a good policy. BOA consistently identifies a policy which balances the pole for the full 1000 steps. Q-Learning and OLPOMDP require at least 1250

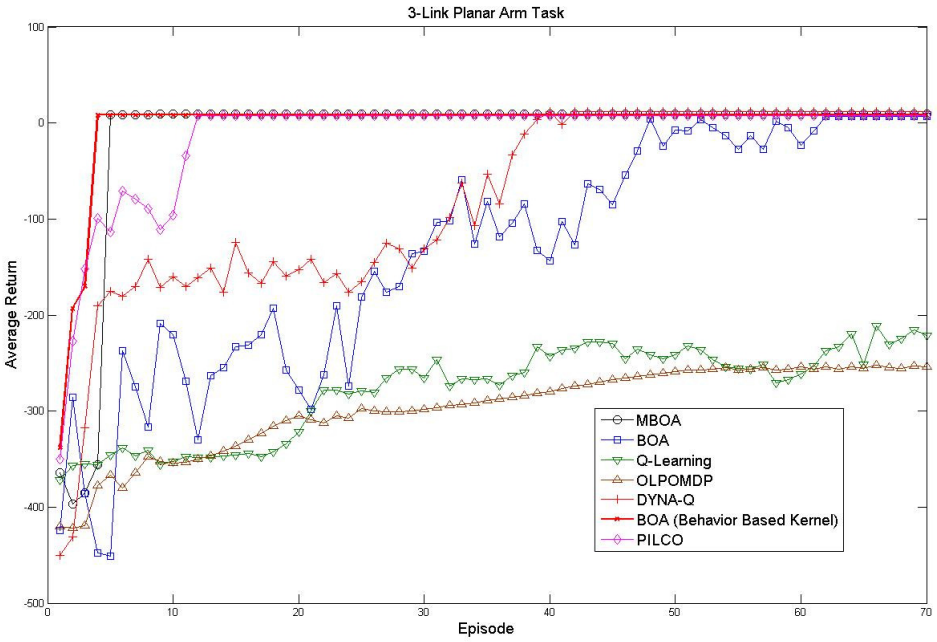


Figure 8: Planar Arm Task: We report the total return per episode averaged over 40 runs of each algorithm.

episodes before similar performance is achieved. LSPI converges faster but cannot match the performance of either the BOA or MBOA. Please note that our problem is distinct from the original LSPI balancing task in the following ways: 1. The force applied by the agent is more restricted. 2. The cart is constrained to move within a fixed boundary. 3. The reward function penalizes unstable policies. After 30 episodes the DYNA-Q algorithm has found a solution comparable to that of MBOA. This performance is unsurprising given the accuracy of the estimated domain models. PILCO learns a high quality model given two episodes of experience and finds an optimal policy by the third episode. In the cart-pole task learning a local model around the balance point is sufficient to perform well and this contributes to PILCO's exceptional performance.

5.5 3-Link Planar Arm Domain

In the 3-link Planar Arm task the goal is to direct the arm tip to a target region on a 2 dimensional plane. Each of the three controllers independently outputs a torque (-1,1) for one arm joint. Each controller is a soft-max action selection policy. Only two features are needed for each controller resulting in 12 total policy parameters (policy parameters are jointly optimized). The features are composed of x and y displacements from the center of the target location to the arm tip location. The reward function penalizes the agent proportional to the distance between the arm tip and the target location. A positive reward is received when the arm tip is placed within the goal space.

The performance of each algorithm is shown in Figure 8. The generalization of the divergence-based kernel, BOA(BBK), is particularly powerful in this case. Much of the policy space is quickly identified to be redundant and is excluded from the search. The agent fixes on an optimal policy almost immediately. Like the Cart-pole task, the transition and reward function of the arm task are linear functions. Once the functions are estimated MBOA quickly identifies the optimal policy (the mean function accurately reflects the true surface). Additional data is needed before the BOA identifies policies with similar quality. The other model-free alternatives require 1000 episodes before converging to the same result. DYNA-Q found a policy comparable to the MBOA, but required more experience. In the three tasks discussed above the MBOA makes better use of the available computational resources and remains robust to model bias. PILCO very quickly finds an optimal policy in this task. After ten episodes PILCO has learned an accurate GP model of the transition dynamics. Thereafter, PILCO finds a near optimal policy. We have extrapolated PILCO's performance beyond the 20th episode. Each PILCO run required 3 weeks to generate the first 20 episodes.

5.6 Bicycle Balancing

Agents in the bicycle balancing task must keep the bicycle balanced for ten minutes of simulated time. For our experiments we use the simulator originally introduced in Randlov and Alstrom (1998). The state of the bicycle is defined by four variables $(\omega, \dot{\omega}, \nu, \dot{\nu})$. The variable ω is the angle of the bicycle with respect to vertical, and $\dot{\omega}$ is its angular velocity. The variable ν is the angle of the handlebars with respect to neutral, and $\dot{\nu}$ is the angular velocity. The goal of the agent is to keep the bicycle from falling. Falling occurs when $|\omega| > \pi/15$. The same discrete action set used in Lagoudakis et al. (2003) is used in our implementations. Actions have two components. The first component is the torque applied to the handlebars $T \in (-1, 0, 1)$, and the second component is the displacement of the rider in the saddle $p \in (-.02, 0, .02)$. Five actions are composed from these values $(T, p) \in ((-1, 0), (1, 0), (0, -.02), (0, .02), (0, 0))$. The reward function is defined to be the squared change in ω, $(\omega_t - \omega_{t+1})^2$, at each time step. An additional -10 penalty is imposed if the bicycle falls before time expires. Rewards are discounted in time insuring that longer runs result in smaller penalties. In our implementation the set of 20 features introduced in Lagoudakis et al. (2003) were used. The softmax policy has 100 parameters. Please note that our LSPI implementation of bicycle does not benefit from the design decisions introduced in Lagoudakis et al. (2003).

In Figure 9 we report the performance of MBOA and LSPI. All other implementations, including MBOA combined with the BBK, show no improvement in 300 episodes. In this case the BBK performs poorly. It overestimates the distance between policies with dissimilar action selection distributions. In the bicycle riding task individual sequences of oscillating actions can produce similar state trajectories. A policy executing right,left,right,left,.... behaves much like a policy executing left,right,left,right,... even though the action selection distributions may be arbitrarily dissimilar. This is a problem with the BBK that must be addressed in future work. By contrast, after 300 episodes MBOA has found a balancing policy. For this experiment we used a linear kernel function with automatic relevance determination. The parameters of the linear kernel function were optimized by maximum

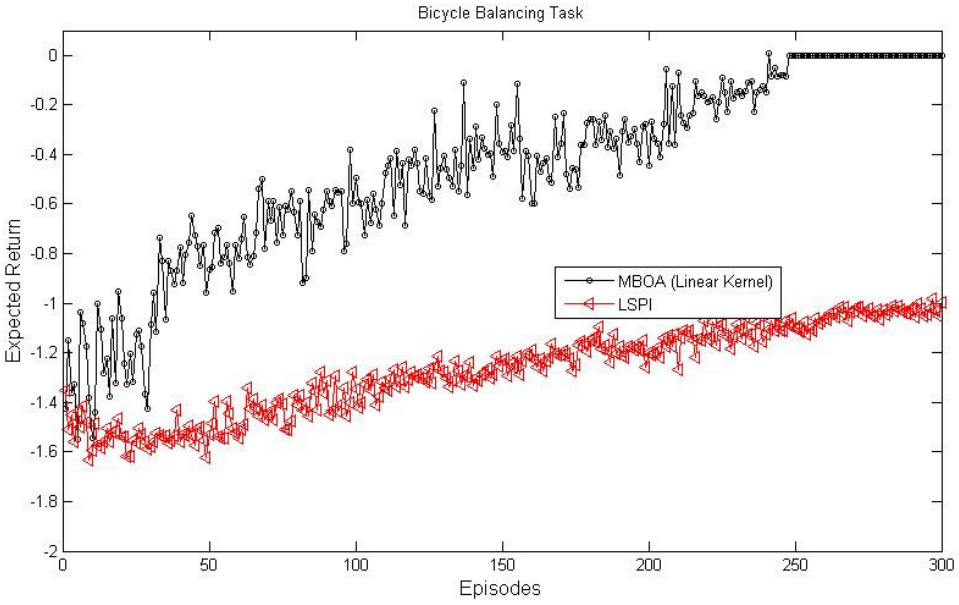


Figure 9: Bicycle Balancing Task: We report the total return per episode averaged over multiple runs of each algorithm (15 runs of MBOA and 30 runs of LSPI).

likelihood after each policy execution. Before optimization of the hyper-parameters the algorithm was made to choose 20 policies with the uninformed values. MBOA uses very simple and highly inaccurate linear models in this task. Errors in the linear models are significant enough to cause DYNA-Q to fail. MBOA exhibits robust performance despite the predictive errors caused by model bias. It converges to a near optimal policy and exhibits excellent data efficiency.

We allowed the PILCO algorithm to run for three weeks. After this period each run had completed only 24 episodes. This was due to the size of the kernel covariance matrices. Taken together these matrices consumed several gigabytes of memory and forced the algorithm to access the hard disk. This is a serious problem with using (sparse) GP models to represent the transition and reward function in MDPs. Unfortunately, the PILCO algorithm had not found a high quality policy given the available experience. We elected to exclude these results from the depicted experiments.

6. Related Work

By extending the work on BO we place our research squarely within the growing body of literature on Bayesian RL. Most closely related is extensive work adapting BO methods to the RL problem (Lizotte et al., 2007; Lizotte, 2008). Like our work, these authors propose modeling the surface of the expected return, that maps policy parameters to returns, with a GP model. We have extended this work by designing new GP models of the expected return that leverage trajectory data and we have empirically demonstrated the performance benefits of these new models.

Gaussian processes have been used to model functions of interest in the RL setting. Related work employing Gaussian processes in RL includes Engel et al. (2003, 2005) in which the value function is modeled as a Gaussian Process, Ghavamzadeh and Engel (2007a,b) wherein GPs were used to model the gradient of the expected return, and the work of Deisenroth and Rasmussen (2011) and Rasmussen and Kuss (2004) where GPs were used to model the transition and reward functions.

Work by Engel et al. (2003, 2005) focuses on the problem of estimating the value function given a fixed action selection policy. GP models are used to approximate the value function; The GP model introduces a smoothness prior on the space of value functions. Similarly, Ghavamzadeh and Engel (2007a,b) focus on the problem of estimating the gradient of the value function for a fixed action selection policy. It should be noted that the kernel described by Ghavamzadeh and Engel (2007b) exploit trajectory information; the authors propose a Fisher kernel for relating sampled trajectories. This facilitates generalization across sampled trajectories and leads to improved estimates of the gradient of the value function. The results of their experiments show consistent improvement, in a set of benchmark problems, over standard Monte-Carlo methods for estimating the gradient of the expected return. However, their kernel cannot be used to compare *policies* as is done in our work. During the process of policy improvement their approach re-samples trajectory data for each new policy. In contrast, the GP models discussed in this article perform off-policy estimation of the expected return and re-use sampled trajectory data to approximate the returns of new policies.

Numerous model-based Bayesian RL algorithms exploit Bayesian priors on the domain model parameters (Dearden et al., 1999; Strens, 2000; Duff, 2003; Deisenroth and Rasmussen, 2011; Rasmussen and Kuss, 2004). The priors encode the agents uncertainty of the world it lives within. Like our algorithm, these methods actively explore to reduce uncertainty. For instance, work by Dearden et al. (1999) introduced a method for representing Bayesian belief states and proposed an action selection policy based on Value of Information (this policy was first introduced in Dearden et al. 1998). Related work by Strens (2000) and Duff (2003) introduced new approaches for approximately solving the Bayesian belief state MDP. An approximate solution of the belief state MDP can be used to derive an action selection policy that approximates Bayes optimal action selection. Our work explores the problem of Bayesian parametric policy search and exploits a heuristic for directed exploration of the parametric policy space. However, we avoid Bayesian modeling of the domain models. Instead, our approach directly represents uncertainty of the expected return. Our empirical results demonstrate how this impacts practical application.

To understand the consequences of this decision consider the recent work by Deisenroth and Rasmussen (2011) whom developed the PILCO algorithm. PILCO learns GP models of the transition and reward functions. The learned transition and reward functions are used to approximate the expected return via prediction of state-reward sequences. The policy is optimized by analytic gradient ascent on the approximated expected return. To represent a GP model of the transition and reward function it is necessary to represent a matrix with T^2 elements (where T is the total number of observed state transitions). This is problematic in the RL setting where agents may take many thousands of steps during their lifetimes. Our approach to BO uses GPs to model the mapping from policies to expected returns. Using GP models to directly model the expected return prevents the kernel matrices from

exploding in size. For instance, in the experiments above our kernel matrices include at most 300^2 elements whereas PILCO's kernel matrices quickly explode to more than 40000^2 elements. Sparse GP inference can help to reduce the costs of dealing with these large matrices. However, as is demonstrated in our experiments we found that sparse inference was not sufficient to overcome the performance problems. The advantage of the approach can be further improved by introducing heuristic local search techniques for discovering the maximum of the expected improvement function.

Work by Kakade (2001) presented a metric based on the Fisher information to derive the natural policy gradient update. Kakade (2001) demonstrated that natural policy gradient methods outperform standard gradient methods in a difficult Tetris domain. Follow up work by Bagnell and Schneider (2003) proposed pursuing a related idea within the path integral framework for RL (the same framework of this paper). Their work considered metrics defined as functions on the distribution over trajectories generated by a fixed policy $P(\xi|\theta)$. In contrast to our goals both works focus on iteratively improving a policy via gradient descent. Furthermore, no explicit attention is paid to using the metric information to guide the exploratory process. However, the insight that policy relationships should be expressed as functions of the trajectory density has played a key role in the development of our behavior based kernel.

Work by Peters et al. (2010) and Kober and Peters (2010) is related to our proposed kernel function. Peters et al. (2010) used a divergence-based bound to control exploration. Specifically, they attempt to maximize the expected reward subject to a bound proportional to the KL-divergence between the empirically observed state-action distribution and the state-action distribution of the new policy. The search for a new policy is necessarily local, restricted by the bound to be close to the current policy. By contrast, our work uses the divergence as a measure of similarity and performs a global, aggressive, search of the policy space. Work by Kober and Peters (2010) derives a lower bound on the importance sampled estimate of the expected return, as was done in Dayan and Hinton (1997), and observes the relationship to the KL-divergence of the reward weighted behavior policy and the target policy. They derive from this relationship an EM-based update for the policy parameters. An explicit effort is made to construct the update such that exploration is accounted for. However, their method of state-dependent exploration is based on random perturbations of the action selection policy. Our method of exploration is instead determined by the posterior uncertainty and does not depend on a behavior policy.

The BBK leverages a generative model of the trajectory distribution (estimated from a finite set of samples) to compute a measure of policy similarity that is robust to policy reformulation (it only depends on the allocation of probability mass). By contrast, most work on kernel methods can be considered model-free in that they do not make distributional assumptions regarding the data generating process. Recent work has explored the advantages of incorporating generative assumptions into the kernel function (Lafferty and Lebanon, 2005; Belkin and Niyogi, 2002; Moreno et al., 2004). In particular, our work is closely related to work by Moreno et al. (2004) which introduces a kernel based on the symmetric KL-divergence. Our BBK is a modification of this kernel for the purpose of GP modeling of the expected return. In addition, we describe a method of estimating the kernel values given off-policy trajectory samples whereas previous work typically assumes access to a generative model.

7. Conclusions

General black box Bayesian optimization algorithms have been adapted to the RL policy search problem, and have shown promise in difficult domains. We show how to improve these algorithms by exploiting trajectory data generated by agents.

For model-free RL, we presented a new kernel function for GP models of the expected return. The kernel compares sequences of action selection decisions from pairs of policies. We motivated our kernel by examining a simple upper bound on the absolute difference of expected returns for two arbitrary policies. We used this upper bound as the basis for our kernel function, argued that the properties of the bound ensure a more reasonable measure of policy relatedness, and demonstrated empirically that this kernel can substantially speed up exploration. The empirical results show that the derived kernel is competitive with the model-based RL algorithms MBOA and Dyna-Q, and converge more quickly than the model-free algorithms tested. Additionally, we showed that in certain circumstances BBK can be combined with the MBOA to improve the quality of exploration.

For model-based RL we presented MBOA for model-based RL. MBOA improves on the standard BOA algorithm by using collections of trajectories to learn an approximate simulator of the decision process. The simulator is used to define an approximation of the underlying objective function. Potentially, such a function provides useful information about the performance of unseen policies which are distant (according to a measure of similarity) from previous data points. Empirically, we show that the MBOA has exceptional data efficiency. Its performance exceeds LSPI, OLPOMDP, Q-Learning with CMAC function approximation, BOA, and DYNA-Q in all but one of our tasks. MBOA performs well even when the learned domain models have serious errors. Overall, MBOA appears to be a useful step toward combining model-based methods with Bayesian optimization for the purposes of handling approximate models and improving data efficiency.

The penalty for taking this approach is straightforward. A substantial increase in computational cost is incurred during optimization of the surrogate function. This occurs when evaluating MBOA's approximate simulator, or when estimating the values of the behavior based kernel. In both cases the cost of evaluating the kernel function grows linearly in the length of the trajectories. However, the cost is compounded during optimization and will make these approaches infeasible when trajectory lengths become huge. Overcoming this barrier is an important step towards taking advantage of trajectory data generated in complex domains.

Acknowledgments

We gratefully acknowledge the support of the Office of Naval Research under grant number N00014-11-1-0106, the Army Research Office under grant number W911NF-09-1-0153, and the National Science Foundation under grant number IIS-0905678.

References

J. Andrew Bagnell and Jeff Schneider. Covariant policy search. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1019–1024,

San Francisco, CA, USA, 2003.

J. Baxter, P. Bartlett, and L. Weaver. Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15(1):351–381, 2001.

M. Belkin and P. Niyogi. Using manifold structure for partially labelled classification. In *Advances in Neural Information Processing Systems*, pages 929–937, Vancouver, B.C., CA, 2002.

E. Brochu, V. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *Technical Report TR-2009-023*, 2009.

P. Dayan and G. Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9:271–278, 1997.

R. Dearden, N. Friedman, and S. Russell. Bayesian q-learning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 761–768, 1998.

R. Dearden, N. Friedman, and D. Andre. Model-based bayesian exploration. In *Uncertainty in Artificial Intelligence*, pages 150–159, Stockholm, Sweden, 1999.

M. Deisenroth and C. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the Twenty-Eigth International Conference on Machine Learning*, pages 465–472, Bellevue, Washington, USA, 2011.

M. Duff. Design for an optimal probe. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 131–138, Washington, DC, USA, 2003.

Y. Engel, S. Mannor, and R. Meir. Bayes meets bellman:the Gaussian process approach to temporal difference learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 154–161, Washington, DC, USA, 2003.

Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, pages 201–208, University of Bonn, Germany, 2005.

M. Ghavamzadeh and Y. Engel. Bayesian actor-critic algorithms. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, pages 297–304, Oregon State University, Corvallis, USA, 2007a.

M. Ghavamzadeh and Y. Engel. Bayesian policy gradient algorithms. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, pages 457–464, Vancouver, B.C., CA, 2007b.

N. Hansen. The CMA evolution strategy: A comparing review. In J.A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms*, pages 75–102. Springer, 2006.

D. Jones, C. Perttunen, and B. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

S. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, pages 1531–1538, Vancouver, B.C., CA, 2001.

J. Kober and J. Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84(1-2):1–33, 2010.

J. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6:129–163, 2005.

M. Lagoudakis, R. Parr, and L. Bartlett. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

D. Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, Edmonton, Canada, 2008.

D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans. Automatic gait optimization with Gaussian process regression. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 944–949, Hyderabad, India, 2007.

J. Mockus. Application of Bayesian approach to numerical methods of global and stochastic optimization. *Global Optimization*, 4(4):347–365, 1994.

P. Moreno, P. Ho, and N. Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In *Advances in Neural Information Processing Systems*, Vancouver, B.C., CA, 2004.

X. Nguyen, M. Wainwright, and M. Jordan. Estimating divergence functionals and the likelihood ratio by penalized convex risk minimization. In *Advances in Neural Information Processing Systems*, pages 1089–1096, Vancouver, B.C., CA, 2007.

J. Peters, K. Mülling, and Y. Altün. Relative entropy policy search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 1607–1612, Atlanta, Georgia, USA, 2010.

M. Pinsker. *Information and Information Stability of Random Variables and Processes*. Holden-Day Inc, San Francisco, CA, USA, 1964. Translated by Amiel Feinstein.

J. Randlov and P. Alstrom. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 463–471, San Francisco, CA, 1998.

C. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 751–759, Vancouver, B.C., CA, 2004.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, Cambridge, MA, USA, 2005.

M. J. A. Strens. A Bayesian framework for reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 943–950, Stanford, CA, USA, 2000.

R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.

Journal of Machine Learning Research 15 (2014) 283-287 Submitted 10/12; Revised 9/13; Published 1/14

Information Theoretical Estimators Toolbox

Zoltán Szabó* ZOLTAN.SZABO@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit
Centre for Computational Statistics and Machine Learning
University College London
Alexandra House, 17 Queen Square, London - WC1N 3AR

Editor: Balázs Kégl

Abstract

We present ITE (information theoretical estimators) a free and open source, multi-platform, Matlab/Octave toolbox that is capable of estimating many different variants of entropy, mutual information, divergence, association measures, cross quantities, and kernels on distributions. Thanks to its highly modular design, ITE supports additionally (i) the combinations of the estimation techniques, (ii) the easy construction and embedding of novel information theoretical estimators, and (iii) their immediate application in information theoretical optimization problems. ITE also includes a prototype application in a central problem class of signal processing, independent subspace analysis and its extensions.

Keywords: entropy-, mutual information-, association-, divergence-, distribution kernel estimation, independent subspace analysis and its extensions, modularity, Matlab/Octave, multi-platform, GNU GPLv3 ($\geq$)

1. Introduction

Since the pioneering work of Shannon (1948), *entropy*, *mutual information*,[1] *association*, *divergence* measures and *kernels on distributions* have found a broad range of applications in many areas of machine learning (Beirlant et al., 1997; Wang et al., 2009; Villmann and Haase, 2010; Basseville, 2013; Póczos et al., 2012; Sriperumbudur et al., 2012). Entropies provide a natural notion to quantify the *uncertainty* of random variables, mutual information and association indices measure the *dependence* among its arguments, divergences and kernels offer efficient tools to define the 'distance' and the inner product of probability measures, respectively.

A central problem based on information theoretical objectives in signal processing is independent subspace analysis (ISA; Cardoso 1998), a cocktail party problem with *independent groups*. One of the most relevant and fundamental hypotheses of the ISA research

∗. The authors would like to thank the anonymous reviewers for their valuable suggestions. This work was supported by the Gatsby Charitable Foundation. The research was carried out as part of the EITKIC_12-1-2012-0001 project, which is supported by the Hungarian Government, managed by the National Development Agency, financed by the Research and Technology Innovation Fund and was performed in cooperation with the EIT ICT Labs Budapest Associate Partner Group. (`www.ictlabs.elte.hu`). The Project was supported by the European Union and co-financed by the European Social Fund (grant agreement no. TÁMOP 4.2.1/B-09/1/KMR-2010-0003).

1. The Shannon mutual information is also known in the literature as the special case of total correlation or multi-information when two variables are considered.

is the *ISA separation principle* (Cardoso, 1998): the ISA task can be solved by ICA (ISA with one-dimensional sources, Hyvärinen et al. 2001; Cichocki and Amari 2002; Choi et al. 2005) followed by clustering of the ICA elements. This principle (i) forms the basis of the state-of-the-art ISA algorithms, (ii) can be used to design algorithms that scale well and efficiently estimate the dimensions of the hidden sources, (iii) has been recently proved (Szabó et al., 2007) and (iv) can be extended to different linear-, controlled-, post nonlinear-, complex valued-, partially observed systems, as well as to systems with nonparametric source dynamics. For a recent review on the topic and ISA applications, see Szabó et al. (2012).

Although there exist many exciting applications of information theoretical measures, to the best of our knowledge, available packages in this domain focus on (i) discrete variables, or (ii) quite specialized applications/information theoretical estimation methods. Our **goal** is to fill this serious gap by coming up with a (i) highly modular, (ii) free and open source, (iii) multi-platform toolbox, the ITE (information theoretical estimators) package, which focuses on *continuous* variables and

1. is capable of estimating *many* different kind of entropy, mutual information, association, divergence measures, distribution kernels based on nonparametric methods.[2]
2. offers a *simple and unified framework* to (i) easily construct new estimators from existing ones or from scratch, and (ii) transparently use the obtained estimators in information theoretical optimization problems,
3. with a *prototype application* in ISA and its extensions.

2. Library Overview

Below we provide a brief overview of the ITE package:

Information Theoretical Measures: The ITE toolbox is capable of estimating numerous important information theoretical quantities including

Entropy: Shannon-, Rényi-, Tsallis-, complex-, Φ-, Sharma-Mittal entropy,

Mutual information: generalized variance, kernel canonical correlation analysis, kernel generalized variance, Hilbert-Schmidt independence criterion, Shannon-, L_2-, Rényi-, Tsallis-, Cauchy-Schwartz quadratic-, Euclidean distance based quadratic-, complex-, χ^2 mutual information; copula-based kernel dependency, multivariate version of Hoeffding's Φ, Schweizer-Wolff's σ and κ, distance covariance and correlation, approximate correntropy independence measure,

Divergence: Kullback-Leibler-, L_2-, Rényi-, Tsallis-, Cauchy-Schwartz-, Euclidean distance based-, Jensen-Shannon-, Jensen-Rényi-, Jensen-Tsallis-, K-, L-, Pearson χ^2-, f-divergences; Hellinger-, Bhattacharyya-, energy-, (non-)symmetric Bregman-, J-distance; maximum mean discrepancy,

Association measure: multivariate (conditional) extensions of Spearman's ρ, (centered) correntropy, correntropy induced metric, correntropy coefficient, centered correntropy induced metric, multivariate extension of Blomqvist's β, lower and upper tail dependence via conditional Spearman's ρ,

Cross quantity: cross-entropy,

2. It is highly advantageous to apply nonparametric approaches: the 'opposite' plug-in type methods—estimating the underlying densities—scale poorly as the dimension is increasing.

Distribution kernel: expected-, Bhattacharyya-, probability product-, (exponentiated) Jensen-Shannon-, (exponentiated) Jensen-Tsallis-, exponentiated Jensen-Rényi kernel.

Independent Process Analysis (IPA): ITE offers solution methods for independent subspace analysis (ISA) and its extensions to different linear-, controlled-, post nonlinear-, complex valued-, partially observed systems, as well as to systems with nonparametric source dynamics; combinations are also possible. The solutions are based on the ISA separation principle and its generalizations (Szabó et al., 2012).

Quick Tests: Beyond IPA, ITE provides quick tests to study the efficiency of the estimators. These tests cover (i) analytical value vs. estimation, (ii) positive semi-definiteness of Gram matrices defined by distribution kernels and (iii) image registration problems.

Modularity: The core idea behind the design of ITE is modularity. The modularity is based on the following four pillars:

1. The estimation of many information theoretical quantities can be reduced to k-nearest neighbor-, minimum spanning tree computation, random projection, ensemble technique, copula estimation, kernel methods.
2. The ISA separation principle and its extensions make it possible to decompose the solutions of the IPA problem family to ICA, clustering, ISA, AR (autoregressive)-, ARX- (AR with exogenous input) and mAR (AR with missing values) identification, gaussianization and nonparametric regression subtasks.
3. Information theoretical identities can relate numerous entropy, mutual information, association, cross- and divergence measures, distribution kernels (Cover and Thomas, 1991).
4. ISA can be formulated via information theoretical objectives (Szabó et al., 2007):

$$J_{\mathrm{I}}(\mathbf{P}) = I\left(\mathbf{y}^1,\ldots,\mathbf{y}^M\right), \qquad J_{\mathrm{Irecursive}}(\mathbf{P}) = \sum_{m=1}^{M-1} I\left(\mathbf{y}^m,\left[\mathbf{y}^{m+1},...,\mathbf{y}^M\right]\right),$$

$$J_{\mathrm{sumH}}(\mathbf{P}) = \sum_{m=1}^{M} H\left(\mathbf{y}^m\right), \qquad J_{\text{sum-I}}(\mathbf{P}) = -\sum_{m=1}^{M} I\left(y_1^m,...,y_{d_m}^M\right),$$

$$J_{\mathrm{Ipairwise}}(\mathbf{P}) = \sum_{m_1\neq m_2} I\left(\mathbf{y}^{m_1},\mathbf{y}^{m_2}\right), \; J_{\mathrm{Ipairwise1d}}(\mathbf{P}) = \sum_{m_1\neq m_2}^{M}\sum_{i_1=1}^{d_{m_1}}\sum_{i_2=1}^{d_{m_2}} I\left(y_{i_1}^{m_1},y_{i_2}^{m_2}\right),$$

where the minimizations are w.r.t. the optimal clustering ($\mathbf{P}$) of the ICA elements.

Dedicated Subtask Solvers, Extension: The ITE package offers dedicated solvers for the obtained subproblems detailed in '*Modularity*:1-2'. Thanks to this flexibility, extension of ITE can be done effortlessly: it is sufficient to add a new switch entry in the subtask solver.

Base and Meta Estimators: One can *derive* new, *meta* (entropy, mutual information, association, divergence, cross quantity, distribution kernel) estimators in ITE from existing base or meta ones by '*Modularity*:3'. The calling syntax of base and meta methods are completely identical thanks to the underlying unified template structure

followed by the estimators. The ITE package also supports an indicator for the importance of multiplicative constants.[3]

We illustrate how easily one can estimate information theoretical quantities in ITE:

```
>Y1 = rand(3,1000); Y2 = rand(3,2000);   %data of interest
>mult = 1;                    %multiplicative constant is important
>co = D_initialization('Jdistance',mult);%initialize the estimator
>D = D_estimation(Y1,Y2,co);              %estimation
```

Next, we demonstrate how one can construct meta estimators in ITE. We consider the definitions of the initialization and the estimation of the J-distance. The KL-divergence, which is symmetrised in J-distance, is estimated based on the existing k-nearest neighbor technique.

```
function [co] = DJdistance_initialization(mult)
co.name = 'Jdistance';           %name of the estimator
co.mult = mult;                  %importance of multiplicative const.
co.member_name = 'KL_kNN_k';     %method used for KL estimation
co.member_co = D_initialization(co.member_name,mult); %initialization

function [D_J] = DJdistance_estimation(X,Y,co)
D_J = D_estimation(X,Y,co.member_co) + D_estimation(Y,X,co.member_co);
```

ISA Objectives and Optimization: Due to the unified syntax of the estimators, one can formulate and solve information theoretical optimization problems in ITE in a high-level view. Our example included in ITE is ISA (and its extensions) whose objective can be expressed by entropy and mutual information terms, see '*Modularity*:4'. The unified template structure in ITE makes it possible to use *any* of the estimators (base/meta) in these cost functions.

A further attractive aspect of ITE is that even in case of unknown subspace dimensions, it offers well-scaling approximation schemes based on spectral clustering methods. Such methods are (i) robust and (ii) scale excellently, a single general desktop computer can handle about a million observations—in our case estimated ICA elements—within several minutes (Yan et al., 2009).

3. Availability and Requirements

The ITE package is *self-contained*, it only needs a Matlab or an Octave environment[4] with standard toolboxes. ITE is *multi-platform*, it has been extensively tested on Windows and Linux; since it is made of standard Matlab/Octave and C++ files, it is expected to work on alternative platforms as well.[5] ITE is released under the free and open source GNU GPLv3 ($\geq$) license. The accompanying source code and the documentation of the toolbox has been enriched with numerous comments, examples, detailed instructions for extensions, and pointers where the interested user can find further mathematical details about the embodied techniques. The ITE package is available at `https://bitbucket.org/szzoli/ite/`.

3. In many applications, it is completely irrelevant whether we estimate, for example, $H(\mathbf{y})$ or $cH(\mathbf{y})$, where $c = c(d)$ is a constant depending only on the *dimension* of $\mathbf{y} \in \mathbb{R}^d$ (d), but *not on the distribution* of $\mathbf{y}$. Such 'up to proportional factor' estimations can often be carried out more efficiently.
4. See `http://www.mathworks.com/products/matlab/` and `http://www.gnu.org/software/octave/`.
5. On Windows (Linux) we suggest using the Visual C++ (GCC) compiler.

References

Michéle Basseville. Divergence measures for statistical data processing - an annotated bibliography. *Signal Processing*, 93:621–633, 2013.

Jan Beirlant, Edward J. Dudewicz, László Győrfi, and Edward C. van der Meulen. Nonparametric entropy estimation: An overview. *International Journal of Mathematical and Statistical Sciences*, 6:17–39, 1997.

Jean-François Cardoso. Multidimensional independent component analysis. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 1941–1944, 1998.

Seungjin Choi, Andrzej Cichocki, Hyung-Min Park, and Soo-Yound Lee. Blind source separation and independent component analysis. *Neural Information Processing - Letters and Reviews*, 6:1–57, 2005.

Andrzej Cichocki and Shun-ichi Amari. *Adaptive Blind Signal and Image Processing*. John Wiley & Sons, 2002.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley and Sons, New York, USA, 1991.

Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.

Barnabás Póczos, Zoubin Ghahramani, and Jeff Schneider. Copula-based kernel dependency measures. In *International Conference on Machine Learning*, pages 775–782, 2012.

Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert R. G. Lanckriet. On the empirical estimation of integral probability metrics. *Electronic Journal of Statistics*, 6:1550–1599, 2012.

Zoltán Szabó, Barnabás Póczos, and András Lőrincz. Undercomplete blind subspace deconvolution. *Journal of Machine Learning Research*, 8:1063–1095, 2007.

Zoltán Szabó, Barnabás Póczos, and András Lőrincz. Separation theorem for independent subspace analysis and its consequences. *Pattern Recognition*, 45:1782–1791, 2012.

Thomas Villmann and Sven Haase. Mathematical aspects of divergence based vector quantization using Fréchet-derivatives. Technical report, University of Applied Sciences Mittweida, 2010.

Quing Wang, Sanjeev R. Kulkarni, and Sergio Verdú. Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Transactions on Information Theory*, 55:2392–2405, 2009.

Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In *International Conference on Knowledge Discovery and Data Mining*, pages 907–916, 2009.

Journal of Machine Learning Research 15 (2014) 289-333 Submitted 9/11; Revised 4/13; Published 1/14

Off-policy Learning With Eligibility Traces: A Survey

Matthieu Geist MATTHIEU.GEIST@SUPELEC.FR
IMS-MaLIS Research Group & UMI 2958 (GeorgiaTech-CNRS)
Supélec
2 rue Edouard Belin
57070 Metz, France

Bruno Scherrer BRUNO.SCHERRER@INRIA.FR
MAIA project-team
INRIA Lorraine
615 rue du Jardin Botanique
54600 Villers-lès-Nancy, France

Editor: Ronald Parr

Abstract

In the framework of Markov Decision Processes, we consider linear *off-policy* learning, that is the problem of learning a linear approximation of the value function of some fixed policy from one trajectory possibly generated by some other policy. We briefly review *on-policy* learning algorithms of the literature (gradient-based and least-squares-based), adopting a unified algorithmic view. Then, we highlight a systematic approach for adapting them to *off-policy* learning *with eligibility traces*. This leads to some known algorithms—off-policy LSTD(λ), LSPE(λ), TD(λ), TDC/GQ(λ)—and suggests new extensions—off-policy FPKF(λ), BRM(λ), gBRM(λ), GTD2(λ). We describe a comprehensive algorithmic derivation of all algorithms in a recursive and memory-efficent form, discuss their known convergence properties and illustrate their relative empirical behavior on Garnet problems. Our experiments suggest that the most standard algorithms on and off-policy LSTD(λ)/LSPE(λ)—and TD(λ) if the feature space dimension is too large for a least-squares approach—perform the best.

Keywords: reinforcement learning, value function estimation, off-policy learning, eligibility traces

1. Introduction

We consider the problem of learning a linear approximation of the value function of some fixed policy in a Markov Decision Process (MDP) framework. This study is performed in the most general situation where learning must be done from a single trajectory possibly generated by some other policy, also known as *off-policy* learning. Given samples, well-known methods for estimating a value function are temporal difference (TD) learning and Monte Carlo (Sutton and Barto, 1998). TD learning with eligibility traces (Sutton and Barto, 1998), known as TD(λ), constitutes a nice bridge between both approaches; by controlling the bias/variance trade-off (Kearns and Singh, 2000), their use can significantly speed up learning. When the value function is approximated through a linear architecture, the depth

λ of the eligibility traces is also known to control the quality of approximation (Tsitsiklis and Van Roy, 1997). Overall, the use of these traces often plays an important practical role.

	gradient-based	least-squares-based
bootstrapping	TD (Sutton and Barto, 1998)	FPKF (Choi and Van Roy, 2006)
residual	gBRM (Baird, 1995)	BRM (Engel, 2005) (Geist and Pietquin, 2010b)
projected fixed point	TDC/GTD2 (Sutton et al., 2009)	LSTD (Bradtke and Barto, 1996) LSPE (Nedić and Bertsekas, 2003)

Table 1: Taxonomy of linearly parameterized estimators for value function approximation (Geist and Pietquin, 2013).

There has been a significant amount of research on parametric linear approximation of the value function, *without eligibility traces* (in the on- or off-policy case). We follow the taxonomy proposed by Geist and Pietquin (2013), briefly recalled in Table 1 and further developed in Section 2. Value function approximators can be categorized depending on the cost function they minimize (based on bootstrapping, on a Bellman residual minimization or on a projected fixed point approach) and on how it is minimized (gradient descent or linear least-squares). Most of these algorithms have been extended to take into account eligibility traces, in the on-policy case. Works on extending these eligibility-trace approaches to off-policy learning are scarcer. They are summarized in Table 2 (algorithms in black). The first motivation of this article is to argue that it is conceptually simple to extend *all* the algorithms of Table 1 so that they can be applied to the off-policy setting *and* use eligibility traces. If this allows re-deriving existing algorithms (in black in Table 2), it also leads to new candidate algorithms (in red in Table 2). The second motivation of this work is to discuss the subtle differences between these intimately-related algorithms, and to provide some comparative insights on their empirical behavior (a topic that has to our knowledge not been considered in the literature, even in the simplest *on-policy* and *no-trace* situation).

	gradient-based	least-squares-based
bootstrapping	off-policy TD(λ) (Bertsekas and Yu, 2009)	off-policy FPKF(λ)
residual	off-policy gBRM(λ)	off-policy BRM(λ)
projected fixed point	GQ(λ) a.k.a. off-policy TDC(λ) (Maei and Sutton, 2010) off-policy GTD2(λ)	off-policy LSTD(λ) off-policy LSPE(λ) (Yu, 2010a)

Table 2: Surveyed off-policy and eligibility-traces approaches. Algorithms in black have been published before (provided references), algorithms in red are new.

The rest of this article is organized as follows. Section 2 introduces the background of Markov Decision Processes, describes the state-of-the-art algorithms for learning without eligibility traces, and gives the fundamental idea to extend the methods to the off-policy situation with eligibility traces. Section 3 details this extension for the least-squares approaches: the resulting algorithms are formalized, and we derive recursive and memory-efficient formula for their implementation (this allows online learning without loss of generality, all the more that half of these algorithms are recursive by their very definition), and we discuss their convergence properties. Section 4 does the same job for stochastic gradient approaches, which offers a smaller computational cost (linear per update, instead of quadratic). Last but not least, Section 5 describes an empirical comparison and Section 6 concludes.

2. Background

We consider a Markov Decision Process (MDP), that is a tuple $\{S, A, P, R, \gamma\}$ in which S is a finite state space identified with $\{1, 2, \ldots, N\}$, A a finite action space, $P \in \mathcal{P}(S)^{S\times A}$ the set of transition probabilities, $R \in \mathbb{R}^{S\times A}$ the reward function and γ the discount factor. A mapping $\pi \in \mathcal{P}(A)^S$ is called a policy. For any policy π, let P^π be the corresponding stochastic transition matrix, and R^π the vector of mean reward when following π, that is of components $E_{a|\pi,s}[R(s,a)]$. The value $V^\pi(s)$ of state s for a policy π is the expected discounted cumulative reward starting in state s and then following the policy π:

$$V^\pi(s) = E_\pi\left[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s\right],$$

where E_π denotes the expectation over trajectories induced by policy π. The value function satisfies the (linear) Bellman equation:

$$\forall s, \quad V^\pi(s) = E_{s',a|s,\pi}[R(s,a) + \gamma V^\pi(s')].$$

It can be rewritten as the fixed-point of the Bellman evaluation operator: $V^\pi = T^\pi V^\pi$ where for all V, $\quad T^\pi V = R^\pi + \gamma P^\pi V$.

In this article, we are interested in learning an approximation of this value function V^π under some constraints. First, we assume our approximation to be linearly parameterized:

$$\forall s, \quad \hat{V}_\theta(s) = \theta^T \phi(s)$$

with $\theta \in \mathbb{R}^p$ being the parameter vector and $\phi(s) \in \mathbb{R}^p$ the feature vector in state s. This encompasses notably the tabular case (exact representation of the value function). Also, we want to estimate the value function V^π (or equivalently the associated parameter θ) from a single finite trajectory generated using a possibly different behavioral policy π_0.[1] Let μ_0 be the stationary distribution of the stochastic matrix $P_0 = P^{\pi_0}$ of the *behavior policy* π_0 (we assume it exists and is unique). Let D_0 be the diagonal matrix of which the elements are $(\mu_0(s_i))_{1\leq i\leq N}$. Let Φ be the matrix of feature vectors:

$$\Phi = [\phi(1) \ldots \phi(N)]^T.$$

1. This can be easily extended to multiple finite trajectories.

As we consider a linear approximation, the considered value functions belong to the space spanned by Φ. The projection Π_0 onto this hypothesis space with respect to the μ_0-quadratic norm, which will be central for the understanding of the algorithms, has the following closed-form:

$$\Pi_0 = \Phi(\Phi^T D_0 \Phi)^{-1} \Phi^T D_0.$$

If π_0 is different from π, it is called an off-policy setting. Notice that all algorithms considered in this paper use this Π_0 projection operator, that is the projection according to the observed data.[2] It would certainly be interesting to consider the projection according to the stationary distribution of π, the (unobserved) target policy: this would reduce off-policy learning to on-policy learning. However, this would require re-weighting samples according to the stationary distribution of the target policy π, which is unknown and probably as difficult to estimate as the value function itself.

2.1 Standard Algorithms For On-policy Learning Without Traces

We now review existing on-policy linearly parameterized temporal difference learning algorithms (see Table 1). In this case, the behavior and target policies are the same, so we omit the subscript 0 for the policy (π) and the projection (Π). We assume that a trajectory $(s_1, a_1,\ r_1, s_2, \ldots,\ s_i, a_i, r_i,\ s_{i+1}, \ldots, s_n, a_n, r_n, s_{n+1})$ sampled according to the policy π is available, and will explain how to compute the i^{th} iterate for several algorithms. For all $j \leq i$, let us introduce the empirical Bellman operator at step j:

$$\begin{aligned} \hat{T}_j \ :\ & \mathbb{R}^S \to \mathbb{R} \\ & V \mapsto r_j + \gamma V(s_{j+1}) \end{aligned}$$

so that $\hat{T}_j V$ is an unbiased estimate of $TV(s_j)$.

Projected fixed point approaches aim at finding the fixed-point of the operator being the composition of the projection onto the hypothesis space and the Bellman operator. In other words, they search for the fixed-point $\hat{V}_\theta = \Pi T \hat{V}_\theta$, Π being the just introduced projection operator. Solving the following fixed-point problem,

$$\theta_i = \operatorname*{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^{i} \left(\hat{T}_j \hat{V}_{\theta_i} - \hat{V}_\omega(s_j) \right)^2,$$

with a least-squares approach corresponds to the Least-Squares Temporal Differences (LSTD) algorithm of Bradtke and Barto (1996). Recently, Sutton et al. (2009) proposed two algorithms reaching the same objective, Temporal Difference with gradient Correction (TDC) and Gradient Temporal Difference 2 (GTD2), by performing a stochastic gradient descent of the function $\theta \mapsto \|\hat{V}_\theta - \Pi T \hat{V}_\theta\|^2$ which is minimal (and equal to 0) when $\hat{V}_\theta = \Pi T \hat{V}_\theta$.

2. As far as we know, there are two notable exceptions. Precup et al. (2001) propose an algorithm that updates parameters according to full trajectories (not according to transitions, as all approaches to be reviewed next). Therefore, the distribution weighting the projection operator is the one of the starting states of these trajectories instead of the one involved by the behavioral policy. Another work to move in a different direction is the off-policy approach of Kolter (2011): samples are weighted such that the projection operator composed with the Bellman operator is non-expansive: this is weaker than finding the projection of the stationary distribution, but offers some guarantees. In this article, we consider only the Π_0 projection.

A related approach consists in building a recursive algorithm that repeatedly mimics the iteration $\hat{V}_{\theta_i} \simeq \Pi T \hat{V}_{\theta_{i-1}}$. In practice, we aim at minimizing

$$\omega \mapsto \sum_{j=1}^{i} \left(\hat{T}_j \hat{V}_{\theta_{i-1}} - \hat{V}_\omega(s_j) \right)^2 .$$

Performing the minimization exactly through a least-squares method leads to the Least-Squares Policy Evaluation (LSPE) algorithm of Bertsekas and Ioffe (1996). If this minimization is approximated by a stochastic gradient descent, this leads to the classical Temporal Difference (TD) algorithm (Sutton and Barto, 1998).

Bootstrapping approaches consist in treating value function approximation after seeing the i^{th} transition as a supervised learning problem, by replacing the unobserved values $V^\pi(s_j)$ at states s_j by some estimate computed from the trajectory until the transition (s_j, s_{j+1}), the best such estimate being $\hat{T}_j \hat{V}_{\theta_{j-1}}$. This amounts to minimizing the following function:

$$\omega \mapsto \sum_{j=1}^{i} \left(\hat{T}_j \hat{V}_{\theta_{j-1}} - \hat{V}_\omega(s_j) \right)^2 . \tag{1}$$

Choi and Van Roy (2006) proposed the Fixed-Point Kalman Filter (FPKF), a least-squares variation of TD that minimizes exactly the function of Equation (1). If the minimization is approximated by a stochastic gradient descent, this gives—again—the classical TD algorithm (Sutton and Barto, 1998).

Finally, **residual approaches** aim at minimizing the distance between the value function and its image through the Bellman operator, $\|V - TV\|_{\mu_0}^2$. Based on a trajectory, this suggests the following function to minimize

$$\omega \mapsto \sum_{j=1}^{i} \left(\hat{T}_j \hat{V}_\omega - \hat{V}_\omega(s_j) \right)^2 ,$$

which is a surrogate of the objective $\|V - TV\|_{\mu_0}^2$ that is biased (Antos et al., 2006). This cost function has originally been proposed by Baird (1995) who minimized it using a stochastic gradient approach (this algorithm being referred here as gBRM for gradient-based Bellman Residual Minimization). Both the *parametric* Gaussian Process Temporal Differences (GPTD) algorithm of Engel (2005) and the *linear* Kalman Temporal Differences (KTD) algorithm of Geist and Pietquin (2010b) can be shown to minimize the above cost using a least-squares approach, and are thus the very same algorithm, that we will refer to as BRM (for Bellman Residual Minimization) in the remaining of this paper.[3]

To sum up, it thus appears that after the i^{th} transition has been observed, the above mentioned algorithms behave according to the following pattern:

$$\text{move from } \theta_{i-1} \text{ to } \theta_i \text{ towards the minimum of } \omega \mapsto \sum_{j=1}^{i} \left(\hat{T}_j \hat{V}_\xi - \hat{V}_\omega(s_j) \right)^2 , \tag{2}$$

either through a least-squares approach or a stochastic gradient descent. Each of the algorithms mentioned above is obtained by substituting θ_i, θ_{i-1}, θ_{j-1} or ω for ξ.

3. Note that this is only true in the linear case. GPTD and KTD were both introduced in a more general setting: GPTD is non-parametric and KTD is motivated by the goal of handling nonlinearities.

2.2 Towards Off-policy Learning With Traces

It is now easy to preview, at least at a high level, how one may extend the previously described algorithms so that they can deal with eligibility traces and off-policy learning.

Eligibility Traces. The idea of eligibility traces amounts to looking for the fixed-point of the following variation of the Bellman operator (Bertsekas and Tsitsiklis, 1996)

$$\forall V \in \mathbb{R}^S, \;\; T^\lambda V = (1-\lambda)\sum_{k=0}^{\infty} \lambda^k T^{k+1} V$$

that makes a geometric average with parameter $\lambda \in (0,1)$ of the powers of the original Bellman operator T. Clearly, any fixed-point of T is a fixed-point of T^λ and vice-versa. After some simple algebra, one can see that:

$$\begin{aligned} T^\lambda V &= (I - \lambda\gamma P)^{-1}(R + (1-\lambda)\gamma P V) \\ &= V + (I - \lambda\gamma P)^{-1}(R + \gamma P V - V). \end{aligned} \tag{3}$$

This leads to the following well-known *temporal difference* expression in some state s

$$\begin{aligned} T^\lambda V(s) &= V(s) + E_\pi\left[\sum_{k=i}^{\infty} (\gamma\lambda)^{k-i}\Big(r_k + \gamma V(s_{k+1}) - V(s_k)\Big)\Big| s_i = s\right] \\ &= V(s) + \sum_{k=i}^{\infty} (\gamma\lambda)^{k-i}\delta_{ik}(s) \end{aligned}$$

where we recall that E_π means that the expectation is done according to the target policy π, and where $\delta_{ik}(s) = E_\pi\left[r_k + \gamma V(s_{k+1}) - V(s_k)\Big| s_i = s\right]$ is the expected temporal-difference (Sutton and Barto, 1998). With $\lambda = 0$, we recover the Bellman evaluation equation. With $\lambda = 1$, this is the definition of the value function as the expected and discounted cumulative reward: $T^1 V(s) = E_\pi[\sum_{k=i}^{\infty} \gamma^{k-i} r_k | s_i = s]$.

Off-policy Learning. As before, we assume that we are given a trajectory $(s_1, a_1, r_1, s_2, \ldots, s_j, a_j, r_j, s_{j+1} \ldots, s_n, a_n, r_n, s_{n+1})$, except now that it may be generated from some behavior policy possibly different from the target policy π of which we want to estimate the value. We are going to describe how to compute the i^{th} iterate for several algorithms. For any $i \leq k$, unbiased estimates of the temporal difference terms $\delta_{ik}(s_k)$ can be computed through importance sampling (Ripley, 1987). Indeed, for all s, a, let us introduce the following weight:

$$\rho(s,a) = \frac{\pi(a|s)}{\pi_0(a|s)}.$$

In our trajectory context, for any j and k, write

$$\rho_j^k = \prod_{l=j}^{k} \rho_l \text{ with } \rho_l = \rho(s_l, a_l)$$

with the convention that if $k < j$, $\rho_j^k = 1$. With these notations,

$$\hat{\delta}_{ik} = \rho_i^k \hat{T}_k V - \rho_i^{k-1} V(s_k)$$

is an unbiased estimate of $\delta_{ik}(s_i)$, from which we may build an estimate $\hat{T}_{j,i}^{\lambda} V$ of $T^{\lambda} V(s_j)$ (we will describe this very construction separately for the least-squares and the stochastic gradient as they slightly differ).

Then, by replacing the empirical operator $\hat{T}_j$ in Equation (2) by $\hat{T}_{j,i}^{\lambda}$, we get the general pattern for off-policy trace-based algorithms:

$$\text{move from } \theta_{i-1} \text{ to } \theta_i \text{ towards the minimum of } \omega \mapsto \sum_{j=1}^{i} \left(\hat{T}_{j,i}^{\lambda} \hat{V}_{\xi} - \hat{V}_{\omega}(s_j) \right)^2, \qquad (4)$$

either through a least-squares approach or a stochastic gradient descent after having instantiated $\xi = \theta_i$, θ_{i-1}, θ_{j-1} or ω. This process, including in particular the precise definition of the empirical operator $\hat{T}_{j,i}^{\lambda}$, will be further developed in the next two sections.[4] Since they are easier to derive, we begin by focusing on least-squares algorithms (right column of Table 2) in Section 3. Then, Section 4 focuses on stochastic gradient-based algorithms (left column of Table 2).

3. Least-squares Extensions To Eligibility Traces And Off-policy Learning

First, we consider the least-squares solution to the problem described in Equation (4). At their i^{th} step, the algorithms that we are about to describe will compute the parameter θ_i by *exactly* solving the following problem:

$$\theta_i = \operatorname*{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^{i} \left(\hat{T}_{j,i}^{\lambda} \hat{V}_{\xi} - \hat{V}_{\omega}(s_j) \right)^2$$

where we define the following empirical *truncated* approximation of T_{λ}:

$$\hat{T}_{j,i}^{\lambda} \; : \; \mathbb{R}^S \to \mathbb{R}$$
$$V \mapsto V(s_j) + \sum_{k=j}^{i} (\gamma\lambda)^{k-j} \hat{\delta}_{jk} = V(s_j) + \sum_{k=j}^{i} (\gamma\lambda)^{k-j} \left(\rho_j^k \hat{T}_k V - \rho_j^{k-1} V(s_k) \right).$$

Though different definitions of this operator may lead to practical implementations, note that $\hat{T}_{j,i}^{\lambda}$ only uses samples seen before time i: this very feature—considered by all existing works in the literature—will enable us to derive recursive and low-memory algorithms.

Recall that a linear parameterization is chosen here, $\hat{V}_{\xi}(s_i) = \xi^T \phi(s_i)$. We adopt the following notations:

$$\phi_i = \phi(s_i),\ \Delta\phi_i = \phi_i - \gamma\rho_i\phi_{i+1} \text{ and } \tilde{\rho}_j^{k-1} = (\gamma\lambda)^{k-j} \rho_j^{k-1}.$$

4. Note that we let the empirical operator $\hat{T}_{j,i}^{\lambda}$ depends on the index j of the sample (as before) but also on the step i of the algorithm. This will be particularly useful for the derivation of the recursive and memory-efficient least-squares algorithms that we present in the next section.

The generic cost function to be solved is therefore:

$$\theta_i = \operatorname*{argmin}_{\omega \in \mathbb{R}^p} J(\omega; \xi) \quad \text{with} \quad J(\omega; \xi) = \sum_{j=1}^{i} (\phi_j^T \xi + \sum_{k=j}^{i} \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta \phi_k^T \xi) - \phi_j^T \omega)^2. \quad (5)$$

Before deriving existing and new least-squares algorithms, as announced, some technical lemmas are required.

The first lemma allows computing directly the inverse of a rank-one perturbed matrix.

Lemma 1 (Sherman-Morrison) *Assume that A is an invertible $n \times n$ matrix and that $u, v \in \mathbb{R}^n$ are two vectors satisfying $1 + v^T A^{-1} u \neq 0$. Then:*

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u}.$$

The next lemma is simply a rewriting of imbricated sums. However, it is quite important here as it will allow stepping from the operator $\hat{T}_{j,i}^{\lambda}$ (operator which depends on future of s_j, so acasual)—*forward view* of eligibility traces—to the recursion over parameters using eligibility traces (dependence on only past samples)—*backward view* of eligibility traces. In other words, the forward view is a theoretical way of mixing backups that shifts parametrically (through the choice of λ) from the standard Bellman operator to the Monte Carlo one. However, it cannot be implemented easily, as it requires knowing the future states. On the other hand, the backward view, which is equivalent (see notably Lemma 2 and Proposition 6), is a more mechanistic and convenient viewpoint that allows performing the same updates using solely information gathered in the states encountered in the past. See Sutton and Barto (1998, Chapter 7) for further discussion on *backward/forward views*.

Lemma 2 *Let $f \in \mathbb{R}^{\mathbb{N} \times \mathbb{N}}$ and $n \in \mathbb{N}$. We have:*

$$\sum_{i=1}^{n} \sum_{j=i}^{n} f(i,j) = \sum_{i=1}^{n} \sum_{j=1}^{i} f(j,i)$$

We are now ready to mechanically derive the off-policy algorithms LSTD(λ), LSPE(λ), FPKF(λ) and BRM(λ). This is what we do in the following subsections.

3.1 Off-policy LSTD(λ)

The Least-Squares Temporal Difference algorithm, that computes directly a fixed-point of the projected Bellman operator, has originally been introduced in the no-trace and on-policy case by Bradtke and Barto (1996). It has been extended to eligibility traces by Boyan (1999), to off-policy (through state-action value function approximation) learning (without traces) by Lagoudakis and Parr (2003), and to off-policy learning with traces by Yu (2010a).

The off-policy LSTD(λ) algorithm actually corresponds to instantiating Problem (5) with $\xi = \theta_i$:

$$\theta_i = \operatorname*{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^{i} (\phi_j^T \theta_i + \sum_{k=j}^{i} \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta \phi_k^T \theta_i) - \phi_j^T \omega)^2.$$

This can be solved by zeroing the gradient respectively to ω:

$$\theta_i = (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} \sum_{j=1}^{i} \phi_j (\phi_j^T \theta_i + \sum_{k=j}^{i} \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta\phi_k^T \theta_i))$$

$$\Leftrightarrow \qquad 0 = \sum_{j=1}^{i} \sum_{k=j}^{i} \phi_j \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta\phi_k^T \theta_i),$$

which, through Lemma 2, is equivalent to:

$$0 = \sum_{j=1}^{i} (\sum_{k=1}^{j} \phi_k \tilde{\rho}_k^{j-1}) (\rho_j r_j - \Delta\phi_j^T \theta_i).$$

Introducing the (importance-based) eligibility vector z_j:

$$z_j = \sum_{k=1}^{j} \phi_k \tilde{\rho}_k^{j-1} = \sum_{k=1}^{j} \phi_k (\gamma\lambda)^{j-k} \prod_{m=k}^{j-1} \rho_m = \gamma\lambda\rho_{j-1} z_{j-1} + \phi_j, \tag{6}$$

one obtains the following batch estimate:

$$\theta_i = (\sum_{j=1}^{i} z_j \Delta\phi_j^T)^{-1} \sum_{j=1}^{i} z_j \rho_j r_j = (A_i)^{-1} b_i \tag{7}$$

where

$$A_i = \sum_{j=1}^{i} z_j \Delta\phi_j^T \quad \text{and} \quad b_i = \sum_{j=1}^{i} z_j \rho_j r_j. \tag{8}$$

Thanks to Lemma 1, the inverse $M_i = (A_i)^{-1}$ can be computed recursively:

$$M_i = (\sum_{j=1}^{i} z_j \Delta\phi_j^T)^{-1} = M_{i-1} - \frac{M_{i-1} z_i \Delta\phi_i^T M_{i-1}}{1 + \Delta\phi_i^T M_{i-1} z_i}.$$

This can be used to derive a recursive estimate:

$$\theta_i = (\sum_{j=1}^{i} z_j \Delta\phi_j^T)^{-1} \sum_{j=1}^{i} z_j \rho_j r_j = (M_{i-1} - \frac{M_{i-1} z_i \Delta\phi_i^T M_{i-1}}{1 + \Delta\phi_i^T M_{i-1} z_i}) (\sum_{j=1}^{i-1} z_j r_j \rho_j + z_i \rho_i r_i)$$

$$= \theta_{i-1} + \frac{M_{i-1} z_i}{1 + \Delta\phi_i^T M_{i-1} z_i} (\rho_i r_i - \Delta\phi_i^T \theta_{i-1}).$$

Writing the gain $K_i = \frac{M_{i-1} z_i}{1 + \Delta\phi_i^T M_{i-1} z_i}$, this gives Algorithm 1.

This algorithm has been proposed and analyzed by Yu (2010a). The author proves the following result: if the *behavior* policy π_0 induces an irreducible Markov chain and chooses with positive probability any action that may be chosen by the *target* policy π, and if the compound (linear) operator $\Pi_0 T^\lambda$ has a unique fixed-point, then off-policy LSTD(λ)

Algorithm 1: Off-policy LSTD(λ)

Initialization;
Initialize vector θ_0 and matrix M_0 ;
Set $z_0 = 0$;

for $i = 1, 2, \ldots$ **do**

Observe ϕ_i, r_i, ϕ_{i+1} ;

Update traces ;
$z_i = \gamma\lambda\rho_{i-1} z_{i-1} + \phi_i$;

Update parameters ;
$K_i = \frac{M_{i-1} z_i}{1 + \Delta\phi_i^T M_{i-1} z_i}$;
$\theta_i = \theta_{i-1} + K_i(\rho_i r_i - \Delta\phi_i^T \theta_{i-1})$;
$M_i = M_{i-1} - K_i(M_{i-1}^T \Delta\phi_i)^T$;

converges to it almost surely.[5] Formally, it converges to the solution θ^* of the so-called *projected fixed-point* equation:

$$V_{\theta^*} = \Pi_0 T^\lambda V_{\theta^*}. \tag{9}$$

Using the expression of the projection Π_0 and the form of the Bellman operator in Equation (3), it can be seen that θ^* satisfies (see Yu, 2010a for details)

$$\theta^* = A^{-1} b$$

where

$$A = \Phi^T D_0 (I - \gamma P)(I - \lambda\gamma P)^{-1}\Phi \quad \text{and} \quad b = \Phi^T D_0 (I - \lambda\gamma P)^{-1} R. \tag{10}$$

The core of the analysis of Yu (2010a) consists in showing that $\frac{1}{i}A_i$ and $\frac{1}{i}b_i$ defined in Equation (8) respectively converge to A and b almost surely. Through Equation (7), this implies the convergence of θ_i to θ^*.

3.2 Off-policy LSPE(λ)

The Least-Squares Policy Evaluation algorithm, that computes iteratively the fixed point of the projected Bellman operator, was originally introduced by Bertsekas and Ioffe (1996) and first analyzed in an on-policy context by Nedić and Bertsekas (2003). Its extension to off-policy learning with traces was briefly mentioned by Yu (2010a).

The off-policy LSPE(λ) algorithm corresponds to instantiate $\xi = \theta_{i-1}$ in Problem (5):

$$\theta_i = \operatorname*{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^{i} (\phi_j^T \theta_{i-1} + \sum_{k=j}^{i} \tilde{\rho}_j^{k-1}(\rho_k r_k - \Delta\phi_k^T \theta_{i-1}) - \phi_j^T \omega)^2.$$

5. It is not always the case that $\Pi_0 T^\lambda$ has a unique fixed-point, see Tsitsiklis and Van Roy (1997) for a counter-example.

This can be solved by zeroing the gradient respectively to ω:

$$\theta_i = (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} \sum_{j=1}^{i} \phi_j (\phi_j^T \theta_{i-1} + \sum_{k=j}^{i} \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta \phi_k^T \theta_{i-1}))$$
$$= \theta_{i-1} + (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} \sum_{j=1}^{i} \sum_{k=j}^{i} \phi_j \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta \phi_k^T \theta_{i-1}).$$

Using Lemma 2 and the definition of the eligibility vector z_j in Equation (6), we get:

$$\theta_i = \theta_{i-1} + (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} \sum_{j=1}^{i} \sum_{k=1}^{j} \phi_k \tilde{\rho}_k^{j-1} (\rho_j r_j - \Delta \phi_j^T \theta_{i-1})$$
$$= \theta_{i-1} + (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} \sum_{j=1}^{i} z_j (\rho_j r_j - \Delta \phi_j^T \theta_{i-1}).$$

Define the matrix N_i as follows:

$$N_i = (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} = N_{i-1} - \frac{N_{i-1} \phi_i \phi_i^T N_{i-1}}{1 + \phi_i^T N_{i-1} \phi_i}, \tag{11}$$

where the second equality follows from Lemma 1. Let A_i and b_i be defined as in the LSTD description in Equation (8). For clarity, we restate their definition along with their recursive writing:

$$A_i = \sum_{j=1}^{i} z_j \Delta \phi_j^T = A_{i-1} + z_i \Delta \phi_{i+1}^T$$
$$b_i = \sum_{j=1}^{i} z_j \rho_j r_j = b_{i-1} + z_i \rho_i r_i.$$

Then, it can be seen that the LSPE(λ) update is:

$$\theta_i = \theta_{i-1} + N_i (b_i - A_i \theta_{i-1}).$$

The overall computation is provided in Algorithm 2.

This algorithm, (briefly) mentioned by Yu (2010a), generalizes the LSPE(λ) algorithm of Bertsekas and Ioffe (1996) to off-policy learning. With respect to LSTD(λ), which computes $\theta_i = (A_i)^{-1} b_i$ at each iteration as stated in Equation (7), LSPE(λ) is fundamentally recursive (as it is based on an iterated fixed-point relation). Along with the almost sure convergence of $\frac{1}{i} A_i$ and $\frac{1}{i} b_i$ to A and b defined in Equation (10), it can be shown that iN_i converges to $N = (\Phi^T D_0 \Phi)^{-1}$—see for instance Nedić and Bertsekas (2003)—so that, asymptotically, LSPE(λ) behaves as:

$$\theta_i = \theta_{i-1} + N(b - A\theta_{i-1}) = Nb + (I - NA)\theta_{i-1}$$

Algorithm 2: Off-policy LSPE(λ)

Initialization;
Initialize vector θ_0 and matrix N_0 ;
Set $z_0 = 0$, $A_0 = 0$ and $b_0 = 0$;

for $i = 1, 2, \ldots$ **do**

Observe ϕ_i, r_i, ϕ_{i+1};

Update traces ;
$z_i = \gamma\lambda\rho_{i-1}z_{i-1} + \phi_i$;

Update parameters ;
$N_i = N_{i-1} - \frac{N_{i-1}\phi_i\phi_i^T N_{i-1}}{1+\phi_i^T N_{i-1}\phi_i}$;
$A_i = A_{i-1} + z_i\Delta\phi_i^T$;
$b_i = b_{i-1} + \rho_i z_i r_i$;
$\theta_i = \theta_{i-1} + N_i(b_i - A_i\theta_{i-1})$;

or using the definition of Π_0, A, b from Equation (10) and T^λ from Equation (3):

$$V_{\theta_i} = \Phi\theta_i = \Phi N b + \Phi(I - NA)\theta_{i-1} = \Pi_0 T^\lambda V_{\theta_{i-1}}. \tag{12}$$

The behavior of this sequence depends on whether the spectral radius of $\Pi_0 T^\lambda$ is smaller than 1 or not. Thus, the analyses of Yu (2010a) and Nedić and Bertsekas (2003) (for the convergence of N_i) imply the following convergence result: under the assumptions required for the convergence of off-policy LSTD(λ), and the additional assumption that the operator $\Pi_0 T^\lambda$ has a spectral radius smaller than 1 (so that it is contracting), LSPE(λ) also converges almost surely to the fixed-point of the compound $\Pi_0 T^\lambda$ operator.

There are two sufficient conditions that can ensure such a desired contraction property. The first one is when one considers on-policy learning, as Nedić and Bertsekas (2003) did when they derived the first convergence proof of (on-policy) LSPE(λ). When the behavior policy π_0 is different from the target policy π, a sufficient condition for contraction is that λ be close enough to 1; indeed, when λ tends to 1, the spectral radius of T^λ tends to zero and can potentially balance an expansion of the projection Π_0. In the off-policy case, when γ is sufficiently big, a small value of λ can make $\Pi_0 T^\lambda$ expansive (see Tsitsiklis and Van Roy 1997 for an example in the case $\lambda = 0$) and off-policy LSPE(λ) will then diverge. Eventually, Equations (9) and (12) show that when $\lambda = 1$, both LSTD(λ) and LSPE(λ) asymptotically coincide (as $T^1 V$ does not depend on V).

3.3 Off-policy FPKF(λ)

The Fixed Point Kalman Filter algorithm is a bootstrapped recursive least-squares approach to value function approximation originally introduced by Choi and Van Roy (2006). Its extensions to eligibility traces and to off-policy learning are new.

The off-policy FPKF(λ) algorithm corresponds to instantiate $\xi = \theta_{j-1}$ in Problem (5):

$$\theta_i = \underset{\omega \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^{i} (\phi_j^T \theta_{j-1} + \sum_{k=j}^{i} \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta\phi_k^T \theta_{j-1}) - \phi_j^T \omega)^2.$$

This can be solved by zeroing the gradient respectively to ω:

$$\theta_i = N_i \sum_{j=1}^{i} \phi_j (\phi_j^T \theta_{j-1} + \sum_{k=j}^{i} \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta\phi_k^T \theta_{j-1})),$$

where N_i is the matrix introduced for LSPE(λ) in Equation (11). For clarity, we restate its definition here and its recursive writing:

$$N_i = (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} = N_{i-1} - \frac{N_{i-1}\phi_i \phi_i^T N_{i-1}}{1 + \phi_i^T N_{i-1} \phi_i}. \tag{13}$$

Using Lemma 2, one obtains:

$$\theta_i = N_i (\sum_{j=1}^{i} \phi_j \phi_j^T \theta_{j-1} + \sum_{j=1}^{i} \sum_{k=1}^{j} \phi_k \tilde{\rho}_k^{j-1} (\rho_j r_j - \Delta\phi_j^T \theta_{k-1})).$$

With respect to the previously described algorithms, the difficulty here is that on the right side there is a dependence with all the previous terms θ_{k-1} for $1 \leq k \leq i$. Using the symmetry of the dot product $\Delta\phi_j^T \theta_{k-1} = \theta_{k-1}^T \Delta\phi_j$, it is possible to write a recursive algorithm by introducing the trace matrix Z_j that integrates the subsequent values of θ_k as follows:

$$Z_j = \sum_{k=1}^{j} \tilde{\rho}_k^{j-1} \phi_k \theta_{k-1}^T = Z_{j-1} + \gamma\lambda\rho_{j-1}\phi_j \theta_{j-1}^T.$$

With this notation we obtain:

$$\theta_i = N_i (\sum_{j=1}^{i} \phi_j \phi_j^T \theta_{j-1} + \sum_{j=1}^{i} (z_j \rho_j r_j - Z_j \Delta\phi_j)).$$

Using Equation (13) and a few algebraic manipulations, we end up with:

$$\theta_i = \theta_{i-1} + N_i (z_i \rho_i r_i - Z_i \Delta\phi_i).$$

This is the parameter update as provided in Algorithm 3.

As LSPE(λ), this algorithm is fundamentally recursive. However, its overall behavior is quite different. As we discussed for LSPE(λ), iN_i can be shown to tend asymptotically to $N = (\Phi^T D_0 \Phi)^{-1}$ and FPKF(λ) iterates eventually resemble:

$$\theta_i = \theta_{i-1} + \frac{1}{i} N (z_i \rho_i r_i - Z_i \Delta\phi_i).$$

Algorithm 3: Off-policy FPKF(λ)

Initialization;
Initialize vector θ_0 and matrix N_0 ;
Set $z_0 = 0$ and $Z_0 = 0$;

for $i = 1, 2, \ldots$ **do**

Observe ϕ_i, r_i, ϕ_{i+1};

Update traces ;
$z_i = \gamma\lambda\rho_{i-1} z_{i-1} + \phi_i$;
$Z_i = \gamma\lambda\rho_{i-1} Z_{i-1} + \phi_i \theta_{i-1}^T$;

Update parameters ;
$N_i = N_{i-1} - \frac{N_{i-1}\phi_i \phi_i^T N_{i-1}}{1+\phi_i^T N_{i-1}\phi_i}$;
$\theta_i = \theta_{i-1} + N_i(z_i\rho_i r_i - Z_i \Delta\phi_i)$;

The term in brackets is a random component (that only depends on the previous transitions) and $\frac{1}{i}$ acts as a learning coefficient that asymptotically tends to 0. In other words, FPKF(λ) has a *stochastic approximation* flavor. In particular, one can see FPKF(0) as a stochastic approximation of LSPE(0). Indeed, asymptotically, FPKF(0) does the following update

$$\theta_i = \theta_{i-1} + \frac{1}{i} N(\rho_i \phi_i r_i - \phi_i \Delta\phi_i^T \theta_{i-1}),$$

and one can notice that $\rho_i\phi_i r_i$ and $\phi_i\Delta\phi_i^T$ are samples of A and b to which A_i and b_i converge through LSPE(0). When $\lambda > 0$, the situation is less clear—up to the fact that since $T^1 V$ does not depend on V, we expect FPKF to asymptotically behave like LSTD and LSPE when λ tends to 1.

Due to its much more involved form (notably the matrix trace Z_j integrating the values of all the values θ_k from the start), it does not seem easy to provide a guarantee for FPKF(λ), even in the on-policy case. To our knowledge, there does not exist any *proof of convergence* for stochastic approximation algorithms in the off-policy case with traces, and a related result for FPKF(λ) thus seems difficult.[6] Based on the above-mentioned relation between FPKF(0) and LSPE(0) and the experiments we have run (see Section 5), we conjecture that off-policy FPKF(λ) has the same asymptotic behavior as LSPE(λ). We leave the formal study of this algorithm for future work.

3.4 Off-policy BRM(λ)

The Bellman Residual Minimization algorithm is a least-squares approach that minimizes directly the Bellman residual. The off-policy BRM(λ) algorithm corresponds to instantiate $\xi = \omega$ in Problem (5):

6. An analysis of TD(λ), with a simplifying assumption that forces the algorithm to stay bounded is given by Yu (2010a). An analysis of GQ(λ) is provided by Maei and Sutton (2010), with an assumption on the second moment of the traces, which—as explained in Proposition 2 of Yu (2010a)—does not hold in general. A full analysis of these algorithms thus remains to be done. See also Sections 4.1 and 4.2.

$$\theta_i = \operatorname*{argmin}_{\omega\in\mathbb{R}^p} \sum_{j=1}^{i} (\phi_j^T\omega + \sum_{k=j}^{i} \tilde{\rho}_j^{k-1}(\rho_k r_k - \Delta\phi_k^T\omega) - \phi_j^T\omega)^2$$
$$= \operatorname*{argmin}_{\omega\in\mathbb{R}^p} \sum_{j=1}^{i} (\sum_{k=j}^{i} \tilde{\rho}_j^{k-1}(\rho_k r_k - \Delta\phi_k^T\omega))^2.$$

Define

$$\psi_{j\to i} = \sum_{k=j}^{i} \tilde{\rho}_j^{k-1}\Delta\phi_k \text{ and } z_{j\to i} = \sum_{k=j}^{i} \tilde{\rho}_j^{k-1}\rho_k r_k.$$

This yields the following batch estimate:

$$\theta_i = \operatorname*{argmin}_{\omega\in\mathbb{R}^p} \sum_{j=1}^{i} (z_{j\to i} - \psi_{j\to i}^T\omega)^2 = (\tilde{A}_i)^{-1}\tilde{b}_i \tag{14}$$

where

$$\tilde{A}_i = \sum_{j=1}^{i} \psi_{j\to i}\psi_{j\to i}^T \quad \text{and} \quad \tilde{b}_i = \sum_{j=1}^{i} \psi_{j\to i} z_{j\to i}.$$

The transformation of this batch estimate into a recursive update rule is somewhat tedious (it involves three "trace" variables), and the details are deferred to Appendix A for clarity. The resulting BRM(λ) method is provided in Algorithm 4. Note that at each step, this algorithm involves the inversion of a 2×2 matrix (involving the 2×2 identity matrix I_2), inversion that admits a straightforward analytical solution. The computational complexity of an iteration of BRM(λ) is thus $O(p^2)$ (as for the preceding least-squares algorithms).

GPTD and KTD, which are close to BRM, have also been extended with some trace mechanism; however, GPTD(λ) (Engel, 2005), KTD(λ) (Geist and Pietquin, 2010a) and the just described BRM(λ) are different algorithms.[7] Briefly, GPTD(λ) is very close to LSTD(λ) and KTD(λ) uses a different Bellman operator.[8] As BRM(λ) builds a linear system whose solution is updated recursively, it resembles LSTD(λ). However, the system it builds is different. The following theorem, proved in Appendix B, partially characterizes the behavior of BRM(λ) and its potential limit.[9]

Theorem 3 *Assume that the stochastic matrix P_0 of the* behavior *policy is irreducible and has stationary distribution μ_0. Further assume that there exists a coefficient $\beta < 1$ such that*

$$\forall(s,a), \quad \lambda\gamma\rho(s,a) \leq \beta. \tag{15}$$

7. GPTD(λ) is not exactly a generalization of GPTD as it does not reduce to it when $\lambda = 0$. It is rather a technical variation that bridges a gap with the Monte Carlo approach.
8. The corresponding loss is $(\hat{T}_{j,i}^0\hat{V}(\omega) - \hat{V}_\omega(s_j) + \gamma\lambda(\hat{T}_{j+1,i}^1\hat{V}(\omega) - \hat{V}_\omega(s_{j+1})))^2$. With $\lambda = 0$ it gives $\hat{T}_{j,i}^0$ and with $\lambda = 1$ it provides $\hat{T}_{j,i}^1$.
9. Our proof is similar to that of Proposition 4 of Bertsekas and Yu (2009). The overall arguments are the following: Equation (15) implies that the traces can be truncated at some depth l, whose influence on the potential limit of the algorithm vanishes when l tends to ∞. For all l, the l-truncated version of the algorithm can easily be analyzed through the ergodic theorem for Markov chains. Making l tend to ∞ allows tying the convergence of the original arguments to that of the truncated version. Eventually, the formula for the limit of the truncated algorithm is computed and one derives the limit.

Algorithm 4: Off-policy BRM(λ)

Initialization;
Initialize vector θ_0 and matrix C_0 ;
Set $y_0 = 0$, $\mathfrak{D}_0 = 0$ and $z_0 = 0$;

for $i = 1, 2, \dots$ **do**

Observe ϕ_i, r_i, ϕ_{i+1};

Pre-update traces ;
$y_i = (\gamma\lambda\rho_{i-1})^2 y_{i-1} + 1$;

Compute ;
$U_i = \begin{pmatrix} \sqrt{y_i}\Delta\phi_i + \frac{\gamma\lambda\rho_{i-1}}{\sqrt{y_i}}\mathfrak{D}_{i-1} & \frac{\gamma\lambda\rho_{i-1}}{\sqrt{y_i}}\mathfrak{D}_{i-1} \end{pmatrix}^T$;
$V_i = \begin{pmatrix} \sqrt{y_i}\Delta\phi_i + \frac{\gamma\lambda\rho_{i-1}}{\sqrt{y_i}}\mathfrak{D}_{i-1} & -\frac{\gamma\lambda\rho_{i-1}}{\sqrt{y_i}}\mathfrak{D}_{i-1} \end{pmatrix}^T$;
$W_i = \begin{pmatrix} \sqrt{y_i}\rho r_i + \frac{\gamma\lambda\rho_{i-1}}{\sqrt{y_i}} z_{i-1} & -\frac{\gamma\lambda\rho_{i-1}}{\sqrt{y_i}} z_{i-1} \end{pmatrix}^T$;

Update parameters ;
$\theta_i = \theta_{i-1} + C_{i-1}U_i \left(I_2 + V_i C_{i-1} U_i\right)^{-1} \left(W_i - V_i\theta_{i-1}\right)$;
$C_i = C_{i-1} - C_{i-1}U_i \left(I_2 + V_i C_{i-1} U_i\right)^{-1} V_i C_{i-1}$;

Post-update traces ;
$\mathfrak{D}_i = (\gamma\lambda\rho_{i-1})\mathfrak{D}_{i-1} + \Delta\phi_i y_i$;
$z_i = (\gamma\lambda\rho_{i-1}) z_{i-1} + r_i\rho_i y_i$;

Then $\frac{1}{i}\tilde{A}_i$ and $\frac{1}{i}\tilde{b}_i$ respectively converge almost surely to

$$\tilde{A} = \Phi^T \left[D - \gamma DP - \gamma P^T D + \gamma^2 D' + S(I - \gamma P) + (I - \gamma P^T)S^T\right]\Phi$$
$$\tilde{b} = \Phi^T \left[(I - \gamma P^T)Q^T D + S\right] R^\pi$$

where we wrote:

$$D = \operatorname{diag}\left((I - (\lambda\gamma)^2\tilde{P}^T)^{-1}\mu_0\right), \qquad Q = (I - \lambda\gamma P)^{-1},$$
$$D' = \operatorname{diag}\left(\tilde{P}^T(I - (\lambda\gamma)^2\tilde{P}^T)^{-1}\mu_0\right), \qquad S = \lambda\gamma(DP - \gamma D')Q,$$

and where $\tilde{P}$ is the matrix whose coordinates are $\tilde{p}_{ss'} = \sum_a \pi(a|s)\rho(s,a)P(s'|s,a)$. Then, the BRM($\lambda$) algorithm converges with probability 1 to $\tilde{A}^{-1}\tilde{b}$.

The assumption given by Equation (15) trivially holds in the on-policy case (in which $\rho(s,a) = 1$ for all (s,a)) and in the off-policy case when $\lambda\gamma$ is sufficiently small with respect to the mismatch between policies. Note in particular that this result implies the almost sure convergence of the GPTD/KTD algorithms in the on-policy and no-trace case, a question that was still open in the literature.[10] The matrix $\tilde{P}$, which is in general not a stochastic

10. See for instance the conclusion of Engel (2005).

matrix, can have a spectral radius bigger than 1; Equation (15) ensures that $(\lambda\gamma)^2\tilde{P}$ has a spectral radius smaller than β so that D and D' are well defined. Removing assumption of Equation (15) does not seem easy, since by tuning $\lambda\gamma$ maliciously, one may force the spectral radius of $(\lambda\gamma)^2\tilde{P}$ to be as close to 1 as one may want, which would make $\tilde{A}$ and $\tilde{b}$ diverge. Though the quantity $\tilde{A}^{-1}\tilde{b}$ may compensate for these divergences, our current proof technique cannot account for this situation and a related analysis constitutes possible future work.

The fundamental idea behind the Bellman Residual approach is to address the computation of the fixed-point of T^λ differently from the previous methods. Instead of computing the projected fixed-point as in Equation (9), one considers the following over-determined system

$$\begin{aligned}
& \Phi\theta \simeq T^\lambda \Phi\theta \\
\Leftrightarrow \quad & \Phi\theta \simeq (I - \lambda\gamma P)^{-1}(R + (1-\lambda)\gamma P\Phi\theta) && \text{by Equation (3)} \\
\Leftrightarrow \quad & \Phi\theta \simeq QR + (1-\lambda)\gamma PQ\Phi\theta \\
\Leftrightarrow \quad & \Psi\theta \simeq QR
\end{aligned}$$

with $\Psi = \Phi - (1-\lambda)\gamma PQ\Phi$, and solves it in a least-squares sense, that is by computing $\theta^* = \bar{A}^{-1}\bar{b}$ with $\bar{A} = \Psi^T\Psi$ and $\bar{b} = \Psi^T QR$. One of the motivations for this approach is that, as opposed to the matrix A of LSTD/LSPE/FPKF, $\bar{A}$ is invertible for all values of λ, and one can always guarantee a finite error bound with respect to the best projection (Schoknecht, 2002; Yu and Bertsekas, 2008; Scherrer, 2010). If the goal of BRM(λ) is to compute $\bar{A}$ and $\bar{b}$ from samples, what it actually computes ($\tilde{A}$ and $\tilde{b}$ as characterized in Theorem 3) will in general be biased because the estimation is based on a single trajectory.[11] Such a bias adds an uncontrolled variance term (Antos et al., 2006) to $\bar{A}$ and $\bar{b}$; an interesting consequence is that $\tilde{A}$ is always non-singular.[12] More precisely, there are two sources of bias in the estimation: one results from the non Monte-Carlo evaluation (the fact that $\lambda < 1$) and the other from the use of the correlated importance sampling factors (as soon as one considers off-policy learning). The interested reader may check that in the on-policy case, and when λ tends to 1, $\tilde{A}$ and $\tilde{b}$ coincide with $\bar{A}$ and $\bar{b}$. However, in the strictly off-policy case, taking $\lambda = 1$ does not prevent the bias due to the correlated importance sampling factors. If we have argued that LSTD/LSPE/FPKF should asymptotically coincide when $\lambda = 1$, we see here that BRM should generally differ in an off-policy situation.

4. Stochastic Gradient Extensions To Eligibility Traces And Off-policy Learning

We have just provided a systematic derivation of all least-squares algorithms for learning with eligibility traces in an off-policy manner. When the number of features p is very large, the $O(p^2)$ complexity involved by a least-squares approach may be prohibitive. In such a situation, a natural alternative is to consider an approach based on a stochastic gradient

11. It is possible to remove the bias when $\lambda = 0$ by using double samples. However, in the case where $\lambda > 0$, the possibility to remove the bias seems much more difficult.

12. $\bar{A}$ is by construction positive definite, and $\tilde{A}$ equals $\bar{A}$ plus a positive term (the variance term), and is thus also positive definite.

descent of the objective function of interest (Bottou and Bousquet, 2011; Sutton et al., 2009; Maei and Sutton, 2010).

In this section, we will describe a systematic derivation of stochastic gradient algorithms for learning in an off-policy manner with eligibility traces. The principle followed is the same as for the least-squares approaches: we shall instantiate the algorithmic pattern of Equation (4) by choosing the value of ξ and update the parameter so as move towards the minimum of $J(\theta_i, \xi)$ using a stochastic gradient descent. To make the pattern of Equation (4) precise, we need to define the empirical approximate operator we use. We will consider the *untruncated* $\hat{T}_{i,n}^{\lambda}$ operators (written in the followings $\hat{T}_i^{\lambda}$, with a slight abuse of notation):

$$\hat{T}_i^{\lambda} V = V(s_i) + \sum_{j=i}^{n} (\gamma\lambda)^{j-i} \left(\rho_i^j \hat{T}_j V - \rho_i^{j-1} V(s_j) \right) \tag{16}$$

where n is the total length of the trajectory.

It should be noted that algorithmic derivations in this section will be a little bit more involved than in the least-squares case. First, by instantiating $\xi = \theta_i$, the pattern given in Equation (4) is actually a fixed-point problem onto which one cannot directly perform a stochastic gradient descent; this issue will be addressed in Section 4.2 through the introduction of an auxiliary objective function, following the approach originally proposed by Sutton et al. (2009). A second difficulty is the following: the just introduced empirical operator $\hat{T}_i^{\lambda}$ depends on the full trajectory after step i (on the future of the process), and is for this reason usually coined a *forward view* estimate. Though it would be possible, in principle, to implement a gradient descent based on this *forward view*, it would not be very memory nor time efficient. Thus, we will follow a usual trick of the literature by deriving recursive algorithms based on a *backward view* estimate that is equivalent to the *forward view* in expectation. To do so, we will repeatedly use the following identity that highlights the fact that the estimate $\hat{T}_i^{\lambda} V$ can be written as a forward recursion:

Lemma 4 *Let $\hat{T}_i^{\lambda}$ be the operator defined in Equation* (16) *and let $V \in \mathbb{R}^S$. We have*

$$\hat{T}_i^{\lambda} V = \rho_i r_i + \gamma\rho_i(1-\lambda) V(s_{i+1}) + \gamma\lambda\rho_i \hat{T}_{i+1}^{\lambda} V.$$

Proof Using notably the identity $\rho_i^j = \rho_i \rho_{i+1}^j$, we have:

$$\begin{aligned}
\hat{T}_i^{\lambda} V &= V(s_i) + \sum_{j=i}^{n} (\gamma\lambda)^{j-i} \left(\rho_i^j \hat{T}_j V - \rho_i^{j-1} V(s_j) \right) \\
&= V(s_i) + \rho_i \hat{T}_i V - V(s_i) + \gamma\lambda\rho_i \sum_{j=i+1}^{n} (\rho_i^j \hat{T}_j V - \rho_i^{j-1} V(s_j)) \\
&= \rho_i \hat{T}_i V + \gamma\lambda\rho_i \left(\hat{T}_{i+1}^{\lambda} V - V(s_{i+1}) \right).
\end{aligned}$$

■

To sum up, the "recipe" that we are about to use to derive off-policy gradient learning algorithms based on eligibility traces will consist of the following steps:

1. write the empirical generic cost function of Equation (4) with the untruncated Bellman operator of Equation (16) ;

2. instantiate ξ and derive the gradient-based update rule (with some additional work for $\xi = \theta_i$, see Section 4.2);

3. turn the *forward view* into an equivalent (in expectation) *backward view*.

The next subsection details the precise derivation of the algorithms.

4.1 Off-policy TD(λ)

The Temporal-Difference algorithm (Sutton and Barto, 1998) is a gradient-based bootstrap approach for value function approximation. Because it is the simplest, we begin by considering this bootstrap approach, that is by instantiating $\xi = \theta_{j-1}$. The cost function to be minimized is therefore:

$$\sum_{j=1}^{i} \left(\hat{T}_j^\lambda \hat{V}_{\theta_{j-1}} - \hat{V}_\omega(s_j) \right)^2 .$$

Minimized with a stochastic gradient descent, the related update rule is (α_i being a standard learning rate and recalling that $\hat{V}_\omega(s_i) = \omega^T \phi(s_i) = \omega^T \phi_i$):

$$\begin{aligned} \theta_i &= \theta_{i-1} - \frac{\alpha_i}{2} \nabla_\omega \left(\hat{T}_i^\lambda \hat{V}_{\theta_{i-1}} - \hat{V}_\omega(s_i) \right)^2 \Big|_{\omega=\theta_{i-1}} \\ &= \theta_{i-1} + \alpha_i \phi_i \left(\hat{T}_i^\lambda \hat{V}_{\theta_{i-1}} - \hat{V}_{\theta_{i-1}}(s_i) \right). \end{aligned} \tag{17}$$

At this point, one could notice that the exact same update rule would have been obtained by instantiating $\xi = \theta_{i-1}$. This was to be expected: as only the last term of the sum is considered for the update, we have $j = i$, and therefore $\xi = \theta_{i-1} = \theta_{j-1}$.

Equation (17) makes use of a λ-TD error defined as

$$\delta_i^\lambda(\omega) = \hat{T}_i^\lambda \hat{V}_\omega - \hat{V}_\omega(s_i).$$

For convenience, let also δ_i be the standard (off-policy) TD error defined as

$$\delta_i(\omega) = \delta_i^{\lambda=0}(\omega) = \rho_i \hat{T}_i \hat{V}_\omega - \hat{V}_\omega(s_i) = \rho_i \left(r_i + \gamma \hat{V}_\omega(s_{i+1}) \right) - \hat{V}_\omega(s_i).$$

The λ-TD error can be expressed as a forward recursion:

Lemma 5 *Let δ_i^λ be the λ-TD error and δ_i be the standard TD error. Then for all ω,*

$$\delta_i^\lambda(\omega) = \delta_i(\omega) + \gamma\lambda\rho_i \delta_{i+1}^\lambda(\omega).$$

Proof This is a corollary of Lemma 4:

$$\begin{aligned} & \hat{T}_i^\lambda V_\omega \;=\; \rho_i r_i + \gamma\rho_i(1-\lambda) V_\omega(s_{i+1}) + \gamma\lambda\rho_i \hat{T}_{i+1}^\lambda V_\omega \\ \Leftrightarrow \quad & \hat{T}_i^\lambda V_\omega - V_\omega(s_i) \;=\; \rho_i r_i + \gamma\rho_i V_\omega(s_{i+1}) - V_\omega(s_i) + \gamma\lambda\rho_i (\hat{T}_{i+1}^\lambda V_\omega - V_\omega(s_{i+1})) \\ \Leftrightarrow \quad & \delta_i^\lambda(\omega) \;=\; \delta_i(\omega) + \gamma\lambda\rho_i \delta_{i+1}^\lambda(\omega). \end{aligned}$$

■

Therefore, we get the following update rule

$$\theta_i = \theta_{i-1} + \alpha_i \phi_i \delta_i^\lambda(\theta_{i-1})$$

with $\delta_i^\lambda(\theta_{i-1}) = \delta_i(\theta_{i-1}) + \gamma\lambda\delta_{i+1}^\lambda(\theta_{i-1})$. The key idea here is to find some backward recursion such that in expectation, when the Markov chain has reached its steady state distribution μ_0, it provides the same result as the forward recursion. Such a backward recursion is given by the following lemma.

Proposition 6 *Let z_i be the eligibility vector, defined by the following recursion:*

$$z_i = \phi_i + \gamma\lambda\rho_{i-1} z_{i-1}.$$

For all ω, we have

$$E_{\mu_0}[\phi_i \delta_i^\lambda(\omega)] = E_{\mu_0}[z_i \delta_i(\omega)].$$

Proof For clarity, we omit the dependence with respect to ω and write below δ_i (resp. δ_i^λ) for $\delta_i(\omega)$ (resp. $\delta_i^\lambda(\omega)$). The result relies on successive applications of Lemma 5. We have:

$$\begin{aligned} E_{\mu_0}[\phi_i \delta_i^\lambda] &= E_{\mu_0}[\phi_i(\delta_i + \gamma\lambda\rho_i \delta_{i+1}^\lambda)] \\ &= E_{\mu_0}[\phi_i \delta_i] + E_{\mu_0}[\phi_i \gamma\lambda\rho_i \delta_{i+1}^\lambda]. \end{aligned}$$

Moreover, we have that $E_{\mu_0}[\phi_i\rho_i\delta_{i+1}^\lambda] = E_{\mu_0}[\phi_{i-1}\rho_{i-1}\delta_i^\lambda]$, as expectation is done according to the stationary distribution, therefore:

$$\begin{aligned} E_{\mu_0}[\phi_i \delta_i^\lambda] &= E_{\mu_0}[\phi_i \delta_i] + \gamma\lambda E_{\mu_0}[\phi_{i-1}\rho_{i-1}\delta_i^\lambda] \\ &= E_{\mu_0}[\phi_i \delta_i] + \gamma\lambda E_{\mu_0}[\phi_{i-1}\rho_{i-1}(\delta_i + \gamma\lambda\rho_i\delta_{i+1}^\lambda)] \\ &= E_{\mu_0}[\delta_i(\phi_i + \gamma\lambda\rho_{i-1}\phi_{i-1} + (\gamma\lambda)^2\rho_{i-1}\rho_{i-2}\phi_{i-2} + \dots)] \\ &= E_{\mu_0}[\delta_i z_i]. \end{aligned}$$

■

This suggests to replace Equation (17) by the following update rule,

$$\theta_i = \theta_{i-1} + \alpha_i z_i \delta_i(\theta_{i-1}),$$

which is equivalent in expectation when the Markov chain has reached its steady state. This is summarized in Algorithm 5.

This algorithm was first proposed in the tabular case by Precup et al. (2000) (who call it per-decision importance sampling). An off-policy TD(λ) algorithm (with function approximation) was proposed by Precup et al. (2001), but it differs significantly from the algorithm just described, since it updates parameters based on full episodic trajectories rather than based on the current transition. Algorithm 5 was actually first proposed much more recently by Bertsekas and Yu (2009).

An important issue for the analysis of this algorithm is the fact that the trace z_i may have an infinite variance, due to importance sampling (Yu, 2010b, Section 3.1). As far as we know, the only existing analysis of off-policy TD(λ) (as provided in Algorithm 5) uses an additional constraint which forces the parameters to be bounded: after each parameter update, the

Algorithm 5: Off-policy TD(λ)

Initialization;
Initialize vector θ_0;
Set $z_0 = 0$;
for $i = 1, 2, \ldots$ **do**
  Observe ϕ_i, r_i, ϕ_{i+1} ;
  Update traces ;
  $z_i = \gamma\lambda\rho_{i-1}z_{i-1} + \phi_i$;
  Update parameters ;
  $\theta_i = \theta_{i-1} + \alpha_i z_i(\rho_i r_i - \Delta\phi_i^T\theta_{i-1})$;

resulting parameter vector is projected onto some predefined compact set. This analysis is performed by Yu (2010b, Section 4.1). Under the standard assumptions of stochastic approximations and most of the assumptions required for the on-policy TD(λ) algorithm, assuming moreover that $\Pi_0 T^\lambda$ is a contraction (which we recall to hold for a big enough λ) and that the predefined compact set used to project the parameter vector is a large enough ball containing the fixed point of $\Pi_0 T^\lambda$, the constrained version of off-policy TD(λ) converges to this fixed-point —therefore, the same solution as off-policy LSTD(λ)—, LSPE(λ) and FPKF(λ). We refer to Yu (2010b, Section 4.1) for further details. An analysis of the unconstrained version of off-policy TD(λ) described in Algorithm 5 is an interesting topic for future research.

4.2 Off-policy TDC(λ) And Off-policy GTD2(λ)

The Temporal Difference with gradient Correction and Gradient Temporal Differences 2 algorithms have been introduced by Sutton et al. (2009) as gradient descent approaches to minimize the norm of the difference between the value function and its image through the projected Bellman operator (they are therefore projected fixed-point approaches). Maei and Sutton (2010) extended TDC to off-policy learning with traces, calling the resulting algorithm GQ(λ).

This corresponds (for all algorithms and extensions) to the case $\xi = \theta_i$, considered in this section. Following the general pattern, at step i, we would like to come up with a new parameter θ_i that moves (from θ_{i-1}) closer to the minimum of the function

$$\omega \mapsto J(\omega, \theta_i) = \left(\hat{T}_j^\lambda \hat{V}_{\theta_i} - \hat{V}_\omega(s_j)\right)^2.$$

This problem is tricky since the function to minimize contains what we want to compute—-θ_i—as a parameter. For this reason we cannot directly perform a stochastic gradient descent of the right hand side. Instead, we will consider an alternative (but equivalent) formulation of the projected fixed-point minimization $\theta = \arg\min_\omega \|V_\omega - \Pi_0 T^\lambda V_\omega\|^2$, and will move from θ_{i-1} to θ_i by making one step of gradient descent of an estimate of the function

$$\theta \mapsto \|V_\theta - \Pi_0 T^\lambda V_\theta\|^2.$$

With the following vectorial notations:

$$\begin{aligned}\hat{\mathbf{V}}_\omega &= \begin{pmatrix}\hat{V}_\omega(s_1) & \dots & \hat{V}_\omega(s_i)\end{pmatrix}^T,\\ \hat{\mathbf{T}}^\lambda\hat{\mathbf{V}}_\omega &= \begin{pmatrix}\hat{T}_1^\lambda\hat{V}_\omega & \dots & \hat{T}_i^\lambda\hat{V}_\omega\end{pmatrix}^T,\\ \tilde{\Phi} &= \begin{bmatrix}\phi(s_1) & \dots & \phi(s_i)\end{bmatrix}^T,\\ \tilde{\Pi}_0 &= \tilde{\Phi}(\tilde{\Phi}^T\tilde{\Phi})^{-1}\tilde{\Phi}^T,\end{aligned}$$

we consider the following objective function:

$$\begin{aligned}J(\theta) &= \left\|\hat{\mathbf{V}}_\theta - \tilde{\Pi}_0\hat{\mathbf{T}}^\lambda\hat{\mathbf{V}}_\theta\right\|^2\\ &= \left(\hat{\mathbf{V}}_\theta - \hat{\mathbf{T}}^\lambda\hat{\mathbf{V}}_\theta\right)^T\tilde{\Pi}_0\left(\hat{\mathbf{V}}_\theta - \hat{\mathbf{T}}^\lambda\hat{\mathbf{V}}_\theta\right)\\ &= \left(\sum_{j=1}^i\delta_j^\lambda(\theta)\phi_j\right)^T\left(\sum_{j=1}^i\phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^i\delta_j^\lambda(\theta)\phi_j\right).\end{aligned}$$

This is the derivation followed by Sutton et al. (2009) in the case $\lambda = 0$ and by Maei and Sutton (2010) in the case $\lambda > 0$ (and off-policy learning). Let us introduce the following notation:

$$g_j^\lambda = \nabla\hat{T}_j^\lambda\hat{V}_\theta. \tag{18}$$

Note that since we consider a linear approximation this quantity does not depend on θ. Noticing that $\nabla\delta_j^\lambda(\theta) = \phi_j - g_j^\lambda$, we can compute $\nabla J(\theta)$:

$$\begin{aligned}-\frac{1}{2}\nabla J(\theta) &= -\frac{1}{2}\nabla\left(\sum_{j=1}^i\delta_j^\lambda(\theta)\phi_j\right)^T\left(\sum_{j=1}^i\phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^i\delta_j^\lambda(\theta)\phi_j\right)\\ &= -\left(\nabla\sum_{j=1}^i\delta_j^\lambda(\theta)\phi_j\right)^T\left(\sum_{j=1}^i\phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^i\delta_j^\lambda(\theta)\phi_j\right)\\ &= \left(\sum_{j=1}^i(\phi_j - g_j^\lambda)\phi_j^T\right)\left(\sum_{j=1}^i\phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^i\delta_j^\lambda(\theta)\phi_j\right)\\ &= \left(\sum_{j=1}^i\delta_j^\lambda(\theta)\phi_j\right) - \left(\sum_{j=1}^i g_j^\lambda\phi_j^T\right)\left(\sum_{j=1}^i\phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^i\delta_j^\lambda(\theta)\phi_j\right).\end{aligned} \tag{19}$$

Let $w_i(\theta)$ be a quasi-stationary estimate of the last part, that can be recognized as the solution of a least-squares problem (regression of λ-TD errors δ_j^λ on features ϕ_j):

$$w_i(\theta) \approx \left(\sum_{j=1}^i\phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^i\delta_j^\lambda(\theta)\phi_j\right) = \operatorname*{argmin}_\omega\sum_{j=1}^i\left(\phi_j^T\omega - \delta_j^\lambda(\theta)\right)^2.$$

The identification with the above least-squares solution suggests to use the following stochastic gradient descent to form the quasi-stationary estimate:

$$w_i = w_{i-1} + \beta_i \phi_i \left(\delta_i^\lambda(\theta_{i-1}) - \phi_i^T w_{i-1} \right).$$

This update rule makes use of the λ-TD error, defined through a *forward view*. As for the previous algorithm, we can use Proposition 6 to obtain the following *backward view* update rule that is equivalent (in expectation when the Markov chain reaches its steady state):

$$w_i = w_{i-1} + \beta_i \left(z_i \delta_i(\theta_{i-1}) - \phi_i(\phi_i^T w_{i-1}) \right). \tag{20}$$

Using this quasi-stationary estimate, the gradient can be approximated as:

$$-\frac{1}{2}\nabla J(\theta) \approx \left(\sum_{j=1}^{i} \delta_j^\lambda(\theta) \phi_j \right) - \left(\sum_{j=1}^{i} g_j^\lambda \phi_j^T \right) w_i.$$

Therefore, a stochastic gradient descent gives the following update rule for the parameter vector θ:

$$\theta_i = \theta_{i-1} + \alpha_i \left(\delta_i^\lambda(\theta_{i-1}) \phi_i - g_i^\lambda \phi_i^T w_i \right). \tag{21}$$

Once again the *forward view* term $\delta_i^\lambda(\theta_{i-1})\phi_i$ can be turned into a *backward view* by using Proposition 6. There remains to work on the term $g_i^\lambda \phi_i^T$.

First, one can notice that the term g_i^λ satisfies a forward recursion.

Lemma 7 *We have*

$$g_i^\lambda = \gamma \rho_i (1-\lambda) \phi_{i+1} + \gamma \lambda \rho_i g_{i+1}^\lambda.$$

Proof This result is simply obtained by applying the gradient to the forward recursion of $\hat{T}_i^\lambda V_\theta$ provided in Lemma 4 (according to θ). ■

Using this, the term $g_i^\lambda \phi_i^T$ can be worked out similarly to the term $\delta_i^\lambda(\theta_{i-1})\phi_i$.

Proposition 8 *Let z_i be the eligibility vector defined in Proposition 6. We have*

$$E_{\mu_0}[g_i^\lambda \phi_i^T] = E_{\mu_0}[\gamma \rho_i (1-\lambda) \phi_{i+1} z_i^T].$$

Proof The proof is similar to that of Proposition 6. Writing $b_i = \gamma \rho_i (1-\lambda)\phi_{i+1}$ and $\eta_i = \gamma \lambda \rho_i$, we have

$$\begin{aligned} E_{\mu_0}[g_i^\lambda \phi_i^T] &= E_{\mu_0}[(b_i + \eta_i g_{i+1}^\lambda)\phi_i^T] \\ &= E_{\mu_0}[b_i \phi_i^T] + E_{\mu_0}[\eta_{i-1}(b_i + \eta_i g_{i+1}^\lambda)\phi_{i-1}^T] \\ &= E_{\mu_0}[b_i z_i^T]. \end{aligned}$$

■

Using this result and Proposition 6, it is natural to replace Equation (21) by an update based on a backward recursion:

$$\theta_i = \theta_{i-1} + \alpha_i \left(z_i \delta_i - \gamma \rho_i (1-\lambda) \phi_{i+1} (z_i^T w_{i-1}) \right). \tag{22}$$

Last but not least, for the estimate w_i to be indeed quasi-stationary, the learning rates should satisfy the following condition (in addition to the classical conditions):

$$\lim_{i\to\infty} \frac{\alpha_i}{\beta_i} = 0.$$

Equations. (22) and (20) define the off-policy TDC(λ) algorithm, summarized in Algorithm 6. It was originally proposed by Maei and Sutton (2010) under the name GQ(λ). We call it off-policy TDC(λ) to highlight the fact that it is the extension of the original TDC algorithm of Sutton et al. (2009) to off-policy learning with traces. One can observe—to our knowledge, this was never mentioned in the literature before—that when $\lambda = 1$, the learning rule of TDC(1) reduces to that of TD(1).

Maei and Sutton (2010) show that the algorithm converges with probability 1 to the same solution as the LSTD(λ) algorithm (that is, to $\theta^* = A^{-1}b$) under some technical assumptions. Contrary to off-policy TD(λ), this algorithm does not requires $\Pi_0 T^\lambda$ to be a contraction in order to be convergent. Unfortunately, one of the assumptions made in the analysis, requiring that the traces z_i have uniformly bounded second moments, is restrictive since in an off-policy setting the traces z_i may easily have an infinite variance (unless the behavior policy is really close to the target policy), as noted by Yu (2010a).[13] A full proof of convergence thus still remains to be done.

Algorithm 6: Off-policy TDC(λ), also known as GQ(λ)

Initialization;
Initialize vector θ_0 and w_0;
Set $z_0 = 0$;

for $i = 1, 2, \ldots$ **do**

Observe ϕ_i, r_i, ϕ_{i+1} ;

Update traces ;
$z_i = \gamma\lambda\rho_{i-1} z_{i-1} + \phi_i$;

Update parameters ;
$\theta_i = \theta_{i-1} + \alpha_i \left(z_i(\rho_i r_i - \Delta\phi_i^T\theta_{i-1}) - \gamma\rho_i(1-\lambda)\phi_{i+1}(z_i^T w_{i-1})\right)$;
$w_i = w_{i-1} + \beta_i \left(z_i(\rho_i r_i - \Delta\phi_i^T\theta_i) - \phi_i(\phi_i^T w_{i-1})\right)$;

Using the same principle—performing a stochastic gradient descent to minimize $J(\theta)$)—, an alternative to TDC, the GTD2 algorithm, was derived by Sutton et al. (2009) in the $\lambda = 0$ case. As far as we know, it has never been extended to off-policy learning with traces; we do it now. Notice that, given the derivation of GQ(λ), obtaining this algorithm is pretty straightforward.

13. See also Randhawa and Juneja (2004).

To do so, we can start back from Equation (19):

$$-\frac{1}{2}\nabla J(\theta) = \left(\sum_{j=1}^{i}(\phi_j - g_j^\lambda)\phi_j^T\right)\left(\sum_{j=1}^{i}\phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^{i}\delta_j^\lambda(\theta)\phi_j\right)$$
$$\approx \left(\sum_{j=1}^{i}(\phi_j - g_j^\lambda)\phi_j^T\right)w_i.$$

This suggests the following alternative update rule (based on forward recursion):

$$\theta_i = \theta_{i-1} + \alpha_i(\phi_i - g_i^\lambda)\phi_i^T w_i.$$

Using Proposition 8, it is natural to use the following alternative update rule, based on a backward recursion:

$$\theta_i = \theta_{i-1} + \alpha_i\left(\phi_i(\phi_i^T w_{i-1}) - \gamma\rho_i(1-\lambda)\phi_{i+1}(z_i^T w_{i-1})\right).$$

The update of w_i remains the same, and put together it gives off-policy GTD2(λ), summarized in Algorithm 7. The analysis of this new algorithm constitutes a potential topic for future research.

Algorithm 7: Off-policy GTD2(λ)

Initialization;
Initialize vector θ_0 and w_0;
Set $z_0 = 0$;
for $i = 1, 2, \ldots$ **do**
  Observe ϕ_i, r_i, ϕ_{i+1} ;
  Update traces ;
  $z_i = \gamma\lambda\rho_{i-1}z_{i-1} + \phi_i$;
  Update parameters ;
  $\theta_i = \theta_{i-1} + \alpha_i\left(\phi_i(\phi_i^T w_{i-1}) - \gamma\rho_i(1-\lambda)\phi_{i+1}(z_i^T w_{i-1})\right)$;
  $w_i = w_{i-1} + \beta_i\left(z_i(\rho_i r_i - \Delta\phi_i^T\theta_i) - \phi_i(\phi_i^T w_{i-1})\right)$;

4.3 Off-policy gBRM(λ)

The algorithm proposed by Baird (1995) minimizes the Bellman residual using a gradient-based approach, in the no-trace and on-policy case. We extend it to eligibility traces and to off-policy learning, which corresponds to instantiate $\xi = \omega$. The cost function to be minimized is then:

$$\sum_{j=1}^{i}\left(\hat{T}_j^\lambda\hat{V}_\omega - \hat{V}_\omega(s_j)\right)^2.$$

Following the negative of the gradient of the last term leads to the following update rule:

$$\begin{aligned}\theta_i &= \theta_{i-1} - \alpha_i \nabla_\omega \left(\hat{T}_i^\lambda \hat{V}_\omega - \hat{V}_\omega(s_i)\right)^2 \Big|_{\omega=\theta_{i-1}} \\ &= \theta_{i-1} - \alpha_i \nabla_\omega \left(\hat{T}_i^\lambda \hat{V}_\omega - \hat{V}_\omega(s_i)\right) \Big|_{\omega=\theta_{i-1}} \left(\hat{T}_i^\lambda \hat{V}_{\theta_{i-1}} - \hat{V}_{\theta_{i-1}}(s_i)\right) \\ &= \theta_{i-1} + \alpha_i \left(\phi_i - g_i^\lambda\right) \delta_i^\lambda(\theta_{i-1}),\end{aligned}$$

recalling the notation $g_i^\lambda = \nabla \hat{T}_i^\lambda \hat{V}_\omega$ first defined in Equation (18).

As usual, this update involves a *forward view*, which we are going to turn into a *backward view*. The term $\phi_i \delta_i^\lambda$ can be worked thanks to Proposition 6. The term $g_i^\lambda \delta_i^\lambda$ is more difficult to handle, as it is the product of two *forward views* (until now, we only considered the product of a *forward view* with a non-recursive term). This can be done thanks to the following original relation (the proof being somewhat tedious, it is deferred to Appendix C):

Proposition 9 *Write* $g_i^\lambda = \nabla_\omega \hat{T}_i^\lambda$ *and define*

$$\begin{aligned} c_i &= 1 + (\gamma\lambda\rho_{i-1})^2 c_{i-1}, \\ \zeta_i &= \gamma\rho_i(1-\lambda)\phi_{i+1}c_i + \gamma\lambda\rho_{i-1}\zeta_{i-1} \\ \text{and } d_i &= \delta_i c_i + \gamma\lambda\rho_{i-1} d_{i-1}. \end{aligned}$$

We have that

$$E_{\mu_0}[\delta_i^\lambda g_i^\lambda] = E_{\mu_0}[\delta_i\zeta_i + d_i\gamma\rho_i(1-\lambda)\phi_{i+1} - \delta_i\gamma\rho_i(1-\lambda)\phi_{i+1}c_i].$$

This result (together with Proposition 6) suggests to update parameters as follows:

$$\theta_i = \theta_{i-1} + \alpha_i \left(\delta_i(z_i + \gamma\rho_i(1-\lambda)\phi_{i+1}c_i - \zeta_i) - d_i\gamma\rho_i(1-\lambda)\phi_{i+1}\right).$$

This gives the off-policy gBRM(λ) algorithm, depicted in Algorithm 8. One can observe that gBRM(1) is equivalent to TD(1) (and thus also TDC(1), *cf.* the comment before the description of Algorithm 6). The analysis of this new algorithm is left for future research.

5. Empirical Study

This section aims at empirically comparing the surveyed algorithms. As they only address the policy evaluation problem, we compare the algorithms in their ability to perform policy evaluation (no control, no policy optimization); however, they may straightforwardly be used in an approximate policy iteration approach (Bertsekas and Tsitsiklis, 1996; Munos, 2003). In order to assess their quality, we consider finite problems where the exact value function can be computed.

More precisely, we consider Garnet problems (Archibald et al., 1995), which are a class of randomly constructed finite MDPs. They do not correspond to any specific application, but are totally abstract while remaining representative of the kind of MDP that might be encountered in practice. In our experiments, a Garnet is parameterized by 4 parameters and is written $\mathcal{G}(n_S, n_A, b, p)$: n_S is the number of states, n_A is the number of actions, b

Algorithm 8: Off-policy gBRM(λ)

Initialization;
Initialize vector θ_0;
Set $z_0 = 0$, $d_0 = 0$, $c_0 = 0$, $\zeta_0 = 0$;
for $i = 1, 2, \ldots$ **do**
  Observe ϕ_i, r_i, ϕ_{i+1} ;
  Update traces ;
  $z_i = \phi_i + \gamma\lambda\rho_{i-1} z_{i-1}$;
  $c_i = 1 + (\gamma\lambda\rho_{i-1})^2 c_{i-1}$;
  $\zeta_i = \gamma\rho_i(1-\lambda)\phi_{i+1}c_i + \gamma\lambda\rho_{i-1}\zeta_{i-1}$;
  $d_i = (\rho_i r_i - \Delta\phi_i^T\theta_{i-1})c_i + \gamma\lambda\rho_{i-1}d_{i-1}$;
  Update parameters ;
  $\theta_i = \theta_{i-1} + \alpha_i \left((\rho_i r_i - \Delta\phi_i^T\theta_{i-1})(z_i + \gamma\rho_i(1-\lambda)\phi_{i+1}c_i - \zeta_i) - d_i\gamma\rho_i(1-\lambda)\phi_{i+1}\right)$;

is a branching factor specifying how many possible next states are possible for each state-action pair (b states are chosen uniformly at random and transition probabilities are set by sampling uniform random $b-1$ cut points between 0 and 1) and p is the number of features (for function approximation). The reward is state-dependent: for a given randomly generated Garnet problem, the reward for each state is uniformly sampled between 0 and 1. Features are chosen randomly: Φ is a $n_S \times p$ feature matrix of which each component is randomly and uniformly sampled between 0 and 1. The discount factor γ is set to 0.95 in all experiments.

We consider two types of problems, "small" and "big", respectively corresponding to instances $\mathcal{G}(30, 2, 2, 8)$ and $\mathcal{G}(100, 4, 3, 20)$. We also consider two types of learning: on-policy and off-policy. In the on-policy setting, for each Garnet a policy π to be evaluated is randomly generated (by sampling randomly $n_A - 1$ cut points between 0 and 1 for each state), and trajectories (to be used for learning) are sampled according to this same policy. In the off-policy setting, the policy π to be evaluated is randomly generated the same way, but trajectories are sampled according to a different (similarly randomly generated) behavior policy π_0.

For all algorithms, we choose $\theta_0 = 0$. For least-squares algorithms (LSTD, LSPE, FPKF and BRM), we set the initial matrices (M_0, N_0, C_0) to $10^3 I$ (the higher this value, the more negligible its effect on estimates; we observed that this parameter did not play a crucial role in practice). We run a first set of experiments in order to set all other parameters (eligibility factor and learning rates). We use the following schedule for the learning rates:

$$\alpha_i = \alpha_0 \frac{\alpha_c}{\alpha_c + i} \text{ and } \beta_i = \beta_0 \frac{\beta_c}{\beta_c + i^{\frac{2}{3}}}.$$

More precisely, we generate 30 problems (MDPs and policies) for each possible combination small/big on-policy/off-policy (leading to four cases). For each problem, we generate one trajectory of length 10^4 using the behavioral policy (which is the randomly generated target

policy in the on-policy case and the behavior policy in the off-policy case), to be used by all algorithms. For each meta-parameter, we consider the following ranges of values: $\lambda \in \{0, 0.4, 0.7, 0.9, 1\}$, $\alpha_0 \in \{10^{-2}, 10^{-1}, 10^{0}\}$, $\alpha_c \in \{10^{1}, 10^{2}, 10^{3}\}$, $\beta_0 \in \{10^{-2}, 10^{-1}, 10^{0}\}$ and $\beta_c \in \{10^{1}, 10^{2}, 10^{3}\}$. Then, we compute the parameter estimates considering all algorithms instantiated with each possible combination of the meta-parameters. This gives for each combination a family $\theta_{i,d}$ with i the number of transitions encountered in the trajectory of the d^{th} problem. Finally, for each case, for all problems and each algorithm, we choose the combination of meta-parameters which minimizes the average error on the last one-tenth of the averaged (over all problems) learning curves (we do this to reduce the sensitivity to the initialization and the transient behavior). Formally, we pick the set of parameters that minimizes the following quantity:

$$\text{err} = \frac{1}{30} \sum_{d=1}^{30} \frac{1}{10^3} \sum_{i=9.10^3}^{10^4} \|\Phi\theta_{i,d} - V^{\pi_d}\|_2.$$

We provide the empirical results of this first set of experiments in Tables 3 to 6. As a complement, we detail in Figure 1 the sensitivity of all algorithms with respect to the *main* parameter λ that controls the eligibility traces (averaged over the 30 problems, with the best global meta-parameters for each choice of λ). We comment these results below.

	λ	α_0	α_c	β_0	β_c	err
LSTD	1.0					2.07
LSPE	1.0					2.07
FPKF	1.0					2.07
BRM	1.0					2.07
TD	1.0	10^{-2}	10^{3}			2.06
gBRM	1.0	10^{-2}	10^{3}			2.06
TDC	1.0	10^{-2}	10^{3}	10^{-2}	10^{1}	2.06
GTD2	1.0	10^{-2}	10^{3}	10^{-1}	10^{2}	2.05

Table 3: Small problem ($\mathcal{G}(30, 2, 2, 8)$), on-policy learning ($\pi = \pi_0$).

Table 3 shows the best global meta-parameters over the 30 considered instances (one trajectory per instance) of a small Garnet problem in an on-policy setting, as well as related efficiency. Numerically, all methods provide equivalent performance (the slight difference of error is not statistically significant, provided the variance of the estimates). All methods use the same eligibility factor ($\lambda = 1$), leading to a Monte Carlo estimate, to reach their best performance. Figure 1 (top, left) shows that this choice of λ does matter and that BRM, gBRM and FPKF are more sensitive to a good choice of the eligibility factor.

Table 4 shows the best global meta-parameters over the 30 considered instances (one trajectory per instance) of a big Garnet problem in an on-policy setting, as well as related performance. These results are consistent with those of the small problem, in the on-policy setting (with rather different meta-parameters, apart from the eligibility factor). Here again, the algorithms need the highest value of λ to perform the best, except TDC and GTD2 that

	λ	α_0	α_c	β_0	β_c	err
LSTD	1.0					1.20
LSPE	1.0					1.20
FPKF	1.0					1.20
BRM	1.0					1.20
TD	1.0	10^{-1}	10^{1}			1.25
gBRM	1.0	10^{-1}	10^{1}			1.25
TDC	0.9	10^{-1}	10^{2}	10^{-1}	10^{2}	1.21
GTD2	0.9	10^{-1}	10^{2}	10^{-2}	10^{3}	1.22

Table 4: Big problem ($\mathcal{G}(100, 4, 3, 20)$), on-policy learning ($\pi = \pi_0$).

take nevertheless a high value of λ. Figure 1 (top, right) suggests that as the problem's size grows, the role of the eligibility factor gets more prominent (but with a similar behavior).

	λ	α_0	α_c	β_0	β_c	err
LSTD	0.4					3.69
LSPE	0.4					3.69
FPKF	0.7					4.74
BRM	0.0					4.42
TD	0.4	10^{-1}	10^{2}			3.85
gBRM	0.0	10^{-2}	10^{1}			10.42
TDC	0.4	10^{-1}	10^{1}	10^{-2}	10^{1}	7.81
GTD2	0.4	10^{-1}	10^{3}	10^{-2}	10^{1}	4.53

Table 5: Small problem ($\mathcal{G}(30, 2, 2, 8)$), off-policy learning ($\pi \neq \pi_0$).

Table 5 reports the best meta-parameters in an off-policy setting for a small problem (still for 30 instances). Regarding the least-squares methods, LSTD and LSPE get the best results, whereas FPKF and BRM suffer more from the off-policy aspect. Regarding gradient methods, TD's performance is good (it is close to that of LSTD/LSPE and better than BRM/FPKF), followed closely by GTD2. TDC and gBRM lead to the worse results. All algorithms use a small or intermediate value of the eligibility factor. Increasing λ would reduce the bias, but the performance would suffer from the variance due to importance sampling, as shown also in Figure 1 (bottom, left).

Eventually, Table 6 shows the meta-parameters and performance in the most difficult situation: the off-policy setting of the big problem. These results are consistent with the off-policy results of the small problem, summarized in Table 5. LSTD and LSPE are the most efficient least-squares algorithms and choose the smallest possible value $\lambda = 0$. FPKF and BRM's performance deteriorate (significantly for the latter). TD behaves very well and GTD2 follows closely. The performance of TDC and gBRM are the worse. Figure 1 (bottom, right) is similar to that of the small problem. It shows that TD (with a good learning rate) is quite stable, in particular more than LSTD/LSPE.

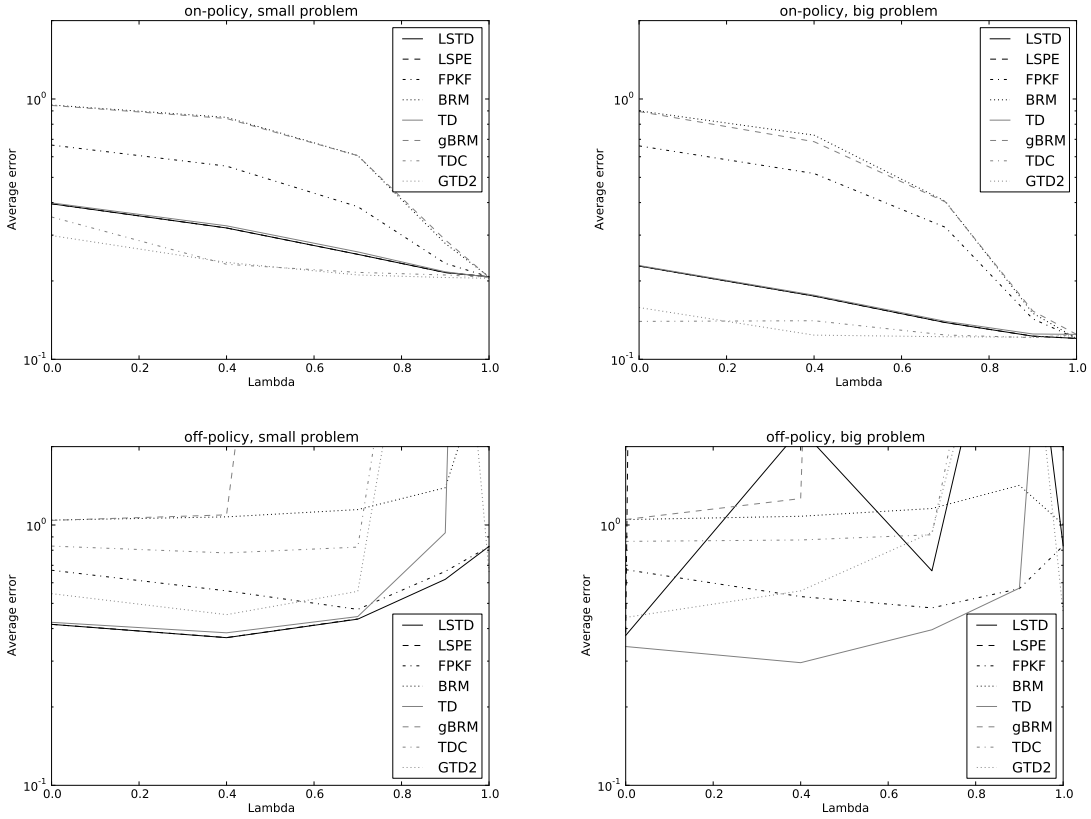


Figure 1: Sensitivity of performance of the algorithms (y-axis, in logarithmic scale) with respect to the eligibility trace parameter λ (x-axis). Left: Small problem ($\mathcal{G}(30, 2, 2, 8)$), right: Big problem ($\mathcal{G}(100, 4, 3, 20)$). Top: on-policy learning ($\pi = \pi_0$), bottom: off-policy learning ($\pi \neq \pi_0$).

The main goal of the series of experiments we have just described was to choose reasonable values for the meta-parameters. We have also used these experiments to quickly comment the relative performance of the algorithms, but this is not statistically significant as this was based on a few (random) problems, onto which meta-parameters have been optimized. Though we will see that the general behavior of the algorithm is globally consistent with what we have seen so far, the series of experiments that we are about to describe aims at providing such a statistically significant performance comparison. For each situation (small and big problems, on- and off-policy), we fix the meta-parameters to the previously reported values and we compare the algorithms on several new instances of the problems. These results are reported on Figures 2 to 5. For each of the 4 problems, we randomly generate 100 instances (MDP and policy to be evaluated). For each such problem, we generate a trajectory of length 10^5. Then, all algorithms learn using this very trajectory. On each figure, we report the average performance (left), measured as the difference between the true

	λ	α_0	α_c	β_0	β_c	err
LSTD	0					3.76
LSPE	0					3.86
FPKF	0.7					4.80
BRM	1.0					10.05
TD	0.4	10^{-1}	10^{1}			2.96
gBRM	0.0	10^{-2}	10^{1}			10.50
TDC	0.0	10^{-1}	10^{1}	10^{-2}	10^{1}	8.65
GTD2	0.0	10^{-1}	10^{3}	10^{-2}	10^{1}	4.41

Table 6: Big problem ($\mathcal{G}(100, 4, 3, 20)$), off-policy learning ($\pi \neq \pi_0$).

value function (computed from the model) and the currently estimated one, $\|V^\pi - \Phi\theta\|_2$, as well as the associated standard deviation (right).

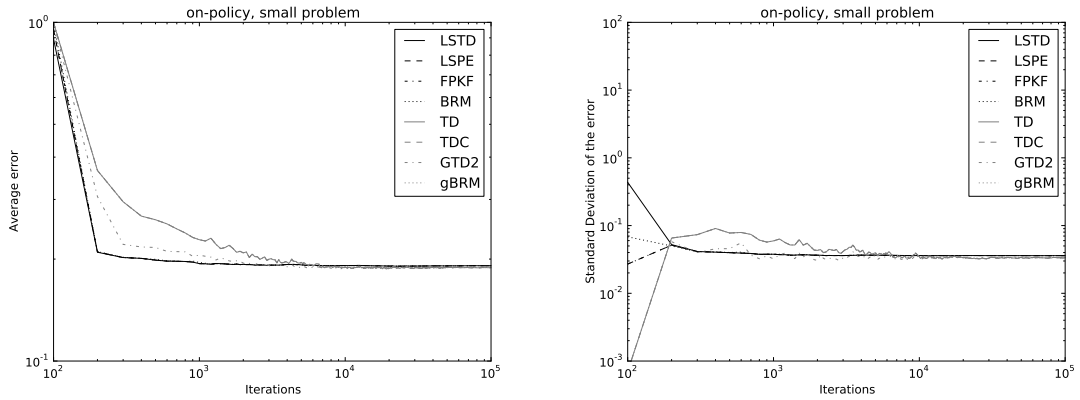


Figure 2: Performance for small problems ($\mathcal{G}(30, 2, 2, 8)$), on-policy learning ($\pi = \pi_0$) (left: average error, right: standard deviation).

We begin by discussing the results in the on-policy setting. Figure 2 compares all algorithms for 100 randomly generated small problems (that is, each run corresponds to different dynamics, reward function, features and evaluated policy), the meta-parameters being those provided in Table 3. All least-squares approaches provide the best results and are bunched together; this was to be expected, as all algorithms use λ equal to 1. The gBRM, TD and TDC algorithms provide the same results (being equivalent with the choice $\lambda = 1$), they are slower than GTD2, which is slower than the least-squares algorithms. Figure 3 compares the algorithms for 100 randomly generated big problems, the meta-parameters being those provided in Table 4. These result are similar to those of the small problem in an off-policy setting, except that TDC has now a different (and slower) behavior, due to the different choice of the eligibility factor ($\lambda = 0.9$). GTD2 is still the better gradient-based algorithm.

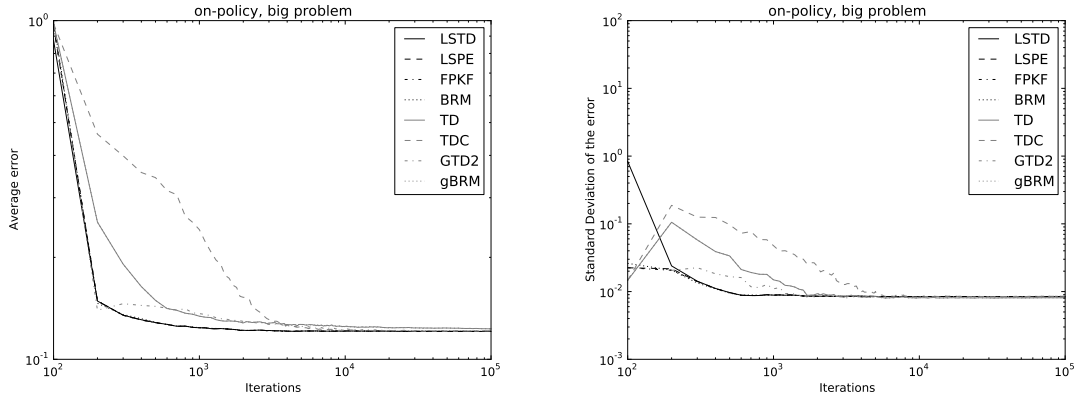


Figure 3: Performance for big problems ($\mathcal{G}(100, 4, 3, 20)$), on-policy learning ($\pi = \pi_0$) (left: average error, right: standard deviation).

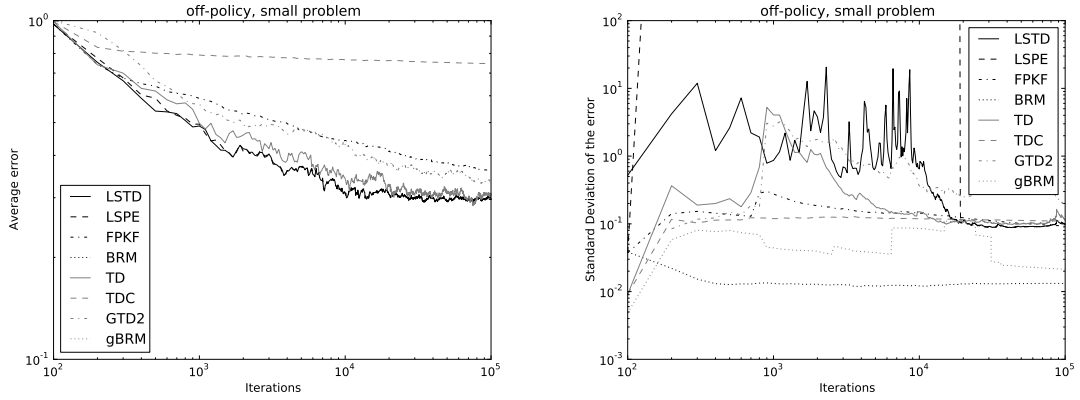


Figure 4: Performance for small problems ($\mathcal{G}(30, 2, 2, 8)$), off-policy learning ($\pi \neq \pi_0$) (left: average error, right: standard deviation).

We now consider the off-policy setting. Figure 4 provides the average performance and standard deviation of the algorithms (meta-parameters being those of Table 5) on 100 small problems. Once again, we can see that LSTD/LSPE provide the best results. The two other least-squares methods (FPKF and BRM) are overtaken by the gradient-based TD algorithm, that follows closely LSTD/LSPE. GTD2 is a little bit slower and TDC is the slowest algorithm. Figure 5 provides the same data for the big problems (with the meta-parameters of Table 6). These results are similar to those of the small problems in an off-policy setting, except that TD is even closer to LSTD/LSPE (but requires the choice of a learning rate).

Summary. Overall, our experiments suggest that the two best algorithms are LSTD and LSPE, since they converge much faster in all situations with less parameter tuning. The gradient-based TD algorithm globally displays a good behavior and constitutes a good alter-

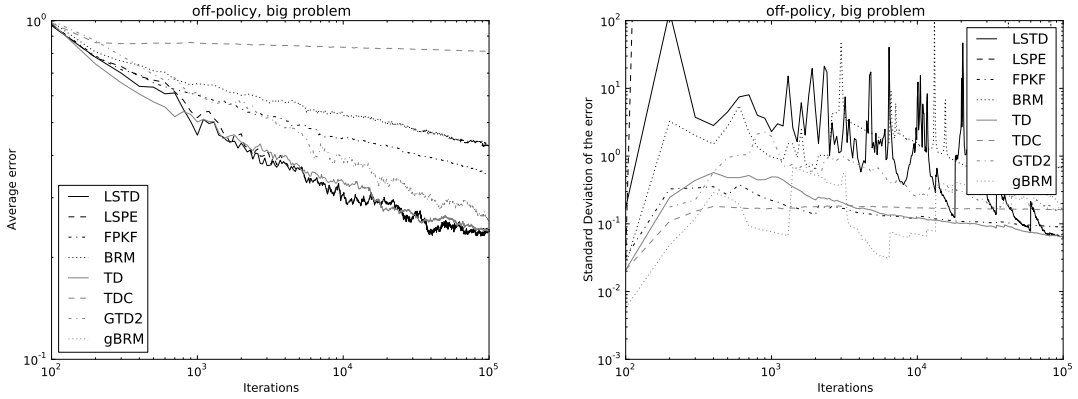


Figure 5: Performance for big problems ($\mathcal{G}(100, 4, 3, 20)$), on-policy learning ($\pi \neq \pi_0$) (left: average error, right: standard deviation).

native when the number p of features is too big for least-squares methods to be implemented. Though some new algorithms/extensions show interesting results (FPKF(λ) is consistently better that the state-of-the-art FPKF by Choi and Van Roy 2006, gBRM works well in the on-policy setting) most of the other algorithms do not seem to be empirically competitive with the *trio* LSTD/LSPE/TD, especially in off-policy situations. In particular, the algorithm introduced specifically for the off-policy setting (TDC/GTD2) are much slower than TD in the off-policy case (but GTD2 is faster in the on-policy experiments, yet with more parameter tuning). Moreover, the condition required for the good behavior of LSPE, FPKF and TD—the contraction of $\Pi_0 T^\lambda$—does not seem to be very restrictive in practice (at least for the Garnet problems we considered): though it is possible to build specific pathological examples where these algorithms diverge, this never happened in our experiments.[14]

6. Conclusion And Future Work

We have considered least-squares and gradient-based algorithms for value estimation in an MDP context. Starting from the on-policy case with no trace, we have recalled that several algorithms (LSTD, LSPE, FPKF and BRM for least-squares approaches, TD, gBRM and TDC/GTD2 for gradient-based approaches) fall in a common algorithmic pattern: Equation (2). Substituting the original Bellman operator by an operator that deals with traces and off-policy samples naturally leads to the state-of-the-art off-policy trace-based versions of LSTD, LSPE, TD and TDC, and suggests natural extensions of FPKF, BRM, gBRM and GTD2. This way, we surveyed many known and new off-policy eligibility trace-based algorithms for policy evaluation.

We have explained how to derive recursive (memory and time-efficient) implementations of all these algorithms and discussed their known convergence properties, including an original analysis of BRM(λ) for sufficiently small λ, that implies the so far not known convergence

14. A preliminary version of this article (Scherrer and Geist, 2011) contains such examples, and also an example where an adversarial choice of λ leads to the divergence of LSTD(λ).

of GPTD/KTD. Interestingly, it appears that the analysis of off-policy trace-based stochastic gradient algorithms under mild assumptions is still an open problem: the only currently known analysis of TD (Yu, 2010a) only applies to a constrained version of the algorithm, and that of TDC (Maei and Sutton, 2010) relies on an assumption on the boundedness of the second moment traces that is restrictive (Yu, 2010a). Filling this theoretical gap, as well as providing complete analyses for the other gradient algorithms and FPFK(λ) and BRM(λ) constitute important future work.

Finally, we have illustrated and compared the behavior of these algorithms; this constitutes the first exhaustive empirical comparison of linear methods.[15] Overall, our study suggests that even if the use of eligibility traces generally improves the efficiency of all algorithms, LSTD and LSPE consistently provide the best estimates; and in situations where the computational cost is prohibitive for a least-squares approach (when the number p of features is large), TD probably constitutes the best alternative.

Appendix A. Derivation Of The Recursive Formulas For BRM(λ)

We here detail the derivation of off-policy BRM(λ). We will need two technical lemmas. The first one is the Woodbury matrix identity which generalizes the Sherman-Morrison formula (given in Lemma 1).

Lemma 10 (Woodbury) *Let A, U, C and V be matrices of correct sizes, then:*

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1}+VA^{-1}U)^{-1}VA^{-1}.$$

The second lemma is a rewriting of imbricated sums:

Lemma 11 *Let $f \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}\times\mathbb{N}}$ and $n \in \mathbb{N}$. We have:*

$$\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=i}^{n} f(i,j,k) = \sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j} f(k,i,j) + \sum_{i=2}^{n}\sum_{j=1}^{i-1}\sum_{k=1}^{j} f(k,j,i).$$

As stated in Equation (14), we have the following batch estimate for BRM(λ):

$$\theta_i = \operatorname*{argmin}_{\omega\in\mathbb{R}^p} \sum_{j=1}^{i} (z_{j\to i} - \psi_{j\to i}^T\omega)^2 = (\tilde{A}_i)^{-1}\tilde{b}_i,$$

where

$$\psi_{j\to i} = \sum_{k=j}^{i} \tilde{\rho}_j^{k-1}\Delta\phi_k \text{ and } z_{j\to i} = \sum_{k=j}^{i} \tilde{\rho}_j^{k-1}\rho_k r_k$$

and

$$\tilde{A}_i = \sum_{j=1}^{i} \psi_{j\to i}\psi_{j\to i}^T \quad \text{and} \quad \tilde{b}_i = \sum_{j=1}^{i} \psi_{j\to i} z_{j\to i}.$$

15. To our knowledge, there does not even exist any work reporting and comparing empirical results of LSTD(0) and FPKF(0).

To obtain a recursive formula, these two sums have to be reworked through Lemma 11. Let us first focus on the latter:

$$\sum_{j=1}^{i} \psi_{j\to i} z_{j\to i} = \sum_{j=1}^{i} \sum_{k=j}^{i} \sum_{m=j}^{i} \tilde{\rho}_j^{k-1} \Delta\phi_k \tilde{\rho}_j^{m-1} \rho_m r_m$$
$$= \sum_{j=1}^{i} \sum_{k=1}^{j} \sum_{m=1}^{k} \tilde{\rho}_m^{j-1} \Delta\phi_j \tilde{\rho}_m^{k-1} \rho_k r_k + \sum_{j=2}^{i} \sum_{k=1}^{j-1} \sum_{m=1}^{k} \tilde{\rho}_m^{k-1} \Delta\phi_k \tilde{\rho}_m^{j-1} \rho_j r_j.$$

Writing

$$y_k = \sum_{m=1}^{k} (\tilde{\rho}_m^{k-1})^2 = 1 + (\gamma\lambda\rho_{k-1})^2 y_{k-1},$$

we have that:

$$\sum_{m=1}^{k} \tilde{\rho}_m^{j-1} \tilde{\rho}_m^{k-1} = \tilde{\rho}_k^{j-1} y_k.$$

Therefore:

$$\sum_{j=1}^{i} \psi_{j\to i} z_{j\to i} = \sum_{j=1}^{i} \sum_{k=1}^{j} \tilde{\rho}_k^{j-1} y_k \Delta\phi_j \rho_k r_k + \sum_{j=2}^{i} \sum_{k=1}^{j-1} \tilde{\rho}_k^{j-1} y_k \Delta\phi_k \rho_j r_j.$$

With the following notations:

$$z_j = \sum_{k=1}^{j} \tilde{\rho}_k^{j-1} y_k \rho_k r_k = \gamma\lambda\rho_{j-1} z_{j-1} + \rho_j r_j y_j$$
$$\text{and} \quad \mathfrak{D}_j = \sum_{k=1}^{j} \tilde{\rho}_k^{j-1} y_k \Delta\phi_k = \gamma\lambda\rho_{j-1} \mathfrak{D}_{j-1} + y_j \Delta\phi_j,$$

and with the convention that $z_0 = 0$ and $\mathfrak{D}_0 = 0$, one can write:

$$\sum_{j=1}^{i} \psi_{j\to i} z_{j\to i} = \sum_{j=1}^{i} (\Delta\phi_j \rho_j r_j y_j + \gamma\lambda\rho_{j-1} (\Delta\phi_j z_{j-1} + \rho_j r_j \mathfrak{D}_{j-1})).$$

Similarly, on can show that:

$$\sum_{j=1}^{i} \psi_{j\to i} \psi_{j\to i}^T = \sum_{j=1}^{i} (\Delta\phi_j \Delta\phi_j^T y_j + \gamma\lambda\rho_{j-1} (\Delta\phi_j \mathfrak{D}_{j-1}^T + \mathfrak{D}_{j-1} \Delta\phi_j^T)).$$

Denoting

$$u_j = \sqrt{y_j} \Delta\phi_j,$$
$$v_j = \frac{\gamma\lambda\rho_{j-1}}{\sqrt{y_j}} \mathfrak{D}_{j-1},$$

and I_2 the 2×2 identity matrix, we have:

$$\sum_{j=1}^{i} \psi_{j\to i}\psi_{j\to i}^T = \sum_{j=1}^{i}((u_j+v_j)(u_j+v_j)^T - v_j v_j^T)$$
$$= \sum_{j=1}^{i-1} \psi_{j\to i}\psi_{j\to i}^T + \underbrace{\begin{pmatrix} u_i+v_i & v_i \end{pmatrix}}_{=U_i} I_2 \underbrace{\begin{pmatrix} (u_i+v_i)^T \\ -v_i^T \end{pmatrix}}_{=V_i}.$$

We can apply the Woodbury identity given in Lemma 10:

$$C_i = \left(\sum_{j=1}^{i} \psi_{j\to i}\psi_{j\to i}^T\right)^{-1} = \left(\sum_{j=1}^{i-1} \psi_{j\to i} z_{j\to i} + U_i I_2 V_i\right)^{-1}$$
$$= C_{i-1} - C_{i-1}U_i \left(I_2 + V_i C_{i-1} U_i\right)^{-1} V_i C_{i-1}.$$

The other sum can also be reworked:

$$\tilde{b}_i = \sum_{j=1}^{i} \psi_{j\to i} z_{j\to i} = \sum_{j=1}^{i} \Delta\phi_j r_j y_j + \gamma\lambda\left(\mathfrak{D}_{j-1} r_j + \Delta\phi_j z_{j-1}\right)$$
$$= \tilde{b}_{i-1} + \Delta\phi_i r_i y_i + \gamma\lambda\left(\mathfrak{D}_{i-1} r_i + \Delta\phi_i z_{i-1}\right) = \tilde{b}_{i-1} + U_i \underbrace{\begin{pmatrix} \sqrt{y_i} r_i + \frac{\gamma\lambda}{\sqrt{y_i}} z_{i-1} \\ -\frac{\gamma\lambda}{\sqrt{y_i}} z_{i-1} \end{pmatrix}}_{=W_i}.$$

Finally, the recursive BRM(λ) estimate can be computed as follows:

$$\theta_i = C_i \tilde{b}_i = \theta_{i-1} + C_{i-1}U_i\left(I_2 + V_i C_{i-1} U_i\right)^{-1}\left(W_i - V_i\theta_{i-1}\right).$$

This gives BRM(λ) as provided in Algorithm 4.

Appendix B. Proof Of Theorem 3: Convergence Of BRM(λ)

The proof of Theorem 3 follows the general idea of that of Proposition 4 of Bertsekas and Yu (2009). It is done in 2 steps. First we argue that the limit of the sequence is linked to that of an alternative algorithm for which one cuts the traces at a certain depth l. Then, we show that for all depth l, this alternative algorithm converges almost surely, we explicitly compute its limit and make l tend to infinity to obtain the limit of BRM(λ).

We will only show that $\frac{1}{i}\tilde{A}_i$ tends to $\tilde{A}$. The argument is similar for $\frac{1}{i}b_i \to \tilde{b}$. Consider the following l-truncated version of the algorithm based on the following alternative traces (we here limit the "memory" of the traces to a size l):

$$y_{k,l} = \sum_{m=\max(1,k-l+1)}^{k} (\tilde{\rho}_m^{k-1})^2,$$
$$\mathfrak{D}_{j,l} = \sum_{k=\max(1,j-l+1)}^{j} \tilde{\rho}_k^{j-1} y_{k,l} \Delta\phi_k,$$

and update the following matrix:

$$\tilde{A}_{i,l} = \tilde{A}_{i-1,l} + \Delta\phi_i \Delta\phi_i^T y_{i,l} + \tilde{\rho}_{i-1}(\Delta\phi_i \mathfrak{D}_{i-1,l}^T + \mathfrak{D}_{i-1,l}\Delta\phi_i^T).$$

The assumption in Equation (15) implies that $\tilde{\rho}_i^{j-1} \leq \beta^{j-i}$, therefore it can be seen that for all k,

$$|y_{k,l} - y_k| = \sum_{m=1}^{\max(0,k-l)} (\tilde{\rho}_m^{k-1})^2 \leq \sum_{m=1}^{\max(0,k-l)} \beta^{2(k-m)} \leq \frac{\beta^{2l}}{1-\beta^2} = \epsilon_1(l)$$

where $\epsilon_1(l)$ tends to 0 when l tends to infinity. Similarly, using the fact that $y_k \leq \frac{1}{1-\beta^2}$ and writing $K = \max_{s,s'} \|\phi(s) - \gamma\phi(s')\|_\infty$, one has for all j,

$$\begin{aligned}
\|\mathfrak{D}_{j,l} - \mathfrak{D}_j\|_\infty &\leq \sum_{k=1}^{\max(0,j-l)} \tilde{\rho}_k^{j-1} \|y_k \Delta\phi_k\|_\infty + \sum_{k=\max(1,j-l+1)}^{j} \tilde{\rho}_k^{j-1} |y_{k,l} - y_k| \|\Delta\phi_k\|_\infty \\
&\leq \sum_{k=1}^{\max(0,j-l)} \tilde{\rho}_k^{j-1} \frac{1}{1-\beta^2} K + \sum_{k=\max(1,j-l+1)}^{j} \tilde{\rho}_k^{j-1} \frac{\beta^{2l}}{1-\beta^2} K \\
&\leq \frac{\beta^l}{1-\beta} \frac{1}{1-\beta^2} K + \frac{1}{1-\beta} \frac{\beta^{2l}}{1-\beta^2} K = \epsilon_2(l)
\end{aligned}$$

where $\epsilon_2(l)$ also tends to 0. Then, it can be seen that:

$$\begin{aligned}
\|\tilde{A}_{i,l} - \tilde{A}_i\|_\infty &= \Big\| \tilde{A}_{i-1,l} - \tilde{A}_{i-1} + \Delta\phi_i \Delta\phi_i^T (y_{i,l} - y_i) \\
&\qquad + \tilde{\rho}_{i-1}(\Delta\phi_i(\mathfrak{D}_{i-1,l}^T - \mathfrak{D}_{i-1}^T) + (\mathfrak{D}_{i-1,l} - \mathfrak{D}_{i-1})\Delta\phi_i^T) \Big\|_\infty \\
&\leq \|\tilde{A}_{i-1,l} - \tilde{A}_{i-1}\|_\infty + \|\Delta\phi_i \Delta\phi_i^T\|_\infty |y_{k,l} - y_k| + 2\beta \|\Delta\phi_i\|_\infty \|\mathfrak{D}_{i-1,l} - \mathfrak{D}_i\|_\infty \\
&\leq \|\tilde{A}_{i-1,l} - \tilde{A}_{i-1}\|_\infty + K^2 \epsilon_1(l) + 2\beta K \epsilon_2(l)
\end{aligned}$$

and, by a recurrence on i, one obtains

$$\left\| \frac{\tilde{A}_{i,l}}{i} - \frac{\tilde{A}_i}{i} \right\|_\infty \leq \epsilon(l)$$

where $\epsilon(l)$ tends to 0 when l tends to infinity. This implies that:

$$\liminf_{l\to\infty} \frac{\tilde{A}_{i,l}}{i} - \epsilon(l) \leq \liminf_{l\to\infty} \frac{\tilde{A}_i}{i} \leq \limsup_{l\to\infty} \frac{\tilde{A}_i}{i} \leq \limsup_{l\to\infty} \frac{\tilde{A}_{i,l}}{i} + \epsilon(l).$$

In other words, one can see that $\lim_{i\to\infty} \frac{\tilde{A}_i}{i}$ and $\lim_{l\to\infty} \lim_{i\to\infty} \frac{\tilde{A}_{i,l}}{i}$ are equal if the latter exists. In the remaining of the proof, we show that the latter limit indeed exists and we compute it explicitly.

Let us fix some l and let us consider the sequence $(\frac{\tilde{A}_{i,l}}{i})$. At some index i, $y_{i,l}$ depends only on the last l samples, while $\mathfrak{D}_{i,l}$ depends on the same samples and the last l values of $y_{j,l}$, thus on the last $2l$ samples. It is then natural to view the computation of $\tilde{A}_{i,l}$, which is

based on $y_{i,l}$, $\mathfrak{D}_{i-1,l}$ and $\Delta\phi_i = \phi_i - \gamma\rho_i\phi_{i+1}$, as being related to a Markov chain of which the states are the $2l+1$ consecutive states of the original chain $(s_{i-2l}, \ldots, s_i, s_{i+1})$. Write E_0 the expectation with respect to its stationary distribution. By the Markov chain Ergodic Theorem, we have with probability 1:

$$\lim_{i\to\infty} \frac{\tilde{A}_{i,l}}{i} = E_0\left[\Delta\phi_{2l}\Delta\phi_{2l}^T y_{2l,l} + \lambda\gamma\rho_{2l-1}(\Delta\phi_{2l}\mathfrak{D}_{2l-1,l}^T + \mathfrak{D}_{2l-1,l}\Delta\phi_{2l}^T)\right]. \tag{23}$$

Let us now explicitly compute this expectation. Write x_i the indicator vector (of which the k^{th} coordinate equals 1 when the state at time i is k and 0 otherwise). One has the following relations: $\phi_i = \Phi^T x_i$. Let us first look at the left part of the above limit:

$$\begin{aligned}
&E_0\left[\Delta\phi_{2l}\Delta\phi_{2l}^T y_{2l,l}\right] \\
&= E_0\left[(\phi_{2l} - \gamma\rho_{2l}\phi_{2l+1})(\phi_{2l} - \gamma\rho_{2l}\phi_{2l+1})^T y_{2l,l}\right] \\
&= E_0\left[\Phi^T(x_{2l} - \gamma\rho_{2l}x_{2l+1})(x_{2l} - \gamma\rho_{2l}x_{2l+1})^T\Phi\left(\sum_{m=l+1}^{2l}(\lambda\gamma)^{2(2l-m)}(\rho_m^{2l-1})^2\right)\right] \\
&= \Phi^T\left\{\sum_{m=l+1}^{2l}(\lambda\gamma)^{2(2l-m)}E_0\left[(\rho_m^{2l-1})^2(x_{2l} - \gamma\rho_{2l}x_{2l+1})(x_{2l} - \gamma\rho_{2l}x_{2l+1})^T\right]\right\}\Phi \\
&= \Phi^T\left\{\sum_{m=l+1}^{2l}(\lambda\gamma)^{2(2l-m)}E_0\left[(X_{m,2l,2l} - \gamma X_{m,2l,2l+1} - \gamma X_{m,2l+1,2l} + \gamma^2 X_{m,2l+1,2l+1})\right]\right\}\Phi
\end{aligned}$$

where we used the definition $\tilde{\rho}_j^{k-1} = (\lambda\gamma)^{k-j}\rho_j^{k-1}$ and the notation $X_{m,i,j} = \rho_m^{i-1}\rho_m^{j-1}x_i x_j^T$. To finish the computation, we will mainly rely on the following Lemma:

Lemma 12 (Some Identities) *Let $\tilde{P}$ be the matrix of which the coordinates are $\tilde{p}_{ss'} = \sum_a \pi(s,a)\rho(s,a)T(s,a,s')$, which is in general not a stochastic matrix. Let μ_0 be the stationary distribution of the behavior policy π_0. Write $\tilde{D}_i = \mathrm{diag}\left((\tilde{P}^T)^i\mu_0\right)$. Then*

$$\begin{aligned}
\forall m \le i,\ E_0[X_{m,i,i}] &= \tilde{D}_{i-m}, \\
\forall m \le i \le j,\ E_0[X_{m,i,j}] &= \tilde{D}_{i-m}P^{j-i}, \\
\forall m \le j \le i,\ E_0[X_{m,i,j}] &= (P^T)^{j-i}\tilde{D}_{i-m}.
\end{aligned}$$

Proof We first observe that:

$$\begin{aligned}
E_0[X_{m,i,i}] &= E_0[(\rho_m^{i-1})^2 x_i x_i^T] \\
&= E_0[(\rho_m^{i-1})^2\,\mathrm{diag}(x_i)] \\
&= \mathrm{diag}\left(E_0[(\rho_m^{i-1})^2 x_i\right).
\end{aligned}$$

To provide the identity, we will thus simply provide a proof by recurrence that $E_0[(\rho_m^{i-1})^2 x_i] = (\tilde{P}^T)^{m-i}\mu_0$. For $i = m$, we have $E_0[x_m] = \mu_0$. Now suppose the relation holds for i and let us prove it for $i+1$.

$$\begin{aligned}
E_0[(\rho_m^i)^2 x_{i+1}] &= E_0\left[E_0[(\rho_m^i)^2 x_{i+1}|\mathcal{F}_i]\right] \\
&= E_0\left[E_0[(\rho_m^{i-1})^2(\rho_i)^2 x_{i+1}|\mathcal{F}_i]\right] \\
&= E_0\left[(\rho_m^{i-1})^2 E_0[(\rho_i)^2 x_{i+1}|\mathcal{F}_i]\right].
\end{aligned}$$

Write $\mathcal{F}_i$ the realization of the process until time i. Recalling that s_i is the state at time i and x_i is the indicator vector corresponding to s_i, one has for all s':

$$\begin{aligned} E_0[(\rho_i)^2 x_{i+1}(s')|\mathcal{F}_i] &= \sum_a \pi_0(s_i,a)\rho(s_i,a)^2 T(s_i,a,s') \\ &= \sum_a \pi(s_i,a)\rho(s_i,a) T(s_i,a,s') \\ &= \tilde{p}_{s_i,s'} \\ &= [\tilde{P}^T x_i](s'). \end{aligned}$$

As this is true for all s', we deduce that $E_0[(\rho_i)^2 x_{i+1}|\mathcal{F}_i] = \tilde{P}^T x_i$ and

$$\begin{aligned} E_0[(\rho_m^i)^2 x_{i+1}] &= E_0[(\rho_m^{i-1})^2 \tilde{P}^T x_i] \\ &= \tilde{P}^T E_0[(\rho_m^{i-1})^2 \tilde{P}^T x_i] \\ &= \tilde{P}^T (\tilde{P}^T)^i \mu_0 \\ &= (\tilde{P}^T)^{i+1} \mu_0 \end{aligned}$$

which concludes the proof by recurrence.

Let us consider the next identity. For $i \leq j$,

$$\begin{aligned} E_0[\rho_m^{i-1}\rho_m^{j-1} x_i x_j^T] &= E_0[E_0[\rho_m^{i-1}\rho_m^{j-1} x_i x_j^T|\mathcal{F}_i]] \\ &= E_0[(\rho_m^{i-1})^2 x_i E_0[\rho_i^{j-1} x_j^T|\mathcal{F}_i]] \\ &= E_0[(\rho_m^{i-1})^2 x_i x_i^T P^{j-i}] \\ &= \mathrm{diag}\left((\tilde{P}^T)^{m-i}\mu_0\right) P^{j-i}. \end{aligned}$$

Eventually, the last identity is obtained by considering $Y_{m,i,j} = X_{m,j,i}^T$. ■

Thus, coming back to our calculus,

$$E_0\left[\Delta\phi_{2l}\Delta\phi_{2l}^T y_{2l,l}\right] = \Phi^T \left\{ \sum_{m=l+1}^{2l} (\lambda\gamma)^{2(2l-m)} \left(\tilde{D}_{2l-m} - \gamma \tilde{D}_{2l-m} P - \gamma P^T \tilde{D}_{2l-m} + \gamma^2 \tilde{D}_{2l+1-m} \right) \right\} \Phi$$

$$= \Phi^T (D_l - \gamma D_l P - \gamma P^T D_l + \gamma^2 D_l') \Phi \tag{24}$$

$$\text{with} \quad D_l = \sum_{j=0}^{l-1} (\lambda\gamma)^{2j} \tilde{D}_j, \quad \text{and} \quad D_l' = \sum_{j=0}^{l-1} (\lambda\gamma)^{2j} \tilde{D}_{j+1}.$$

Similarly, the second term on the right side of Equation (23) satisfies:

$$
\begin{aligned}
&E_0\left[\rho_{2l-1}\mathfrak{D}_{2l-1,l}\Delta\phi_{2l}^T\right] \\
&= E_0\left[\rho_{2l-1}\sum_{k=l}^{2l-1}\tilde{\rho}_k^{2l-2}y_{k,l}\Delta\phi_k\Delta\phi_{2l}^T\right] \\
&= E_0\left[\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\rho_k^{2l-1}\left(\sum_{m=k-l+1}^{k}(\tilde{\rho}_m^{k-1})^2\right)\Phi^T(x_k-\gamma\rho_k x_{k+1})(x_{2l}-\gamma\rho_{2l}x_{2l+1})^T\Phi\Delta\phi_{2l}^T\right] \\
&= \Phi^T\Bigg(\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k} \\
&\qquad\qquad \sum_{m=k-l+1}^{k}(\lambda\gamma)^{2(k-m)}E_0\left[\rho_m^{2l-1}\rho_m^{k-1}(x_k-\gamma\rho_k x_{k+1})(x_{2l}-\gamma\rho_{2l}x_{2l+1})^T\right]\Bigg)\Phi \\
&= \Phi^T\Bigg(\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\sum_{m=k-l+1}^{k}(\lambda\gamma)^{2(k-m)} \\
&\qquad\qquad E_0\left[X_{m,k,2l}-\gamma X_{m,k+1,2l}-\gamma X_{m,k,2l+1}+\gamma^2 X_{m,k+1,2l+1}\right]\Bigg)\Phi \\
&= \Phi^T\Bigg(\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\sum_{m=k-l+1}^{k}(\lambda\gamma)^{2(k-m)} \\
&\qquad\qquad \left(\tilde{D}_{k-m}P^{2l-k}-\gamma\tilde{D}_{k+1-m}P^{2l-k-1}-\gamma\tilde{D}_{k-m}P^{2l+1-k}+\gamma^2\tilde{D}_{k+1-m}P^{2l-k}\right)\Bigg)\Phi \\
&= \Phi^T\Bigg(\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\sum_{m=k-l+1}^{k}(\lambda\gamma)^{2(k-m)} \\
&\qquad\qquad \left(\tilde{D}_{k-m}P^{2l-k}(I-\gamma P)-\gamma\tilde{D}_{k+1-m}P^{2l-1-k}(I-\gamma P)\right)\Bigg)\Phi \\
&= \Phi^T\left(\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\sum_{m=k-l+1}^{k}(\lambda\gamma)^{2(k-m)}\left(\tilde{D}_{k-m}P-\gamma\tilde{D}_{k+1-m}\right)P^{2l-1-k}(I-\gamma P)\right)\Phi \\
&= \Phi^T\left(\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\left(D_lP-\gamma D_l'\right)P^{2l-1-k}(I-\gamma P)\right)\Phi \\
&= \Phi^T\left(D_lP-\gamma D_l'\right)Q_l(I-\gamma P)\Phi
\end{aligned}
$$

with $Q_l = \sum_{j=0}^{l-1}(\lambda\gamma P)^j$.

Gathering this and Equation (24), we see that the limit of $\frac{A_{i,l}}{i}$ expressed in Equation (23) equals:

$$\Phi^T \left[D_l - \gamma D_l P - \gamma P^T D_l + \gamma^2 D'_l \right.$$
$$\left. + \lambda\gamma \left((D_l P - \gamma D'_l) Q_l (I - \gamma P) + (I - \gamma P^T) Q_l^T (P^T D_l - \gamma D'_l) \right) \right] \Phi.$$

When l tends to infinity, Q_l tends to $Q = (I - \lambda\gamma P)^{-1}$. The assumption of Equation (15) ensures that $(\lambda\gamma)\tilde{P}$ has spectral radius smaller than 1, and thus when l tends to infinity, D_l tends to $D = \text{diag}\left((I - (\lambda\gamma)^2 \tilde{P}^T)^{-1} \mu_0 \right)$ and D'_l to $D' = \text{diag}\left(\tilde{P}^T (I - (\lambda\gamma)^2 \tilde{P}^T)^{-1} \mu_0 \right)$. In other words, $\lim_{l\to\infty} \lim_{i\to\infty} \frac{\tilde{A}_{i,l}}{i}$ exists with probability 1 and equals:

$$\Phi^T \left[D - \gamma D P - \gamma P^T D + \gamma^2 D' \right.$$
$$\left. + \lambda\gamma \left((DP - \gamma D') Q (I - \gamma P) + (I - \gamma P^T) Q^T (P^T D - \gamma D') \right) \right] \Phi.$$

Eventually, this shows that $\lim_{i\to\infty} \frac{\tilde{A}_i}{i}$ exists with probability 1 and shares the same value.

A similar reasoning allows to show that $\lim_{i\to\infty} \frac{\tilde{b}_i}{i}$ exists and equals

$$\Phi^T \left[(I - \gamma P^T) Q^T D + \lambda\gamma (DP - \gamma D') Q \right] R^\pi. \quad \blacksquare$$

Appendix C. Proof Of Proposition 9

To prove Proposition 9, we need the following technical lemma.

Lemma 13 *Let α_i and β_i be two forward recursions defined as*

$$\alpha_i = a_i + \eta_i \alpha_{i+1}$$
$$\text{and } \beta_i = b_i + \eta_i \beta_{i+1}.$$

Assume that for any function f we have that (this is typically true if the index i refers to a state sampled according to some stationary distribution, which is the case we are interested in)

$$E[f(a_i, b_i, \eta_i)] = E[f(a_{i-1}, b_{i-1}, \eta_{i-1})].$$

Let also u_i, v_i and w_i be the backward recursions defined as

$$w_i = 1 + \eta_{i-1}^2 w_{i-1},$$
$$u_i = a_i w_i + \eta_{i-1} u_{i-1},$$
$$v_i = b_i w_i + \eta_{i-1} v_{i-1}.$$

Then, we have

$$E[\alpha_i \beta_i] = E[a_i v_i + b_i u_i - a_i b_i w_i].$$

Proof The proof looks like the one of Proposition 6, but is a little bit more complicated. A key equality, to be applied repeatedly, is:

$$\alpha_i \beta_i = (a_i + \eta_i \alpha_{i+1})(b_i + \eta_i \beta_{i+1})$$
$$= a_i \beta_i + b_i \alpha_i + \eta_i^2 \alpha_{i+1} \beta_{i+1} - a_i b_i.$$

Another equality to be used repeatedly makes use of the "stationarity" assumption. For any $k \geq 0$ we have:

$$E[(\prod_{j=0}^{k} \eta_{i-j}^2)\alpha_{i+1}\beta_{i+1}] = E[(\prod_{j=1}^{k+1} \eta_{i-j}^2)\alpha_i\beta_i].$$

These two identities can be used to work the term of interest:

$$\begin{aligned}
E[\alpha_i\beta_i] &= E[(a_i + \eta_i\alpha_{i+1})(b_i + \eta_i\beta_{i+1})] \\
&= E[a_i\beta_i] + E[b_i\alpha_i] + E[\eta_i^2\alpha_{i+1}\beta_{i+1}] - E[a_ib_i] \\
&= E[a_i\beta_i] + E[b_i\alpha_i] - E[a_ib_i] + E[\eta_{i-1}^2(a_i + \eta_i\alpha_{i+1})(b_i + \eta_i\beta_{i+1})] \\
&= E[a_i(1+\eta_{i-1}^2)\beta_i] + E[b_i(1+\eta_{i-1}^2)\alpha_i] - E[a_ib_i(1+\eta_{i-1}^2)] + E[(\eta_{i-1}\eta_i)^2\alpha_{i+1}\beta_{i+1}].
\end{aligned}$$

This process can be repeated, giving

$$E[\alpha_i\beta_i] = E[(a_i\beta_i + b_i\alpha_i - a_ib_i)(1 + \eta_{i-1}^2 + (\eta_{i-1}\eta_{i-2})^2 + \dots)].$$

We have that

$$w_i = 1 + \eta_{i-1}^2 w_{i-1} = 1 + \eta_{i-1}^2 + (\eta_{i-1}\eta_{i-2})^2 + \dots,$$

therefore

$$E[\alpha_i\beta_i] = E[a_iw_i\beta_i] + E[b_iw_i\alpha_i] - E[a_ib_iw_i].$$

We can work on the first term:

$$\begin{aligned}
E[a_iw_i\beta_i] &= E[a_iw_i(b_i + \eta_i\beta_{i+1})] \\
&= E[a_iw_ib_i] + E[a_{i-1}w_{i-1}\eta_{i-1}(b_i + \eta_i\beta_{i+1})] \\
&= E[b_i(a_iw_i + \eta_{i-1}(a_{i-1}w_{i-1}) + \eta_{i-1}\eta_{i-2}(a_{i-2}w_{i-2}) + \dots)] \\
&= E[b_iu_i].
\end{aligned}$$

The work on the second term is symmetric:

$$E[b_iw_i\alpha_i] = E[a_iv_i].$$

This finishes proving the result. ■

The proof of Proposition 9 is a simple application of the preceding technical lemma. By lemma 5, we have that

$$\underbrace{\delta_i^\lambda}_{\doteq\alpha_i} = \underbrace{\delta_i}_{\doteq a_i} + \underbrace{\gamma\lambda\rho_i}_{\doteq\eta_i}\underbrace{\delta_{i+1}^\lambda}_{\doteq\alpha_{i+1}}.$$

By lemma 7, we have that

$$\underbrace{g_i^\lambda}_{\doteq\beta_i} = \underbrace{\gamma\rho_i(1-\lambda)\phi_{i+1}}_{\doteq b_i} + \underbrace{\gamma\lambda\rho_i}_{\doteq\eta_i}\underbrace{g_{i+1}^\lambda}_{\doteq\beta_{i+1}}.$$

The result is then a direct application of Lemma 13.

References

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 *Conference on Learning Theory (COLT)*, 2006.

T. Archibald, K. McKinnon, and L. Thomas. On the generation of Markov decision processes. *Journal of the Operational Research Society*, 46:354–361, 1995.

L.C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *International Conference on Machine Learning (ICML)*, 1995.

D.P. Bertsekas and S. Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. Technical report, MIT, 1996.

D.P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

D.P. Bertsekas and H. Yu. Projected equation methods for approximate solution of large linear systems. *Journal of Computational and Applied Mathematics*, 227:27–50, 2009.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In S. Sra, S. Nowozin, and S.J. Wright, editors, *Optimization for Machine Learning*, pages 351–368. MIT Press, 2011.

J.A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2-3):233–246, 1999.

S.J. Bradtke and A.G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

D. Choi and B. Van Roy. A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning. *Discrete Event Dynamic Systems*, 16:207–239, 2006.

Y. Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, Hebrew University, 2005.

M. Geist and O. Pietquin. Eligibility traces through colored noises. In *IEEE International Conference on Ultra Modern Control Systems (ICUMT)*, 2010a.

M. Geist and O. Pietquin. Kalman temporal differences. *Journal of Artifical Intelligence Research (JAIR)*, 39:483–532, 2010b.

M. Geist and O. Pietquin. Algorithmic survey of parametric value function approximation. *IEEE Transactions on Neural Networks and Learning Systems*, 24(6):845 – 867, 2013.

M. Kearns and S. Singh. Bias-variance error bounds for temporal difference updates. In *Conference on Learning Theory (COLT)*, 2000.

J.Z. Kolter. The fixed points of off-policy TD. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

M.G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003. ISSN 1533-7928.

H.R. Maei and R.S. Sutton. GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Conference on Artificial General Intelligence (AGI)*, 2010.

R. Munos. Error bounds for approximate policy iteration. In *International Conference on Machine Learning (ICML)*, 2003.

A. Nedić and D.P. Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, 13:79–110, 2003.

D. Precup, R.S. Sutton, and S.P. Singh. Eligibility traces for off-policy policy evaluation. In *International Conference on Machine Learning (ICML)*, 2000.

D. Precup, R.S. Sutton, and S. Dasgupta. Off-policy temporal-difference learning with function approximation. In *International Conference on Machine Learning (ICML)*, 2001.

R.S. Randhawa and S. Juneja. Combining importance sampling and temporal difference control variates to simulate Markov chains. *ACM Transactions on Modeling and Computer Simulation*, 14(1):1–30, 2004.

B.D. Ripley. *Stochastic Simulation*. Wiley & Sons, 1987.

B. Scherrer. Should one compute the temporal difference fix point or minimize the Bellman residual? The unified oblique projection view. In *International Conference on Machine Learning (ICML)*, 2010.

B. Scherrer and M. Geist. Recursive least-squares learning with eligibility traces. In *European Workshop on Reinforcement Learning (EWRL)*, 2011.

R. Schoknecht. Optimality of reinforcement learning algorithms with linear function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. MIT Press, 3rd edition, 1998.

R.S. Sutton, H.R. Maei, D. Precup, S. Bhatnagar, D. Silver, Cs. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning (ICML)*, 2009.

J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

H. Yu. Convergence of least-squares temporal difference methods under general conditions. In *International Conference on Machine Learning (ICML)*, 2010a.

H. Yu. Least squares temporal difference methods: An analysis under general condtions. Technical Report C-2010-39, University of Helsinki, September 2010b.

H. Yu and D.P. Bertsekas. New error bounds for approximations from projected linear equations. Technical Report C-2008-43, Dept. Computer Science, Univ. of Helsinki, July 2008.

Journal of Machine Learning Research 15 (2014) 335-366 Submitted 8/13; Revised 12/13; Published 1/14

Early Stopping and Non-parametric Regression: An Optimal Data-dependent Stopping Rule

Garvesh Raskutti RASKUTTI@STAT.WISC.EDU
Department of Statistics
University of Wisconsin-Madison
Madison, WI 53706-1799, USA

Martin J. Wainwright WAINWRIG@BERKELEY.EDU
Bin Yu BINYU@STAT.BERKELEY.EDU
Department of Statistics[*]
University of California
Berkeley, CA 94720-1776, USA

Editor: Sara van de Geer

Abstract

Early stopping is a form of regularization based on choosing when to stop running an iterative algorithm. Focusing on non-parametric regression in a reproducing kernel Hilbert space, we analyze the early stopping strategy for a form of gradient-descent applied to the least-squares loss function. We propose a data-dependent stopping rule that does not involve hold-out or cross-validation data, and we prove upper bounds on the squared error of the resulting function estimate, measured in either the $L^2(\mathbb{P})$ and $L^2(\mathbb{P}_n)$ norm. These upper bounds lead to minimax-optimal rates for various kernel classes, including Sobolev smoothness classes and other forms of reproducing kernel Hilbert spaces. We show through simulation that our stopping rule compares favorably to two other stopping rules, one based on hold-out data and the other based on Stein's unbiased risk estimate. We also establish a tight connection between our early stopping strategy and the solution path of a kernel ridge regression estimator.

Keywords: early stopping, non-parametric regression, kernel ridge regression, stopping rule, reproducing kernel hilbert space, rademacher complexity, empirical processes

1. Introduction

The phenomenon of overfitting is ubiquitous throughout statistics. It is especially problematic in nonparametric problems, where some form of regularization is essential in order to prevent it. In the non-parametric setting, the most classical form of regularization is that of Tikhonov regularization, where a quadratic smoothness penalty is added to the least-squares loss. An alternative and algorithmic approach to regularization is based on early stopping of an iterative algorithm, such as gradient descent applied to the unregularized loss function. The main advantage of early stopping for regularization, as compared to penalized forms, is lower computational complexity.

∗. Also in the Department of Electrical Engineering and Computer Science.

The idea of early stopping has a fairly lengthy history, dating back to the 1970's in the context of the Landweber iteration. For instance, see the paper by Strand (1974) as well as the subsequent papers (Anderssen and Prenter, 1981; Wahba, 1987). Early stopping has also been widely used in neural networks (Morgan and Bourlard, 1990), for which stochastic gradient descent is used to estimate the network parameters. Past work has provided intuitive arguments for the benefits of early stopping. Roughly speaking, it is clear that each step of an iterative algorithm will reduce bias but increase variance, so early stopping ensures the variance of the estimator is not too high. However, prior to the 1990s, there had been little theoretical justification for these claims. A more recent line of work has developed a theory for various forms of early stopping, including boosting algorithms (Bartlett and Traskin, 2007; Buhlmann and Yu, 2003; Freund and Schapire, 1997; Jiang, 2004; Mason et al., 1999; Yao et al., 2007; Zhang and Yu, 2005), greedy methods (Barron et al., 2008), gradient descent over reproducing kernel Hilbert spaces (Caponneto, 2006; Caponetto and Yao, 2006; Vito et al., 2010; Yao et al., 2007), the conjugate gradient algorithm (Blanchard and Kramer, 2010), and the power method for eigenvalue computation (Orecchia and Mahoney, 2011). Most relevant to our work is the paper of Buhlmann and Yu (2003), who derived optimal mean-squared error bounds for L^2-boosting with early stopping in the case of fixed design regression. However, these optimal rates are based on an "oracle" stopping rule, one that cannot be computed based on the data. Thus, their work left open the following natural question: is there a data-dependent and easily computable stopping rule that produces a minimax-optimal estimator?

The main contribution of this paper is to answer this question in the affirmative for a certain class of non-parametric regression problems, in which the underlying regression function belongs to a reproducing kernel Hilbert space (RKHS). In this setting, a standard estimator is the method of kernel ridge regression (Wahba, 1990), which minimizes a weighted sum of the least-squares loss with a squared Hilbert norm penalty as a regularizer. Instead of a penalized form of regression, we analyze early stopping of an iterative update that is equivalent to gradient descent on the least-squares loss in an appropriately chosen coordinate system. By analyzing the mean-squared error of our iterative update, we derive a data-dependent stopping rule that provides the optimal trade-off between the estimated bias and variance at each iteration. In particular, our stopping rule is based on the first time that a running sum of step-sizes after t steps increases above the critical trade-off between bias and variance. For Sobolev spaces and other types of kernel classes, we show that the function estimate obtained by this stopping rule achieves minimax-optimal estimation rates in both the empirical and generalization norms. Importantly, our stopping rule does not require the use of cross-validation or hold-out data.

In more detail, our first main result (Theorem 1) provides bounds on the squared prediction error for all iterates prior to the stopping time, and a lower bound on the squared error for all iterations after the stopping time. These bounds are applicable to the case of fixed design, where as our second main result (Theorem 2) provides similar types of upper bounds for randomly sampled covariates. These bounds are stated in terms of the squared $L^2(\mathbb{P})$ norm or generalization error, as opposed to the in-sample prediction error, or equivalently, the $L^2(\mathbb{P}_n)$ seminorm defined by the data. Both of these theorems apply to any reproducing kernel, and lead to specific predictions for different kernel classes, depending on their eigendecay. For the case of low rank kernel classes and Sobolev spaces, we prove that

our stopping rule yields a function estimate that achieves the minimax optimal rate (up to a constant pre-factor), so that the bounds from our analysis are essentially unimprovable. Our proof is based on a combination of analytic techniques (Buhlmann and Yu, 2003) with techniques from empirical process theory (van de Geer, 2000). We complement these theoretical results with simulation studies that compare its performance to other rules, in particular a method using hold-out data to estimate the risk, as well as a second method based on Stein's Unbiased Risk Estimate (SURE). In our experiments for first-order Sobolev kernels, we find that our stopping rule performs favorably compared to these alternatives, especially as the sample size grows. In Section 3.4, we provide an explicit link between our early stopping strategy and the kernel ridge regression estimator.

2. Background and Problem Formulation

We begin by introducing some background on non-parametric regression and reproducing kernel Hilbert spaces, before turning to a precise formulation of the problem studied in this paper.

2.1 Non-parametric Regression and Kernel Classes

Suppose that our goal is to use a covariate $X \in \mathcal{X}$ to predict a real-valued response $Y \in \mathbb{R}$. We do so by using a function $f : \mathcal{X} \to \mathbb{R}$, where the value $f(x)$ represents our prediction of Y based on the realization $X = x$. In terms of mean-squared error, the optimal choice is the *regression function* defined by $f^*(x) := \mathbb{E}[Y \mid x]$. In the problem of non-parametric regression with random design, we observe n samples of the form $\{(x_i, y_i), i = 1, \ldots, n\}$, each drawn independently from some joint distribution on the Cartesian product $\mathcal{X} \times \mathbb{R}$, and our goal is to estimate the regression function f^*. Equivalently, we observe samples of the form

$$y_i = f^*(x_i) + w_i, \quad \text{for } i = 1, 2, \ldots, n,$$

where $w_i := y_i - f^*(x_i)$ are independent zero-mean noise random variables. Throughout this paper, we assume that the random variables w_i are *sub-Gaussian* with parameter σ, meaning that

$$\mathbb{E}[e^{tw_i}] \leq e^{t^2\sigma^2/2} \quad \text{for all } t \in \mathbb{R}.$$

For instance, this sub-Gaussian condition is satisfied for normal variates $w_i \sim N(0, \sigma^2)$, but it also holds for various non-Gaussian random variables. Parts of our analysis also apply to the fixed design setting, in which we condition on a particular realization $\{x_i\}_{i=1}^n$ of the covariates.

In order to estimate the regression function, we make use of the machinery of reproducing kernel Hilbert spaces (Aronszajn, 1950; Wahba, 1990; Gu and Zhu, 2001). Using $\mathbb{P}$ to denote the marginal distribution of the covariates, we consider a Hilbert space $\mathcal{H} \subset L^2(\mathbb{P})$, meaning a family of functions $g : \mathcal{X} \to \mathbb{R}$, with $\|g\|_{L^2(\mathbb{P})} < \infty$, and an associated inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ under which $\mathcal{H}$ is complete. The space $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) if there exists a symmetric function $\mathbb{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ such that: (a) for each $x \in \mathcal{X}$, the function $\mathbb{K}(\cdot, x)$ belongs to the Hilbert space $\mathcal{H}$, and (b) we have the reproducing relation

$f(x) = \langle f, \mathbb{K}(\cdot, x)\rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$. Any such kernel function must be positive semidefinite. Moreover, under suitable regularity conditions, Mercer's theorem (1909) guarantees that the kernel has an eigen-expansion of the form

$$\mathbb{K}(x, x') = \sum_{k=1}^{\infty} \lambda_k \phi_k(x)\phi_k(x'),$$

where $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq 0$ are a non-negative sequence of eigenvalues, and $\{\phi_k\}_{k=1}^{\infty}$ are the associated eigenfunctions, taken to be orthonormal in $L^2(\mathbb{P})$. The decay rate of the eigenvalues will play a crucial role in our analysis.

Since the eigenfunctions $\{\phi_k\}_{k=1}^{\infty}$ form an orthonormal basis, any function $f \in \mathcal{H}$ has an expansion of the form $f(x) = \sum_{k=1}^{\infty} \sqrt{\lambda_k} a_k \phi_k(x)$, where for all k such that $\lambda_k > 0$, the coefficients

$$a_k := \frac{1}{\sqrt{\lambda_k}} \langle f, \phi_k \rangle_{L^2(\mathbb{P})} = \int_{\mathcal{X}} f(x)\phi_k(x)\, d\mathbb{P}(x)$$

are rescaled versions of the generalized Fourier coefficients.[1] Associated with any two functions in $\mathcal{H}$—where $f = \sum_{k=1}^{\infty} \sqrt{\lambda_k} a_k \phi_k$ and $g = \sum_{k=1}^{\infty} \sqrt{\lambda_k} b_k \phi_k$—are two distinct inner products. The first is the usual inner product in the space $L^2(\mathbb{P})$—namely, $\langle f, g\rangle_{L^2(\mathbb{P})} := \int_{\mathcal{X}} f(x)g(x)\, d\mathbb{P}(x)$. By Parseval's theorem, it has an equivalent representation in terms of the rescaled expansion coefficients and kernel eigenvalues—that is,

$$\langle f, g\rangle_{L^2(\mathbb{P})} = \sum_{k=1}^{\infty} \lambda_k a_k b_k.$$

The second inner product, denoted by $\langle f, g\rangle_{\mathcal{H}}$, is the one that defines the Hilbert space; it can be written in terms of the rescaled expansion coefficients as

$$\langle f, g\rangle_{\mathcal{H}} = \sum_{k=1}^{\infty} a_k b_k.$$

Using this definition, the unit ball for the Hilbert space $\mathcal{H}$ with eigenvalues $\{\lambda_k\}_{k=1}^{\infty}$ and eigenfunctions $\{\phi_k\}_{k=1}^{\infty}$ takes the form

$$\mathbb{B}_{\mathcal{H}}(1) := \big\{ f = \sum_{k=1}^{\infty} \sqrt{\lambda_k} b_k \phi_k \quad \text{for some} \quad \sum_{k=1}^{\infty} b_k^2 \leq 1 \big\}.$$

The class of reproducing kernel Hilbert spaces contains many interesting classes that are widely used in practice, including polynomials of degree d, Sobolev spaces of varying smoothness, and Gaussian kernels. For more background and examples on reproducing kernel Hilbert spaces, we refer the reader to various standard references (Aronszajn, 1950; Saitoh, 1988; Schölkopf and Smola, 2002; Wahba, 1990; Weinert, 1982).

Throughout this paper, we assume that any function f in the unit ball of the Hilbert space is uniformly bounded, meaning that there is some constant $B < \infty$ such that

$$\|f\|_{\infty} := \sup_{x \in \mathcal{X}} |f(x)| \leq B \qquad \text{for all } f \in \mathbb{B}_{\mathcal{H}}(1). \tag{1}$$

1. We have chosen this particular rescaling for later theoretical convenience.

This boundedness condition (1) is satisfied for any RKHS with a kernel such that $\sup_{x \in \mathcal{X}} \mathbb{K}(x, x) \leq B$. Kernels of this type include the Gaussian and Laplacian kernels, the kernels underlying Sobolev and other spline classes, as well as as well as any trace class kernel with trignometric eigenfunctions. The boundedness condition (1) is quite standard in non-asymptotic analysis of non-parametric regression procedures (e.g., van de Geer, 2000). We study non-parametric regression when the unknown function f^* is viewed as fixed, meaning that no prior is imposed on the function space.

2.2 Gradient Update Equation

We now turn to the form of the gradient update that we study in this paper. Given the samples $\{(x_i, y_i)\}_{i=1}^n$, consider minimizing the least-squares loss function

$$\mathcal{L}(f) := \frac{1}{2n} \sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2$$

over some subset of the Hilbert space $\mathcal{H}$. By the representer theorem (Kimeldorf and Wahba, 1971), it suffices to restrict attention to functions f belonging to the span of the kernel functions defined on the data points—namely, the span of $\{\mathbb{K}(\cdot, x_i), i = 1, \ldots, n\}$. Accordingly, we adopt the parameterization

$$f(\cdot) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \omega_i \mathbb{K}(\cdot, x_i), \tag{2}$$

for some coefficient vector $\omega \in \mathbb{R}^n$. Here the rescaling by $1/\sqrt{n}$ is for later theoretical convenience.

Our gradient descent procedure is based on a parameterization of the least-squares loss that involves the *empirical kernel matrix* $K \in \mathbb{R}^{n \times n}$ with entries

$$[K]_{ij} = \frac{1}{n} \mathbb{K}(x_i, x_j) \qquad \text{for } i, j = 1, 2, \ldots, n.$$

For any positive semidefinite kernel function, this matrix must be positive semidefinite, and so has a unique symmetric square root denoted by $\sqrt{K}$. By first introducing the convenient shorthand $y_1^n := \big(y_1 \; y_2 \; \cdots y_n\big) \in \mathbb{R}^n$, we can write the least-squares loss in the form

$$\mathcal{L}(\omega) = \frac{1}{2n} \|y_1^n - \sqrt{n} K \omega\|_2^2.$$

A direct approach would be to perform gradient descent on this form of the least-squares loss. For our purposes, it turns out to be more natural to perform gradient descent in the transformed co-ordinate system $\theta = \sqrt{K}\, \omega$. Some straightforward calculations (see Appendix A for details) yield that the gradient descent algorithm in this new co-ordinate system generates a sequence of vectors $\{\theta_t\}_{t=0}^{\infty}$ via the recursion

$$\theta_{t+1} = \theta_t - \alpha_t \Big(K\, \theta_t - \frac{1}{\sqrt{n}} \sqrt{K}\, y_1^n \Big), \tag{3}$$

where $\{\alpha_t\}_{t=0}^{\infty}$ is a sequence of positive step sizes (to be chosen by the user). We assume throughout that the gradient descent procedure is initialized with $\theta_0 = 0$.

The parameter estimate θ_t at iteration t defines a function estimate f_t in the following way. We first compute[2] the weight vector $\omega^t = \sqrt{K^{-1}}\,\theta_t$, which then defines the function estimate $f_t(\cdot) = \frac{1}{\sqrt{n}} \sum_{i=1}^n \omega_i^t \mathbb{K}(\cdot, x_i)$ as before. In this paper, our goal is to study how the sequence $\{f_t\}_{t=0}^{\infty}$ evolves as an approximation to the true regression function f^*. We measure the error in two different ways: the $L^2(\mathbb{P}_n)$ norm

$$\|f_t - f^*\|_n^2 := \frac{1}{n} \sum_{i=1}^n \big(f_t(x_i) - f^*(x_i)\big)^2$$

compares the functions only at the observed design points, whereas the $L^2(\mathbb{P})$-norm

$$\|f_t - f^*\|_2^2 := \mathbb{E}\Big[\big(f_t(X) - f^*(X)\big)^2\Big]$$

corresponds to the usual mean-squared error.

2.3 Overfitting and Early Stopping

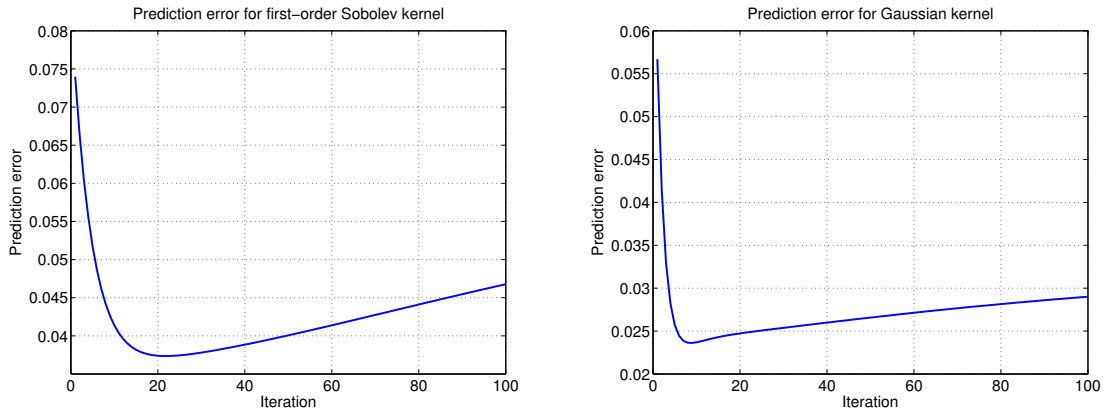
In order to illustrate the phenomenon of interest in this paper, we performed some simulations on a simple problem. In particular, we formed $n = 100$ i.i.d. observations of the form $y = f^*(x_i) + w_i$, where $w_i \sim N(0,1)$, and using the fixed design $x_i = i/n$ for $i = 1, \ldots, n$. We then implemented the gradient descent update (3) with initialization $\theta_0 = 0$ and constant step sizes $\alpha_t = 0.25$. We performed this experiment with the regression function $f^*(x) = |x - 1/2| - 1/2$, and two different choices of kernel functions. The kernel $\mathbb{K}(x, x') = \min\{x, x'\}$ on the unit square $[0,1] \times [0,1]$ generates an RKHS of Lipschitz functions, whereas the Gaussian kernel $\mathbb{K}(x, x') = \exp(-\frac{1}{2}(x - x')^2)$ generates a smoother class of infinitely differentiable functions.

Figure 1 provides plots of the squared prediction error $\|f_t - f^*\|_n^2$ as a function of the iteration number t. For both kernels, the prediction error decreases fairly rapidly, reaching a minimum before or around $T \approx 20$ iterations, before then beginning to increase. As the analysis of this paper will clarify, too many iterations lead to fitting the noise in the data (i.e., the additive perturbations w_i), as opposed to the underlying function f^*. In a nutshell, the goal of this paper is to quantify precisely the meaning of "too many" iterations, and in a data-dependent and easily computable manner.

3. Main Results and Consequences

In more detail, our main contribution is to formulate a data-dependent stopping rule, meaning a mapping from the data $\{(x_i, y_i)\}_{i=1}^n$ to a positive integer $\widehat{T}$, such that the two forms of prediction error $\|f_{\widehat{T}} - f^*\|_n$ and $\|f_{\widehat{T}} - f^*\|_2$ are minimal. In our formulation of such a

2. If the empirical matrix K is not invertible, then we use the pseudoinverse. Note that it may appear as though a matrix inversion is required to estimate ω^t for each t which is computationally intensive. However, the weights ω^t may be computed directly via the iteration $\omega^{t+1} = \omega^t - \alpha_t K(\omega^t - \frac{y_1^n}{\sqrt{n}})$. However, the equivalent update (3) is more convenient for our analysis.



(a) (b)

Figure 1: Behavior of gradient descent update (3) with constant step size $\alpha = 0.25$ applied to least-squares loss with $n = 100$ with equi-distant design points $x_i = i/n$ for $i = 1, \ldots, n$, and regression function $f^*(x) = |x-1/2|-1/2$. Each panel gives plots the $L^2(\mathbb{P}_n)$ error $\|f_t - f^*\|_n^2$ as a function of the iteration number $t = 1, 2, \ldots, 100$. (a) For the first-order Sobolev kernel $\mathbb{K}(x, x') = \min\{x, x'\}$. (b) For the Gaussian kernel $\mathbb{K}(x, x') = \exp(-\frac{1}{2}(x - x')^2)$.

stopping rule, two quantities play an important role: first, the *running sum* of the step sizes

$$\eta_t := \sum_{\tau=0}^{t-1} \alpha_\tau,$$

and secondly, the eigenvalues $\widehat{\lambda}_1 \geq \widehat{\lambda}_2 \geq \cdots \geq \widehat{\lambda}_n \geq 0$ of the empirical kernel matrix K previously defined (2.2). The kernel matrix and hence these eigenvalues are computable from the data. We also note that there is a large body of work on fast computation of kernel eigenvalues (e.g., see Drineas and Mahoney, 2005 and references therein).

3.1 Stopping Rules and General Error Bounds

Our stopping rule involves the use of a model complexity measure, familiar from past work on uniform laws over kernel classes (Bartlett et al., 2005; Koltchinskii, 2006; Mendelson, 2002), known as the local empirical Rademacher complexity. For the kernel classes studied in this paper, it takes the form

$$\widehat{\mathcal{R}}_K(\varepsilon) := \left[\frac{1}{n}\sum_{i=1}^{n} \min\left\{\widehat{\lambda}_i, \varepsilon^2\right\}\right]^{1/2}. \tag{4}$$

For a given noise variance $\sigma > 0$, a closely related quantity—one of central importance to our analysis—is the *critical empirical radius* $\widehat{\varepsilon}_n > 0$, defined to be the smallest positive

solution to the inequality

$$\widehat{\mathcal{R}}_K(\varepsilon) \leq \varepsilon^2/(2e\sigma). \tag{5}$$

The existence and uniqueness of $\widehat{\varepsilon}_n$ is guaranteed for any reproducing kernel Hilbert space; see Appendix D for details. As clarified in our proof, this inequality plays a key role in trading off the bias and variance in a kernel regression estimate.

Our stopping rule is defined in terms of an analogous inequality that involves the running sum $\eta_t = \sum_{\tau=0}^{t-1} \alpha_\tau$ of the step sizes. Throughout this paper, we assume that the step sizes are chosen to satisfy the following properties:

- Boundedness: $0 \leq \alpha_\tau \leq \min\{1, 1/\widehat{\lambda}_1\}$ for all $\tau = 0, 1, 2, \ldots$.
- Non-increasing: $\alpha_{\tau+1} \leq \alpha_\tau$ for all $\tau = 0, 1, 2, \ldots$.
- Infinite travel: the running sum $\eta_t = \sum_{\tau=0}^{t-1} \alpha_\tau$ diverges as $t \to +\infty$.

We refer to any sequence $\{\alpha_\tau\}_{\tau=0}^{\infty}$ that satisfies these conditions as a *valid stepsize sequence*. We then define the *stopping time*

$$\widehat{T} := \arg\min\left\{ t \in \mathbb{N} \mid \widehat{\mathcal{R}}_K\big(1/\sqrt{\eta_t}\big) > (2e\sigma\eta_t)^{-1} \right\} - 1. \tag{6}$$

As discussed in Appendix D, the integer $\widehat{T}$ belongs to the interval $[0, \infty)$ and is unique for any valid stepsize sequence. As will be clarified in our proof, the intuition underlying the stopping rule (6) is that the sum of the step-sizes η_t acts as a tuning parameter that controls the bias-variance tradeoff. The minimizing value is specified by a fixed point of the local Rademacher complexity, in a manner analogous to certain calculations in empirical process theory (van de Geer, 2000; Mendelson, 2002). The stated choice of $\widehat{T}$ optimizes the bias-variance trade-off.

The following result applies to any sequence $\{f_t\}_{t=0}^{\infty}$ of function estimates generated by the gradient iteration (3) with a valid stepsize sequence.

Theorem 1 *Given the stopping time $\widehat{T}$ defined by the rule* (6) *and critical radius $\widehat{\varepsilon}_n$ defined in Equation* (5)*, there are universal positive constants (c_1, c_2) such that the following events hold with probability at least $1 - c_1 \exp(-c_2 n \widehat{\varepsilon}_n^2)$:*

(a) For all iterations $t = 1, 2, ..., \widehat{T}$:

$$\|f_t - f^*\|_n^2 \leq \frac{4}{e\,\eta_t}.$$

(b) At the iteration $\widehat{T}$ chosen according to the stopping rule (6)*, we have*

$$\|f_{\widehat{T}} - f^*\|_n^2 \leq 12\,\widehat{\varepsilon}_n^2.$$

(c) Moreover, for all $t > \widehat{T}$,

$$\mathbb{E}[\|f_t - f^*\|_n^2] \geq \frac{\sigma^2}{4} \eta_t \widehat{\mathcal{R}}_K^2(\eta_t^{-1/2}).$$

3.1.1 Remarks

Although the bounds (a) and (b) are stated as high probability claims, a simple integration argument can be used to show that the expected mean-squared error (over the noise variables, with the design fixed) satisfies a bound of the form

$$\mathbb{E}\big[\|f_t - f^*\|_n^2\big] \leq \frac{4}{e\,\eta_t} \quad \text{for all } t \leq \widehat{T}.$$

To be clear, note that the critical radius $\widehat{\varepsilon}_n$ cannot be made arbitrarily small, since it must satisfy the defining inequality (5). But as will be clarified in corollaries to follow, this critical radius is essentially optimal: we show how the bounds in Theorem 1 lead to minimax-optimal rates for various function classes. The interpretation of Theorem 1 is as follows: if the sum of the step-sizes η_t remains below the threshold defined by (6), applying the gradient update (3) reduces the prediction error. Moreover, note that for Hilbert spaces with a larger kernel complexity, the stopping time $\widehat{T}$ is smaller, since fitting functions in a larger class incurs a greater risk of overfitting.

Finally, the lower bound (c) shows that for large t, running the iterative algorithm beyond the optimal stopping point leads to inconsistent estimators for infinite rank kernels. More concretely, let us suppose that $\widehat{\lambda}_i > 0$ for all $1 \leq i \leq n$. In this case, we have

$$\eta_t \widehat{\mathcal{R}}_K^2(\eta_t^{-1/2}) = \frac{1}{n}\sum_{i=1}^{n} \min(\hat{\lambda}_i \eta_t, 1),$$

which converges to 1 as $t \to$. Consequently, part (c) implies that $\liminf_{t\to\infty} \mathbb{E}[\|f_t - f^*\|_n^2] \geq \frac{\sigma^2}{4}$ as $t \to \infty$, thereby showing the inconsistency of the method.

The statement of Theorem 1 is for the case of fixed design points $\{x_i\}_{i=1}^n$, so that the probability is taken only over the sub-Gaussian noise variables $\{w_i\}_{i=1}^n$ In the case of random design point $x_i \sim \mathbb{P}$ i.i.d. , we can also provide bounds on generalization error in the form the $L^2(\mathbb{P})$-norm $\|f_t - f^*\|_2$. In this setting, for the purposes of comparing to minimax lower bounds, it is also useful to state some results in terms of the population analog of the local empirical Rademacher complexity (4), namely the quantity

$$\mathcal{R}_{\mathbb{K}}(\varepsilon) := \left[\frac{1}{n}\sum_{j=1}^{\infty} \min\left\{\lambda_j, \varepsilon^2\right\}\right]^{1/2}, \tag{7}$$

where λ_j correspond to the eigenvalues of the population kernel $\mathbb{K}$ defined in (3). Using this complexity measure, we define the *critical population rate* ε_n to be the smallest positive solution to the inequality

$$40\,\mathcal{R}_{\mathbb{K}}(\varepsilon) \leq \frac{\varepsilon^2}{\sigma}. \tag{8}$$

(Our choice of the pre-factor 40 is for later theoretical convenience.) In contrast to the critical empirical rate $\widehat{\varepsilon}_n$, this quantity is not data-dependent, since it is specified by the population eigenvalues of kernel operator underlying the RKHS.

Theorem 2 (Random design) *Suppose that in addition to the conditions of Theorem 1, the design variables $\{x_i\}_{i=1}^n$ are sampled i.i.d. according to $\mathbb{P}$ and that the population critical radius ε_n satisfies inequality* (8)*. Then there are universal constants $c_j, j = 1, 2, 3$ such that*

$$\|f_{\widehat{T}} - f^*\|_2^2 \leq c_3 \varepsilon_n^2$$

with probability at least $1 - c_1 \exp(-c_2 n \varepsilon_n^2)$.

Theorems 1 and 2 are general results that apply to any reproducing kernel Hilbert space. Their proofs involve combination of direct analysis of our iterative update (3), combined with techniques from empirical process theory and concentration of measure (van de Geer, 2000; Ledoux, 2001); see Section 4 for the details.

It is worthwhile to compare with the past work of Buhlmann and Yu (2003) (hereafter BY), who also provide some theory for gradient descent, referred to as L^2-boosting in their paper, but focusing exclusively on the fixed design case. Our theory applies to random as well as fixed design, and a broader set of stepsize choices. The most significant difference between Theorem 1 in our paper and Theorem 3 of BY is that we provide a data-dependent stopping rule, whereas their analysis does not lead to a stopping rule that can be computed from the data.

3.2 Consequences for Specific Kernel Classes

Let us now illustrate some consequences of our general theory for special choices of kernels that are of interest in practice.

3.2.1 Kernels with Polynomial Eigendecay

We begin with the class of RKHSs whose eigenvalues satisfy a *polynomial decay condition*, meaning that

$$\lambda_k \leq C\big(\frac{1}{k}\big)^{2\beta} \qquad \text{for some } \beta > 1/2 \text{ and constant } C. \tag{9}$$

Among other examples, this type of scaling covers various types of Sobolev spaces, consisting of functions with β derivatives (Birman and Solomjak, 1967; Gu, 2002). As a very special case, the first-order Sobolev kernel $\mathbb{K}(x, x') = \min\{x, x'\}$ on the unit square $[0, 1] \times [0, 1]$ generates an RKHS of functions that are differentiable almost everywhere, given by

$$\mathcal{H} := \big\{f : [0, 1] \to \mathbb{R} \mid \; f(0) = 0, \quad \int_0^1 (f'(x))^2 dx < \infty\big\}, \tag{10}$$

For the uniform measure on $[0, 1]$, this class exhibits polynomial eigendecay (9) with $\beta = 1$. For any class that satisfies the polynomial decay condition, we have the following corollary:

Corollary 3 *Suppose that in addition to the assumptions of Theorem 2, the kernel class $\mathcal{H}$ satisfies the polynomial eigenvalue decay* (9) *for some parameter $\beta > 1/2$. Then there is a universal constant c_5 such that*

$$\mathbb{E}\big[\|f_{\widehat{T}} - f^*\|_2^2\big] \leq c_5 \big(\frac{\sigma^2}{n}\big)^{\frac{2\beta}{2\beta+1}}. \tag{11}$$

Moreover, if $\lambda_k \geq c\,(1/k)^{2\beta}$ *for all* $k = 1, 2, \ldots$, *then*

$$\mathbb{E}\big[\|f_t - f^*\|_2^2\big] \geq \frac{1}{4}\min\big\{1,\ \sigma^2\frac{(\eta_t)^{\frac{1}{2\beta}}}{n}\big\} \quad \textit{for all iterations } t = 1, 2, \ldots.$$

The proof, provided in Section 4.3, involves showing that the population critical rate (7) is of the order $\mathcal{O}(n^{-\frac{2\beta}{2\beta+1}})$. By known results on non-parametric regression (Stone, 1985; Yang and Barron, 1999), the error bound (11) is minimax-optimal.

In the special case of the first-order spline family (10), Corollary 3 guarantees that

$$\mathbb{E}[\|f_{\widehat{T}} - f^*\|_2^2] \precsim \big(\frac{\sigma^2}{n}\big)^{2/3}. \tag{12}$$

In order to test the accuracy of this prediction, we performed the following set of simulations. First, we generated samples from the observation model

$$y_i = f^*(x_i) + w_i, \qquad \text{for } i = 1, 2, \ldots, n, \tag{13}$$

where $x_i = i/n$, and $w_i \sim N(0, \sigma^2)$ are i.i.d. noise terms. We present results for the function $f^*(x) = |x - 1/2| - 1/2$, a piecewise linear function belonging to the first-order Sobelev class. For all our experiments, the noise variance σ^2 was set to one, but so as to have a data-dependent method, this knowledge was not provided to the estimator. There is a large body of work on estimating the noise variance σ^2 in non-parametric regression. For our simulations, we use a simple method due to Hall and Marron (1990). They proved that their estimator is ratio consistent, which is sufficient for our purposes.

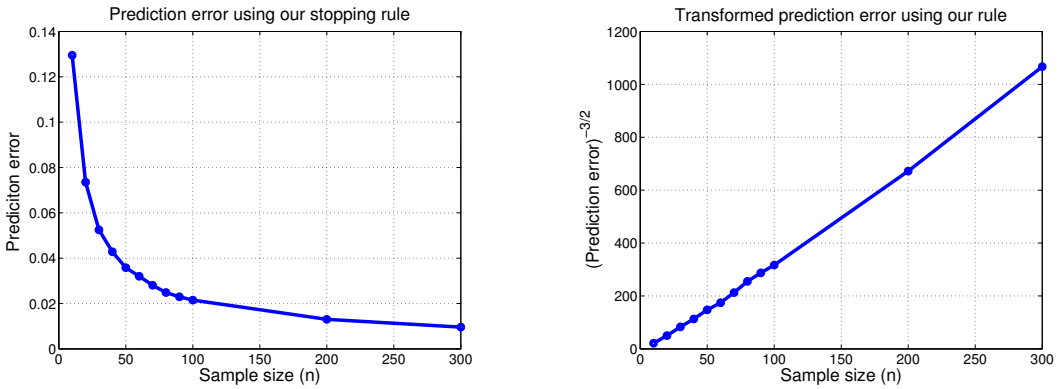
For a range of sample sizes n between 10 and 300, we performed the updates (3) with constant stepsize $\alpha = 0.25$, stopping at the specified time $\widehat{T}$. For each sample size, we performed 10,000 independent trials, and averaged the resulting prediction errors. In panel (a) of Figure 2, we plot the mean-squared error versus the sample size, which shows consistency of the method. The bound (12) makes a more specific prediction: the mean-squared error raised to the power $-3/2$ should scale linearly with the sample size. As shown in panel (b) of Figure 2, the simulation results do indeed reveal this predicted linear relationship. We also performed the same experiments for the case of randomly drawn designs $x_i \sim \text{Unif}(0, 1)$. In this case, we observed similar results, but with more trials required to average out the additional randomness in the design.

3.2.2 Finite Rank Kernels

We now turn to the class of RKHSs based on finite-rank kernels, meaning that there is some finite integer $m < \infty$ such that $\lambda_j = 0$ for all $j \geq m + 1$. For instance, the kernel function $\mathbb{K}(x, x') = (1 + xx')^2$ is a finite rank kernel with $m = 2$, and it generates the RKHS of all quadratic functions. More generally, for any integer $d \geq 2$, the kernel $\mathbb{K}(x, x') = (1 + xx')^d$ generates the RKHS of all polynomials with degree at most d. For any such kernel, we have the following corollary:

Corollary 4 *If, in addition to the conditions of Theorem 2, the kernel has finite rank* m, *then*

$$\mathbb{E}\big[\|\widehat{f}_{\widehat{T}} - f^*\|_2^2\big] \leq c_5\,\sigma^2\frac{m}{n}.$$



(a) (b)

Figure 2: Prediction error obtained from the stopping rule (6) applied to a regression model with n samples of the form $f^*(x_i) + w_i$ at equidistant design points $x_i = i/n$ for $i = 0, 1, \ldots 99$, and i.i.d. Gaussian noise $w_i \sim N(0, 1)$. For these simulations, the true regression function is given by $f^*(x) = |x - \frac{1}{2}| - \frac{1}{2}$. (a) Mean-squared error (MSE) using the stopping rule (6) versus the sample size n. Each point is based on $10,000$ independent realizations of the noise variables $\{w_i\}_{i=1}^n$. (b) Plots of the quantity $MSE^{-3/2}$ versus sample size n. As predicted by the theory, this form of plotting yields a straight line.

For any rank m-kernel, the rate $\frac{m}{n}$ is minimax optimal in terms of squared $L^2(\mathbb{P})$ error; this fact follows as a consequence of more general lower bounds due to Raskutti et al. (2012).

3.3 Comparison with Other Stopping Rules

In this section, we provide a comparison of our stopping rule to two other stopping rules, as well as a oracle method that involves knowledge of f^*, and so cannot be computed in practice.

3.3.1 Hold-out Method

We begin by comparing to a simple hold-out method that performs gradient descent using 50% of the data, and uses the other 50% of the data to estimate the risk. In more detail, assuming that the sample size is even for simplicity, we split the full data set $\{x_i\}_{i=1}^n$ into two equally sized subsets S_{tr} and S_{te}. The data indexed by the training set S_{tr} is used to estimate the function $f_{\text{tr,t}}$ using the gradient descent update (3). At each iteration $t = 0, 1, 2, \ldots$, the data indexed by S_{te} is used to estimate the risk via $R_{\text{HO}}(f_t) = \frac{1}{n} \sum_{i \in S_{te}} \big(y_i - f_{\text{tr,t}}(x_i)\big)^2$, which defines the stopping rule

$$\widehat{T}_{\text{HO}} := \arg\min \Big\{ t \in \mathbb{N} \mid R_{\text{HO}}(f_{\text{tr},\, t+1}) > R_{\text{HO}}(f_{\text{tr,t}}) \Big\} - 1. \tag{14}$$

A line of past work (Yao et al., 2007; Bauer et al., 2007; Caponneto, 2006; Caponetto and Yao, 2006, 2010; Vito et al., 2010) has analyzed stopping rules based on this type of hold-out rule. For instance, Caponneto (2006) analyzes a hold-out method, and shows that it yields rates that are optimal for Sobolev spaces with $\beta \leq 1$ but not in general. A major drawback of using a hold-out rule is that it "wastes" a constant fraction of the data, thereby leading to inflated mean-squared error.

3.3.2 SURE Method

Alternatively, we can use Stein's Unbiased Risk estimate (SURE) to define another stopping rule. Gradient descent is based on the shrinkage matrix $\tilde{S}_t = \prod_{\tau=0}^{t-1}(I - \alpha_\tau K)$. Based on this fact, it can be shown that the SURE estimator (Stein, 1981) takes the form

$$R_{\text{SU}}(f_t) = \frac{1}{n}\{n\sigma^2 + (y_1^n)^T(\tilde{S}_t)^2 y_1^n - 2\sigma^2 \operatorname{trace}(\tilde{S}_t)\}.$$

This risk estimate can be used to define the associated stopping rule

$$\widehat{T}_{\text{SU}} := \arg\min\left\{t \in \mathbb{N} \mid R_{\text{SU}}(f_{t+1}) > R_{\text{SU}}(f_t)\right\} - 1. \tag{15}$$

In contrast with hold-out, the SURE stopping rule (15) makes use of all the data. However, we are not aware of any theoretical guarantees for early stopping based on the SURE rule.

For any valid sequence of stepsizes, it can be shown that both stopping rules (14) and (15) define a unique stopping time. Note that our stopping rule $\widehat{T}$ based on (6) requires estimation of both the empirical eigenvalues, and the noise variance σ^2. In contrast, the SURE-based rule requires estimation of σ^2 but not the empirical eigenvalues, whereas the hold-out rule requires no parameters to be estimated, but a percentage of the data is used to estimate the risk.

3.3.3 Oracle Method

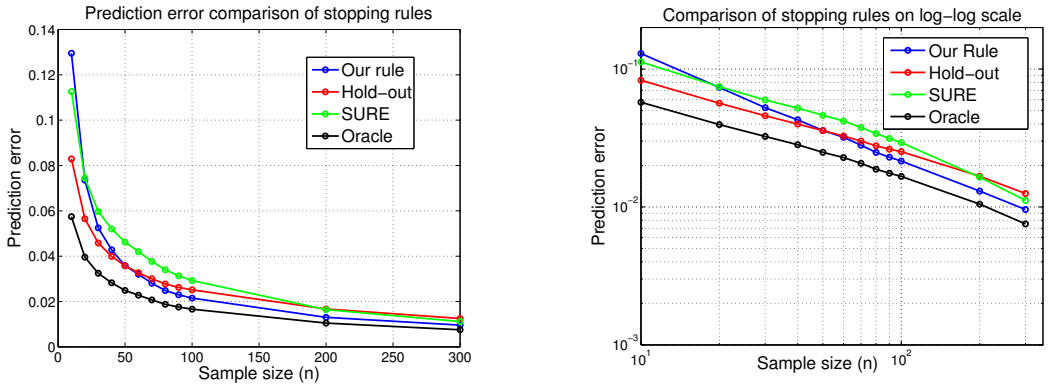
As a third point of reference, we also plot the mean-squared error for an "oracle" method. It is allowed to base its stopping time on the exact in-sample prediction error $R_{\text{OR}}(f_t) = \|f_t - f^*\|_n^2$, which defines the oracle stopping rule

$$\widehat{T}_{\text{OR}} := \arg\min\left\{t \in \mathbb{N} \mid R_{\text{OR}}(f_{t+1}) > R_{\text{OR}}(f_t)\right\} - 1. \tag{16}$$

Note that this stopping rule is not computable from the data, since it assumes exact knowledge of the function f^* that we are trying to estimate.

In order to compare our stopping rule (6) with these alternatives, we generated i.i.d. samples from the previously described model (see Equation (13) and the following discussion). We varied the sample size n from 10 to 300, and for each sample size, we performed $M = 10,000$ independent trials (randomizations of the noise variables $\{w_i\}_{i=1}^n$), and computed the average of squared prediction error over these M trials.

Figure 3 compares the resulting mean-squared errors of our stopping rule (6), the hold-out stopping rule (14), the SURE-based stopping rule (15), and the oracle stopping rule (16). Panel (a) shows the mean-squared error versus sample size, whereas panel (b) shows the



(a) (b)

Figure 3: Illustration of the performance of different stopping rules for kernel gradient descent with the kernel $\mathbb{K}(x,x) = \min\{|x|, |x'|\}$ and noisy samples of the function $f^*(x) = |x - \frac{1}{2}| - \frac{1}{2}$. In each case, we applied the gradient update (3) with constant stepsizes $\alpha_t = 1$ for all t. Each curve corresponds to the mean-squared error, estimated by averaging over $M = 10,000$ independent trials, versus the sample size for $n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300\}$. Each panel shows MSE curves for four different stopping rules: (i) the stopping rule (6); (ii) holding out 50% of the data and using (14); (iii) the SURE stopping rule (15); and (iv) the oracle stopping rule (14). (a) MSE versus sample size on a standard scale. (b) MSE versus sample size on a log-log scale.

same curves in terms of logarithm of mean-squared error. Our proposed rule exhibits better performance than the hold-out and SURE-based rules for sample sizes n larger than 50. On the flip side, since the construction of our stopping rule is based on the assumption that f^* belongs to a known RKHS, it is unclear how robust it would be to model mis-specification. In contrast, the hold-out and SURE-based stopping rules are generic methods, not based directly on the RKHS structure, so might be more robust to model mis-specification. Thus, one interesting direction is to explore the robustness of our stopping rule. On the theoretical front, it would be interesting to determine whether the hold-out and/or SURE-based stopping rules can be proven to achieve minimax optimal rates for general kernels, as we have established for our stopping rule.

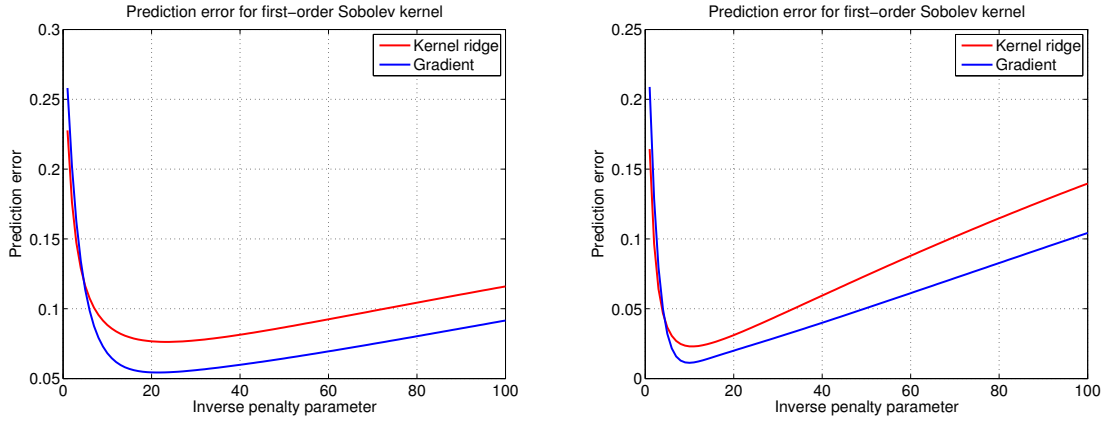
3.4 Connections to Kernel Ridge Regression

We conclude by presenting an interesting link between our early stopping procedure and kernel ridge regression. The kernel ridge regression (KRR) estimate is defined as

$$\widehat{f}_\nu := \arg\min_{f \in \mathcal{H}} \Big\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \frac{1}{2\nu} \|f\|_{\mathcal{H}}^2 \Big\}, \tag{17}$$

where ν is the (inverse) regularization parameter. For any $\nu < \infty$, the objective is strongly convex, so that the KRR solution is unique.

Friedman and Popescu (2004) observed through simulations that the regularization paths for early stopping of gradient descent and ridge regression are similar, but did not provide any theoretical explanation of this fact. As an illustration of this empirical phenomenon, Figure 4 compares the prediction error $\|\widehat{f}_\nu - f^*\|_n^2$ of the kernel ridge regression estimate over the interval $\nu \in [1, 100]$ versus that of the gradient update (3) over the first 100 iterations. Note that the curves, while not identical, are qualitatively very similar.



(a) (b)

Figure 4: Comparison of the prediction error of the path of kernel ridge regression estimates (17) obtained by varying $\nu \in [1, 100]$ to those of the gradient updates (3) over 100 iterations with constant step size. All simulations were performed with the kernel $\mathbb{K}(x, x') = \min\{|x|, |x'|\}$ based on $n = 100$ samples at the design points $x_i = i/n$ with $f^*(x) = |x - \frac{1}{2}| - \frac{1}{2}$. (a) Noise variance $\sigma^2 = 1$. (b) Noise variance $\sigma^2 = 2$.

From past theoretical work (van de Geer, 2000; Mendelson, 2002), kernel ridge regression with the appropriate setting of the penalty parameter ν is known to achieve minimax-optimal error for various kernel classes. These classes include the Sobolev and finite-rank kernels for which we have previously established that our stopping rule (6) yields optimal rates. In this section, we provide a theoretical basis for these connections. More precisely, we prove that if the inverse penalty parameter ν is chosen using the same criterion as our stopping rule, then the prediction error satisfies the same type of bounds, with ν now playing the role of the running sum η_t.

Define $\widehat{\nu} > 0$ to be the smallest positive solution to the inequality

$$\big(4\sigma\nu\big)^{-1} < \widehat{\mathcal{R}}_K(1/\sqrt{\nu}). \tag{18}$$

Note that this criterion is identical to the one underlying our stopping rule, except that the continuous parameter ν replaces the discrete parameter $\eta_t = \sum_{\tau=0}^{t-1} \alpha_\tau$.

Proposition 5 *Consider the kernel ridge regression estimator* (17) *applied to n i.i.d. samples $\{(x_i, y_i)\}$ with σ-sub Gaussian noise. Then there are universal constants (c_1, c_2, c_3) such that with probability at least $1 - c_1 \exp(-c_2\, n\, \widehat{\varepsilon}_n^2)$:*

(a) For all $0 < \nu \leq \widehat{\nu}$, we have

$$\|\widehat{f}_\nu - f^*\|_n^2 \leq \frac{2}{\nu}$$

(b) With $\widehat{\nu}$ chosen according to the rule (18)*, we have*

$$\|\widehat{f}_{\widehat{\nu}} - f^*\|_n^2 \leq c_3\, \widehat{\varepsilon}_n^2.$$

(c) Moreover, for all $\nu > \widehat{\nu}$, we have

$$\mathbb{E}[\|\widehat{f}_\nu - f^*\|_n^2] \geq \frac{\sigma^2}{4} \nu \widehat{\mathcal{R}}_K^2(\nu^{-1/2}).$$

Note that apart from a slightly different leading constant, the upper bound (a) is *identical* to the upper bound in Theorem 1 part (a). The only difference is that the inverse regularization parameter ν replaces the running sum $\eta_t = \sum_{\tau=0}^{t-1} \alpha_\tau$. Similarly, part (b) of Proposition 5 guarantees that the kernel ridge regression (17) has prediction error that is upper bounded by the empirical critical rate $\widehat{\varepsilon}_n^2$, as in part (b) of Theorem 1. Let us emphasize that bounds of this type on kernel ridge regression have been derived in past work (Mendelson, 2002; Zhang, 2005; van de Geer, 2000). The novelty here is that the structure of our result reveals the intimate connection to early stopping, and in fact, the proofs follow a parallel thread.

In conjunction, Proposition 5 and Theorem 1 provide a theoretical explanation for why, as shown in Figure 4, the paths of the gradient descent update (3) and kernel ridge regression estimate (17) are so similar. However, it is important to emphasize that from a computational point of view, early stopping has certain advantages over kernel ridge regression. In general, solving a quadratic program of the form (17) requires on the order of $\mathcal{O}(n^3)$ basic operations, and this must be done repeatedly for each new choice of ν. On the other hand, by its very construction, the iterates of the gradient algorithm correspond to the desired path of solutions, and each gradient update involves multiplication by the kernel matrix, incurring $\mathcal{O}(n^2)$ operations.

4. Proofs

We now turn to the proofs of our main results. The main steps in each proof are provided in the main text, with some of the more technical results deferred to the appendix.

4.1 Proof of Theorem 1

In order to derive upper bounds on the $L^2(\mathbb{P}_n)$-error in Theorem 1, we first rewrite the gradient update (3) in an alternative form. For each iteration $t = 0, 1, 2, \ldots$, let us introduce the shorthand

$$f_t(x_1^n) := \begin{bmatrix} f_t(x_1) & f_t(x_2) & \cdots & f_t(x_n) \end{bmatrix} \in \mathbb{R}^n,$$

corresponding to the n-vector obtained by evaluating the function f^t at all design points, and the short-hand

$$w := \big[w_1, w_2, ..., w_n\big] \in \mathbb{R}^n,$$

corresponding to the vector of zero mean sub-Gaussian noise random variables. From Equation (2), we have the relation

$$f^t(x_1^n) = \frac{1}{\sqrt{n}} K\, \omega^t \;=\; \frac{1}{\sqrt{n}} \sqrt{K}\, \theta_t.$$

Consequently, by multiplying both sides of the gradient update (3) by $\sqrt{K}$, we find that the sequence $\{f_t(x_1^n)\}_{t=0}^{\infty}$ evolves according to the recursion

$$f_{t+1}(x_1^n) = f_t(x_1^n) - \alpha_t K\,(f_t(x_1^n) - y_1^n) \;=\; \Big(I_{n\times n} - \alpha_t K\Big) f_t(x_1^n) + \alpha_t K\, y_1^n. \tag{19}$$

Since $\theta_0 = 0$, the sequence is initialized with $f_0(x_1^n) = 0$. The recursion (19) lies at the heart of our analysis.

Letting $r = \operatorname{rank}(K)$, the empirical kernel matrix has the eigendecomposition $K = U\Lambda U^T$, where $U \in \mathbb{R}^{n\times n}$ is an orthonormal matrix (satisfying $UU^T = U^TU = I_{n\times n}$) and

$$\Lambda := \operatorname{diag}(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_r, 0, 0, \cdots, 0)$$

is the diagonal matrix of eigenvalues, augmented with $n-r$ zero eigenvalues as needed. We then define a sequence of diagonal *shrinkage matrices* S^t as follows:

$$S^t := \prod_{\tau=0}^{t-1} (I_{n\times n} - \alpha_\tau \Lambda) \in \mathbb{R}^{n\times n}.$$

The matrix S^t indicates the extent of shrinkage towards the origin; since $0 \;\leq\; \alpha_t \;\leq \min\{1, 1/\widehat{\lambda}_1\}$ for all iterations t, in the positive semodefinite ordering, we have the sandwich relation

$$0 \preceq S^{t+1} \preceq S^t \preceq I_{n\times n}.$$

Moreover, the following lemma shows that the $L^2(\mathbb{P}_n)$-error at each iteration can be bounded in terms of the eigendecomposition and these shrinkage matrices:

Lemma 6 (Bias/variance decomposition) *At each iteration $t = 0, 1, 2, \ldots$,*

$$\|f_t - f^*\|_n^2 \leq \underbrace{\frac{2}{n}\sum_{j=1}^{r} (S^t)_{jj}^2 [U^T f^*(x_1^n)]_j^2 + \frac{2}{n}\sum_{j=r+1}^{n} [U^T f^*(x_1^n)]_j^2}_{\textit{Squared Bias } B_t^2} + \underbrace{\frac{2}{n}\sum_{j=1}^{r} (1 - S_{jj}^t)^2 [U^T w]_j^2}_{\textit{Variance } V_t}. \tag{20}$$

Moreover, we have the lower bound $\mathbb{E}[\|f_t - f^*\|_n^2] \geq \mathbb{E}[V_t]$.

See Appendix B.1 for the proof of this intermediate claim.

In order to complete the proof of the upper bound in Theorem 1, our next step is to obtain high probability upper bounds on these two terms. We summarize our conclusions in an additional lemma, and use it to complete the proof of Theorem 1(a) before returning to prove it.

Lemma 7 (Bounds on the bias and variance) *For all iterations $t = 1, 2, \ldots$, the squared bias is upper bounded as*

$$B_t^2 \leq \frac{1}{e\,\eta_t}, \tag{21}$$

Moreover, there is a universal constant $c_1 > 0$ such that, for any iteration $t = 1, 2, \ldots, \widehat{T}$,

$$V_t \;\leq\; 5\sigma^2\,\eta_t \mathcal{R}_K^2\big(1/\sqrt{\eta_t}\big) \tag{22}$$

with probability at least $1 - \exp\big(-c_1\, n\widehat{\varepsilon}_n^2\big)$. Moreover we have $\mathbb{E}[V_t] \geq \frac{\sigma^2}{4}\,\eta_t \mathcal{R}_K^2\big(1/\sqrt{\eta_t}\big)$.

We can now complete the proof of Theorem 1(a). Conditioned on the event $V_t \leq 5\sigma^2 \eta_t \mathcal{R}_K^2\big(1/\sqrt{\eta_t}\big)$, we have

$$\|f_t - f^*\|_n^2 \overset{(i)}{\leq} B_t^2 + V_t \overset{(ii)}{\leq} \frac{1}{e\,\eta_t} + 5\sigma^2\,\eta_t \mathcal{R}_K^2\big(1/\sqrt{\eta_t}\big) \overset{(iii)}{\leq} \frac{4}{e\,\eta_t},$$

where inequality (i) follows from (20) in Lemma 6, and inequality (ii) follows from the bounds in Lemma 7 and (iii) follows since $t \leq \widehat{T}$. The lower bound (c) follows from (22).

Turning to the proof of part (b), using the upper bound from (a)

$$\|f_{\widehat{T}} - f^*\|_n^2 \leq \frac{1}{e\,\eta_{\widehat{T}}} + \frac{5}{\eta_{\widehat{T}}} \;\leq\; \frac{4}{e\eta_{\widehat{T}}}.$$

Based on the definition of $\widehat{T}$ and $\widehat{\varepsilon}_n$, we are guaranteed that $\frac{1}{\eta_{\widehat{T}+1}} \leq \widehat{\varepsilon}_n^2$, Moreover, by the non-decreasing nature of our step sizes, we have $\alpha_{\widehat{T}+1} \leq \alpha_{\widehat{T}}$, which implies that $\eta_{\widehat{T}+1} \leq 2\eta_{\widehat{T}}$, and hence

$$\frac{1}{\eta_{\widehat{T}}} \leq \frac{2}{\eta_{\widehat{T}+1}} \;\leq\; 2\widehat{\varepsilon}_n^2.$$

Putting together the pieces establishes the bound claimed in part (b).

It remains to establish the bias and variance bounds stated in Lemma 7, and we do so in the following subsections. The following auxiliary lemma plays a role in both proofs:

Lemma 8 (Properties of shrinkage matrices) *For all indices $j \in \{1, 2, \ldots, r\}$, the shrinkage matrices S^t satisfy the bounds*

$$0 \leq\; (S^t)_{jj}^2 \leq \frac{1}{2e\eta_t\widehat{\lambda}_j}, \quad \textit{and} \tag{23}$$

$$\frac{1}{2}\min\{1, \eta_t\widehat{\lambda}_j\} \leq 1 - S_{jj}^t \;\leq \min\{1, \eta_t\widehat{\lambda}_j\}. \tag{24}$$

See Appendix B.2 for the proof of this result.

4.1.1 Bounding the Squared Bias

Let us now prove the upper bound (21) on the squared bias. We bound each of the two terms in the definition (20) of B_t^2 in term. Applying the upper bound (23) from Lemma 8, we see that

$$\frac{2}{n}\sum_{j=1}^{r}(S^t)_{jj}^2[U^T f^*(x_1^n)]_j^2 \leq \frac{1}{e\ n\ \eta_t}\ \sum_{j=1}^{r}\frac{[U^T f^*(x_1^n)]_j^2}{\widehat{\lambda}_j}.$$

Now consider the linear operator $\Phi_X : \ell^2(\mathbb{N}) \to \mathbb{R}^n$ defined element-wise via $[\Phi_X]_{jk} = \phi_j(x_k)$. Similarly, we define a (diagonal) linear operator $D : \ell^2(\mathbb{N}) \to \ell^2(\mathbb{N})$ with entries $[D]_{jj} = \lambda_j$ and $[D]_{jk} = 0$ for $j \neq k$. With these definitions, the vector $f(x_1^n) \in \mathbb{R}^n$ can be expressed in terms of some sequence $a \in \ell^2(\mathbb{N})$ in the form

$$f(x_1^n) = \Phi_X D^{1/2} a.$$

In terms of these quantities, we can write $K = \frac{1}{n}\Phi_X D \Phi_X^T$. Moreover, as previously noted, we also have $K = U\Lambda U^T$ where $\Lambda = \operatorname{diag}\{\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n\}$, and $U \in \mathbb{R}^{n\times n}$ is orthonormal. Combining the two representations, we conclude that

$$\frac{\Phi_X D^{1/2}}{\sqrt{n}} = U\Lambda^{1/2}\Psi^*,$$

for some linear operator $\Psi : \mathbb{R}^n \to \ell^2(\mathbb{N})$ (with adjoint Ψ^*) such that $\Psi^*\Psi = I_{n\times n}$. Using this equality, we have

$$\begin{aligned}
\frac{1}{e\,\eta_t\,n}\sum_{j=1}^{r}\frac{[U^T f^*(X)]_j^2}{\widehat{\lambda}_j} &= \frac{1}{e\,\eta_t\,n}\ \sum_{j=1}^{r}\frac{[U^T\Phi_X D^{1/2}a]_j^2}{\widehat{\lambda}_j} \\
&= \frac{1}{e\,\eta_t}\ \sum_{j=1}^{r}\frac{[U^T U\Lambda^{1/2}V^* a]_j^2}{\widehat{\lambda}_j} \\
&= \frac{1}{e\,\eta_t}\ \sum_{j=1}^{r}\frac{\widehat{\lambda}_j\,[\Psi^* a]_j^2}{\widehat{\lambda}_j} \\
&\leq \frac{1}{e\,\eta_t}\ \|\Psi^* a\|_2^2 \\
&\leq \frac{1}{e\,\eta_t}, \qquad (25)
\end{aligned}$$

Here the final step follows from the fact that Ψ is a unitary operator, so that $\|\Psi^* a\|_2^2 \leq \|a\|_2^2 = \|f^*\|_{\mathcal{H}}^2 \leq 1$.

Turning to the second term in the definition (20), we have

$$\begin{aligned}\sum_{j=r+1}^{n} [U^T f^*(x_1^n)]_j^2 &= \frac{2}{n} \sum_{j=r+1}^{n} [U^T \Phi_X D^{1/2} a]_j^2 \\ &= \sum_{j=r+1}^{n} [U^T U \Lambda^{1/2} \Psi^* a]_j^2 \\ &= \sum_{j=r+1}^{n} [\Lambda^{1/2} \Psi^* a]_j^2 \\ &= 0, \end{aligned} \tag{26}$$

where the final step uses the fact that $\Lambda_{jj}^{1/2} = 0$ for all $j \in \{r+1, \ldots, n\}$ by construction. Combining the upper bounds (25) and (26) with the definition (20) of B_t^2 yields the claim (21).

4.1.2 Controlling the Variance

Let us now prove the bounds (22) on the variance term V_t. (To simplify the proof, we assume throughout that $\sigma = 1$; the general case can be recovered by a simple rescaling argument). By the definition of V_t, we have

$$V_t = \frac{2}{n} \sum_{j=1}^{r} (1 - S_{jj}^t)^2 [U^T w]_j^2 \;=\; \frac{2}{n} \operatorname{trace}(U Q U^T \, w w^T),$$

where $Q = \operatorname{diag}\{(1 - S_{jj}^t)^2, \; j = 1, \ldots, n\}$ is a diagonal matrix. Since $\mathbb{E}[ww^T] \leq I_{n \times n}$ by assumption, we have $\mathbb{E}[V_t] = \frac{2}{n} \operatorname{trace}(Q)$. Using the upper bound in Equation (24) from Lemma 8, we have

$$\frac{1}{n} \operatorname{trace}(Q) \leq \frac{1}{n} \sum_{j=1}^{r} \min\{1, (\eta_t \widehat{\lambda}_j)^2\} \;=\; \eta_t \left(\mathcal{R}_K(1/\sqrt{\eta_t}) \right)^2,$$

where the final equality uses the definition of $\mathcal{R}_K$. Putting together the pieces, we see that

$$\mathbb{E}[V_t] \leq 2 \, \eta_t \left(\mathcal{R}_K(1/\sqrt{\eta_t}) \right)^2.$$

Similarly, using the lower bound in Equation (24), we can show that

$$\mathbb{E}[V_t] \geq \frac{\sigma^2}{4} \eta_t \left(\mathcal{R}_K(1/\sqrt{\eta_t}) \right)^2.$$

Our next step is to obtain a bound on the two-sided tail probability $\mathbb{P}[|V_t - \mathbb{E}[V_t]| \geq \delta]$, for which we make use of a result on two-sided deviations for quadratic forms in sub-Gaussian variables. In particular, consider a random variable of the form $Q_n = \sum_{i,j=1}^{n} a_{ij}(Z_i Z_j -$

$\mathbb{E}[Z_i Z_j]$) where $\{Z_i\}_{i=1}^n$ are i.i.d. zero-mean and sub-Gaussian variables (with parameter 1). Wright (1973) proves that there is a constant c such that

$$\mathbb{P}\big[|Q - \mathbb{E}[Q]| \geq \delta\big] \leq \exp\Big(- c \ \min\big\{ \frac{\delta}{|||A|||_{\rm op}}, \frac{\delta^2}{|||A|||_{\rm F}^2} \big\}\Big) \quad \text{for all } u > 0, \tag{27}$$

where $(|||A|||_{\rm op}, |||A|||_{\rm F})$ are (respectively) the operator and Frobenius norms of the matrix $A = \{a_{ij}\}_{i,j=1}^n$.

If we apply this result with $A = \frac{2}{n} U Q U^T$ and $Z_i = w_i$, then we have $Q = V_t$, and moreover

$$|||A|||_{\rm op} \leq \frac{2}{n}, \quad \text{and}$$

$$|||A|||_{\rm F}^2 \ = \ \frac{4}{n^2} \operatorname{trace}(U^T Q U^T U Q U^T) = \frac{4}{n^2} \operatorname{trace}(Q^2) \leq \frac{4}{n^2} \operatorname{trace}(Q) \leq \frac{4}{n} \eta_t \left(\mathcal{R}_K(1/\sqrt{\eta_t}) \right).$$

Consequently, the bound (27) implies that

$$\mathbb{P}\big[|V_t - \mathbb{E}[V_t]| \geq \delta\big] \leq \exp\big(- 4c\, n\, \delta \min\{1, \delta \Big(\eta_t \mathcal{R}_K(1/\sqrt{\eta_t}) \Big)^{-1} \}\big).$$

Since $t \leq \widehat{T}$ setting $\delta = 3\sigma^2 \eta_t \left(\mathcal{R}_K(1/\sqrt{\eta_t}) \right)$, the claim (22) follows.

4.2 Proof of Theorem 2

This proof is based on the following two steps:

- first, proving that the error $\|f_{\widehat{T}} - f^*\|_2$ in the $L^2(\mathbb{P})$ norm is, with high probability, close to the error in the $L^2(\mathbb{P}_n)$ norm, and
- second, showing the empirical critical radius $\widehat{\varepsilon}_n$ defined in Equation (5) is upper bounded by the population critical radius ε_n defined in Equation (8).

Our proof is based on a number of more technical auxiliary lemmas, proved in the appendices. The first lemma provides a high probability bound on the Hilbert norm of the estimate $f_{\widehat{T}}$.

Lemma 9 *There exist universal constants c_1 and $c_2 > 0$ such that $\|f_t\|_{\mathcal{H}} \leq 2$ for all $t \leq \widehat{T}$ with probability greater than or equal to $1 - c_1 \exp(-c_2 n \widehat{\varepsilon}_n^2)$.*

See Appendix E.1 for the proof of this claim. Our second lemma shows in any bounded RKHS, the $L^2(\mathbb{P})$ and $L^2(\mathbb{P}_n)$ norms are uniformly close up to the population critical radius ε_n over a Hilbert ball of constant radius:

Lemma 10 *Consider a Hilbert space such that $\|g\|_\infty \leq B$ for all $g \in \mathbb{B}_{\mathcal{H}}(3)$. Then there exist universal constants (c_1, c_2, c_3) such that for any $t \geq \varepsilon_n$, we have*

$$|\|g\|_n^2 - \|g\|_2^2| \ \leq \ c_1 t^2,$$

with probability at least $1 - c_2 \exp(-c_3 n t^2)$.

This claim follows from known results on reproducing kernel Hilbert spaces (e.g., Lemma 5.16 in the paper van de Geer, 2000 and Theorem 2.1 in the paper Bartlett et al., 2005). Our final lemma, proved in Appendix E.2, relates the critical empirical radius $\widehat{\varepsilon}_n$ to the population radius ε_n:

Lemma 11 *There exist constants c_1 and c_2 such that $\widehat{\varepsilon}_n \leq \varepsilon_n$ holds with probability at least $1 - c_1 \exp(-c_2 n \varepsilon_n^2)$.*

With these lemmas in hand, the proof of the theorem is straightforward. First, from Lemma 9, we have $\|f_{\widehat{T}}\|_{\mathcal{H}} \leq 2$ and hence by triangle inequality, $\|f_{\widehat{T}} - f^*\|_{\mathcal{H}} \leq 3$ with high probability as well. Next, applying Lemma 10 with $t = \varepsilon_n$, we find that

$$\|f_{\widehat{T}} - f^*\|_2^2 \leq \|f_{\widehat{T}} - f^*\|_n^2 + c_1 \varepsilon_n^2 \leq c_4(\widehat{\varepsilon}_n^2 + \varepsilon_n^2),$$

with probability greater than $1 - c_2 \exp(-c_3 n \varepsilon_n^2)$. Finally, applying Lemma 11 yields that the bound $\|f_{\widehat{T}} - f^*\|_2^2 \leq c \varepsilon_n^2$ holds with the claimed probability.

4.3 Proof of Corollaries

In each case, it suffices to upper bound the generalization rate ε_n^2 previously defined.

4.3.1 Proof of Corollary 4

In this case, we have

$$\mathcal{R}_{\mathbb{K}}(\epsilon) = \frac{1}{\sqrt{n}} \sqrt{\sum_{j=1}^{m} \min\{\lambda_j, \epsilon^2\}} \;\leq\; \sqrt{\frac{m}{n}}\, \epsilon$$

so that $\varepsilon_n^2 = c' \sigma^2 \frac{m}{n}$.

4.3.2 Proof of Corollary 3

For any $M \geq 1$, we have

$$\begin{aligned} \mathcal{R}_{\mathbb{K}}(\epsilon) \;=\; \frac{1}{\sqrt{n}} \sqrt{\sum_{j=1}^{\infty} \min\{C\, j^{-2\beta}, \epsilon^2\}} &\leq \sqrt{\frac{M}{n}}\epsilon + \sqrt{\frac{C}{n}} \sqrt{\sum_{j=\lceil M \rceil}^{\infty} j^{-2\beta}} \\ &\leq \sqrt{\frac{M}{n}}\epsilon + \sqrt{\frac{C'}{n}} \sqrt{\int_{M}^{\infty} t^{-2\beta} dt} \\ &\leq \sqrt{\frac{M}{n}}\epsilon + C'' \frac{1}{\sqrt{n}} (1/M)^{\beta - \frac{1}{2}}. \end{aligned}$$

Setting $M = \epsilon^{-1/\beta}$ yields $\mathcal{R}_{\mathbb{K}}(\epsilon) \leq C^* \epsilon^{1 - \frac{1}{2\beta}}$. Consequently, the critical inequality $\mathcal{R}_{\mathbb{K}}(\epsilon) \leq 40\epsilon^2/\sigma$ is satisfied for $\varepsilon_n \asymp (\sigma^2/n)^{\frac{2\beta}{2\beta+1}}$, as claimed.

4.4 Proof of Proposition 5

We now turn to the proof of our results on the kernel ridge regression estimate (17). The proof follows a very similar structure to that of Theorem 1. Recall the eigendecomposition $K = U\Lambda U^T$ of the empirical kernel matrix, and that we use r to denote its rank. For each $\nu > 0$, we define the *ridge shrinkage matrix*

$$R^\nu := \big(I_{n\times n} + \nu\Lambda\big)^{-1}. \tag{28}$$

We then have the following analog of Lemma 7 from the proof of Theorem 1:

Lemma 12 (Bias/variance decomposition for kernel ridge regression) *For any $\nu > 0$, the prediction error for the estimate $\widehat{f}_\nu$ is bounded as*

$$\|\widehat{f}_\nu - f^*\|_n^2 \leq \frac{2}{n}\sum_{j=1}^{r}[R^\nu]_{jj}^2[U^T f^*(x_1^n)]_j^2 + \frac{2}{n}\sum_{j=r+1}^{n}[U^T f^*(x_1^n)]_j^2 + \frac{2}{n}\sum_{j=1}^{r}\big(1 - R^\nu_{jj}\big)^2[U^T w]_j^2.$$

Note that Lemma 12 is identical to Lemma 7 with the shrinkage matrices S^t replaced by their analogues R^ν. See Appendix C.1 for the proof of this claim.

Our next step is to show that the diagonal elements of the shrinkage matrices R^ν are bounded:

Lemma 13 (Properties of kernel ridge shrinkage) *For all indices $j \in \{1, 2, \ldots, r\}$, the diagonal entries R^ν satisfy the bounds*

$$\begin{aligned} 0 \;\leq\; (R^\nu_{jj})^2 \leq \frac{1}{4\nu\widehat{\lambda}_j}, \quad \text{and} \\ \frac{1}{2}\min\big\{1, \nu\widehat{\lambda}_j\big\} \;\leq\; 1 - R^\nu_{jj} \;\leq\; \min\big\{1, \nu\widehat{\lambda}_j\big\}. \end{aligned} \tag{29}$$

Note that this is the analog of Lemma 8 from Theorem 1, albeit with the constant $\frac{1}{4}$ in the bound (29) instead of $\frac{1}{2e}$. See Appendix C.2 for the proof of this claim. With these lemmas in place, the remainder of the proof follows as in the proof of Theorem 1.

5. Discussion

In this paper, we have analyzed the early stopping strategy as applied to gradient descent on the non-parametric least squares loss. Our main contribution was to propose an easily computable and data-dependent stopping rule, and to provide upper bounds on the empirical $L^2(\mathbb{P}_n)$ error (Theorem 1) and generlization $L^2(\mathbb{P})$ error (Theorem 2). We demonstrate in Corollaries 3 and 4 that our stopping rule yields minimax optimal rates for both low rank kernel classes and Sobolev spaces. Our simulation results confirm that our stopping rule yields theoretically optimal rates of convergence for Lipschitz kernels, and performs favorably in comparison to stopping rules based on hold-out data and Stein's Unbiased Risk Estimate. We also showed that early stopping with sum of step-sizes $\eta_t = \sum_{k=0}^{t-1}\alpha_k$ has a regularization path that satisfies almost identical mean-squared error bounds as kernel ridge regression indexed by penalty parameter ν.

Our analysis and stopping rule may be improved and extended in a number of ways. First, it would interesting to see how our stopping rule can be adapted to mis-specified models. As specified, our method relies on computation of the eigenvalues of the kernel matrix. A stopping rule based on approximate eigenvalue computations, for instance via some form of sub-sampling (Drineas and Mahoney, 2005), would be interesting to study as well.

Acknowledgments

This work was partially supported by NSF grant DMS-1107000 to MJW and BY. In addition, BY was partially supported by the NSF grant SES-0835531 (CDI), ARO-W911NF-11-1-0114 and the Center for Science of Information (CSoI), an US NSF Science and Technology Center, under grant agreement CCF-0939370, and MJW was also partially supported ONR MURI grant N00014-11-1-086. During this work, GR received partial support from a Berkeley Graduate Fellowship.

Appendix A. Derivation of Gradient Descent Updates

In this appendix, we provide the details of how the gradient descent updates (3) are obtained. In terms of the transformed vector $\theta = \sqrt{K}\,\omega$, the least-squares objective takes the form

$$\widetilde{\mathcal{L}}(\theta) := \frac{1}{2n}\|y_1^n - \sqrt{n}\sqrt{K}\,\theta\|_2^2 \;=\; \frac{1}{2n}\|y_1^n\|_2^2 - \frac{1}{\sqrt{n}}\langle y_1^n,\, \sqrt{K}\,\theta\rangle + \frac{1}{2}(\theta)^T K\theta.$$

Given a sequence $\{\alpha_t\}_{t=0}^{\infty}$, the gradient descent algorithm operates via the recursion $\theta_{t+1} = \theta_t - \alpha_t \nabla\widetilde{\mathcal{L}}(\theta^t)$. Taking the gradient of $\widetilde{\mathcal{L}}$ yields

$$\nabla\widetilde{\mathcal{L}}(\theta) = K\,\theta - \frac{1}{\sqrt{n}}\sqrt{K}\,y_1^n.$$

Substituting into the gradient descent update yields the claim (3).

Appendix B. Auxiliary Lemmas for Theorem 1

In this appendix, we collect together the proofs of the lemmas for Theorem 1.

B.1 Proof of Lemma 6

We prove this lemma by analyzing the gradient descent iteration in an alternative coordinate system. In particular, given a vector $f^t(x_1^n) \in \mathbb{R}^n$ and the SVD $K = U\Lambda U^T$ of the empirical kernel matrix, we define the vector $\gamma^t = \frac{1}{\sqrt{n}}U^T f^t(x_1^n)$. In this new-coordinate system, our goal is to estimate the vector $\gamma^* = \frac{1}{\sqrt{n}}U^T f^*(x_1^n)$. Recalling the alternative form (19) of the gradient recursion, some simple algebra yields that the sequence $\{\gamma^t\}_{t=0}^{\infty}$ evolves as

$$\gamma^{t+1} = \gamma^t + \alpha_t\Lambda\frac{\tilde{w}}{\sqrt{n}} - \alpha_t\Lambda(\gamma^t - \gamma^*),$$

where $\tilde{w} := U^T w$ is a rotated noise vector. Since $\gamma^0 = 0$, unwrapping this recursion then yields $\gamma^t - \gamma^* = \big(I - S^t\big)\frac{\tilde{w}}{\sqrt{n}} - S^t\gamma^*$, where we have made use of the previously defined shrinkage matrices S^t. Using the inequality $\|a+b\|_2^2 \leq 2(\|a\|_2^2 + \|b\|_2^2)$, we find that

$$\begin{aligned}\|\gamma^t - \gamma^*\|_2^2 &\leq \frac{2}{n}\|(I-S^t)\tilde{w}\|_2^2 + 2\|S^t\gamma^*\|_2^2 \\ &= \frac{2}{n}\|(I-S^t)\tilde{w}\|_2^2 + 2\sum_{j=1}^{r}[S^t]_{jj}^2(\gamma_{jj}^*)^2 + 2\sum_{j=r+1}^{n}(\gamma_{jj}^*)^2.\end{aligned}$$

where the equality uses the fact that $\widehat{\lambda}_j = 0$ for all $j \in \{r+1, \ldots, n\}$. Finally, the orthogonality of U implies that $\|\gamma^t - \gamma^*\|_2^2 = \frac{1}{n}\|f^t(x_1^n) - f^*(x_1^n)\|_2^2$, from which the upper bound (20) follows.

B.2 Proof of Lemma 8

Using the definition of S^t and the elementary inequality $1 - u \leq \exp(-u)$, we have

$$[S^t]_{jj}^2 = \Big(\prod_{\tau=0}^{t-1}(1-\alpha_\tau\widehat{\lambda}_j)\Big)^2 \leq \exp(-2\eta_t\widehat{\lambda}_j) \overset{(i)}{\leq} \frac{1}{2e\eta_t\widehat{\lambda}_j},$$

where inequality (i) follows from the fact that $\sup_{u\in\mathbb{R}}\big\{u\exp(-u)\big\} = 1/e$.

Turning to the second set of inequalities, we have $1 - [S^t]_{jj} = 1 - \prod_{\tau=0}^{t-1}(1-\alpha_\tau\widehat{\lambda}_j)$. By induction, it can be shown that

$$1 - [S^t]_{jj} \leq 1 - \max\{0, 1-\eta_t\widehat{\lambda}_j\} = \min\{1, \eta_t\widehat{\lambda}_j\}.$$

As for the remaining claim, we have

$$\begin{aligned}1 - \prod_{\tau=0}^{t-1}(1-\alpha_\tau\widehat{\lambda}_j) &\overset{(i)}{\geq} 1 - \exp(-\eta_t\widehat{\lambda}_i) \\ &\overset{(ii)}{\geq} 1 - (1+\eta_t\widehat{\lambda}_i)^{-1} \\ &= \frac{\eta_t\widehat{\lambda}_i}{1+\eta_t\widehat{\lambda}_i} \\ &\geq \frac{1}{2}\min\{1, \eta_t\widehat{\lambda}_i\},\end{aligned}$$

where step (i) follows from the inequality $1 - u \leq \exp(-u)$; and step (ii) follows from the inequality $\exp(-u) \leq (1+u)^{-1}$, valid for $u > 0$.

Appendix C. Auxiliary Results for Proposition 5

In this appendix, we prove the auxiliary lemmas used in the proof of Proposition 5 on kernel ridge regression.

C.1 Proof of Lemma 12

By definition of the KRR estimate, we have $\left(K + \frac{1}{\nu} I\right) f_\nu(x_1^n) = K y_1^n$. Consequently, some straightforward algebra yields the relation

$$U^T f_\nu(x_1^n) = (I - R^\nu) U^T y_1^n,$$

where the shrinkage matrix R^ν was previously defined (28). The remainder of the proof follows using identical steps to the proof of Lemma 6 with S^t replaced by R^ν.

C.2 Proof of Lemma 13

By definition (28) of the shrinkage matrix, we have $[R^\nu]_{jj}^2 = (1 + \nu\widehat{\lambda}_j)^{-2} \leq \frac{1}{4\nu\widehat{\lambda}_j}$. Moreover, we also have

$$1 - [R^\nu]_{jj} = 1 - (1 + \nu\widehat{\lambda}_j)^{-1} = \frac{\nu\widehat{\lambda}_j}{1 + \nu\widehat{\lambda}_j} \leq \min\{1, \nu\widehat{\lambda}_j\}, \quad \text{and}$$

$$1 - [R^\nu]_{jj} = \frac{\nu\widehat{\lambda}_j}{1 + \nu\widehat{\lambda}_j} \geq \frac{1}{2}\min\{1, \nu\widehat{\lambda}_j\}.$$

Appendix D. Properties of the Empirical Rademacher Complexity

In this section, we prove that the $\widehat{\varepsilon}_n$ lies in the interval $(0, \infty)$, and is unique. Recall that the stopping point $\widehat{T}$ is defined as $\widehat{\varepsilon}_n := \arg\min\left\{\epsilon > 0 \mid \widehat{\mathcal{R}}_K(\epsilon) > \epsilon^2/(2e\sigma)\right\}$. Re-arranging and substituting for $\widehat{\mathcal{R}}_K(\epsilon)$ yields the equivalent expression

$$\widehat{\varepsilon}_n := \arg\min\left\{\epsilon > 0 \mid \sum_{i=1}^n \min\left\{\epsilon^{-2}\widehat{\lambda}_i, 1\right\} > n\epsilon^2/(4e^2\sigma^2)\right\}.$$

Note that $\sum_{i=1}^n \min\left\{\epsilon^{-2}\widehat{\lambda}_i, 1\right\}$ is non-increasing in ϵ while $n\epsilon^2$ is increasing in ϵ. Furthermore when $\epsilon = 0$, $0 = n\epsilon^2 < \sum_{i=1}^n \min\left\{\epsilon^{-2}\widehat{\lambda}_i, 1\right\} > 0$ while for $\epsilon = \infty$, $\sum_{i=1}^n \min\left\{\eta_t\widehat{\lambda}_i, 1\right\} < n\epsilon^2$, recalling that $\eta_t = \sum_{\tau=0}^{t-1}\alpha_\tau$. Hence $\widehat{\varepsilon}_n$ exists. Further, $\widehat{\mathcal{R}}_K(\epsilon)$ is a continuous function of ϵ since it is the sum of n continuous functions, Therefore, the critical radius $\widehat{\varepsilon}_n$ exists, is unique and satisfies the fixed point equation

$$\widehat{\mathcal{R}}_K(\widehat{\varepsilon}_n) = \widehat{\varepsilon}_n^2/(2e\sigma).$$

Finally, we show that the integer $\widehat{T}$ belongs to the interval $[0, \infty)$ and is unique for any valid sequence of step-sizes. Be the definition of $\widehat{T}$ given by the stopping rule (6) and $\widehat{\varepsilon}_n$, we have $\frac{1}{\eta_{\widehat{T}+1}} \leq \widehat{\varepsilon}_n^2 \leq \frac{1}{\eta_{\widehat{T}}}$. Since $\eta_0 = 0$ and $\eta_t \to \infty$ as $t \to \infty$ and $\widehat{\varepsilon}_n \in (0, \infty)$, there exists a unique stopping point $\widehat{T}$ in the interval $[0, \infty)$.

Appendix E. Auxiliary Results for Theorem 2

This appendix is devoted to the proofs of auxiliary lemmas used in the proof for Theorem 2.

E.1 Proof of Lemma 9

Let us write $f_t = \sum_{k=0}^{\infty} \sqrt{\lambda_k} a_k \phi_k$, so that $\|f_t\|_{\mathcal{H}}^2 = \sum_{k=0}^{\infty} a_k^2$. Recall the linear operator $\Phi_X : \ell^2(\mathbb{N}) \to \mathbb{R}^n$ defined element-wise via $[\Phi_X]_{jk} = \phi_j(x_k)$ and the diagonal operator $D : \ell^2(\mathbb{N}) \to \ell^2(\mathbb{N})$ with entries $[D]_{jj} = \lambda_j$ and $[D]_{jk} = 0$ for $j \neq k$. By the definition of the gradient update (3), we have the relation $a = \frac{1}{n} D^{1/2} \Phi_X^T K^{-1} f_t(x_1^n)$. Since $\frac{1}{n} \Phi_X D \Phi_X^T = K$,

$$\|f_t\|_{\mathcal{H}}^2 = \|a\|_2^2 = \frac{1}{n} f_t(x_1^n)^T K^{-1} f_t(x_1^n). \tag{30}$$

Recall the eigendecomposition $K = U \Lambda U^T$ with $\Lambda = \operatorname{diag}(\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots \widehat{\lambda}_r)$, and the relation $U^T f^t(x_1^n) = (I - S^t) U^T y_1^n$. Substituting into Equation (30) yields

$$\begin{aligned}
\|f_t\|_{\mathcal{H}}^2 &= \frac{1}{n} (y_1^n)^T U (I - S^t)^2 \Lambda^{-1} U^T y_1^n \\
&\overset{(i)}{=} \frac{1}{n} (f^*(x_1^n) + w)^T U (I - S_t)^2 \Lambda^{-1} U^T (f^*(x_1^n) + w) \\
&= \underbrace{\frac{2}{n} w^T U (I - S_t)^2 \Lambda^{-1} U^T f^*(x_1^n)}_{A_t} + \underbrace{\frac{1}{n} w^T U (I - S_t)^2 \Lambda^{-1} U^T w}_{B_t} \\
&\quad + \underbrace{\frac{1}{n} f^*(x_1^n)^T U (I - S_t)^2 \Lambda^{-1} U^T f^*(x_1^n)}_{C_t}
\end{aligned}$$

where equality (i) follows from the observation equation $y_1^n = f^*(x_1^n) + w$. From Lemma 8, we have $1 - S_{jj}^t \leq 1$, and hence $C_t \leq \frac{1}{n} f^*(x_1^n)^T U \Lambda^{-1} U^T f^*(x_1^n) \overset{(i)}{\leq} 1$, where the last step follows from the analysis in Section 4.1.1.

It remains to derive upper bounds on the random variables A_t and B_t.

E.1.1 Bounding A_t

Since the elements of w are i.i.d, zero-mean and sub-Gaussian with parameter σ, we have $\mathbb{P}[|A_t| \geq 1] \leq 2 \exp(-\frac{n}{2\sigma^2 \nu^2})$, where $\nu^2 := \frac{4}{n} [f^*(x_1^n)]^T U (I - S_t)^4 \Lambda^{-2} U^T f^*(x_1^n)$. Since $(1 - (S_t)_{jj}) \leq 1$, we have

$$\begin{aligned}
\nu^2 \leq \frac{4}{n} f^*(x_1^n)^T U (I - S_t) \Lambda^{-2} U^T f^*(x_1^n) &\leq \frac{4}{n} \sum_{j=1}^{r} \frac{[U^T f^*(x_1^n)]_j^2}{\widehat{\lambda}_j^2} \min(1, \eta_t \widehat{\lambda}_j) \\
&\leq 4 \frac{\eta_t}{n} \sum_{j=1}^{r} \frac{[U^T f^*(x_1^n)]_j^2}{\widehat{\lambda}_j} \\
&\leq 4 \eta_t,
\end{aligned}$$

where the final inequality follows from the analysis in Section 4.1.1.

E.1.2 Bounding B_t

We begin by noting that

$$B_t = \frac{1}{n} \sum_{j=1}^{r} \frac{(1 - S_{jj}^t)^2}{\widehat{\lambda}_j} [U^T w]_j^2 = \frac{1}{n} \operatorname{trace}(U Q U^T, w w^T),$$

where $Q = \mathrm{diag}\{\frac{(1-S_{jj}^t)^2}{\widehat{\lambda}_j},\ j = 1, 2, \ldots r\}$. Consequently, B_t is a quadratic form in zero-mean sub-Gaussian variables, and using the tail bound (27), we have

$$\mathbb{P}\big[|B_t - \mathbb{E}[B_t]| \geq 1\big[\big] \leq \exp(-c\min\{n|||UQU^T|||_{\mathrm{op}}^{-1}, n^2|||UQU^T|||_{\mathrm{F}}^{-2}\})$$

for a universal constant c. It remains to bound $\mathbb{E}[B_t]$, $|||UQU^T|||_{\mathrm{op}}$ and $|||UQU^T|||_{\mathrm{F}}$.

We first bound the mean. Since $\mathbb{E}[ww^T] \preceq \sigma^2 I_{n\times n}$ by assumption, we have

$$\mathbb{E}[B_t] \ \leq \frac{\sigma^2}{n}\,\mathrm{trace}(Q)\frac{1}{n}\sum_{j=1}^r = \ (\frac{(1-S_{jj}^t)^2}{\widehat{\lambda}_j}) \ \leq \frac{\eta_t}{n}\sum_{j=1}^r \min((\eta_t\widehat{\lambda}_j)^{-1}, \eta_t\widehat{\lambda}_j)$$

But by the definition (6) of the stopping rule and the fact that $t \leq \widehat{T}$, we have

$$\frac{\eta_t}{n}\sum_{j=1}^r \min((\eta_t\widehat{\lambda}_j)^{-1}, \eta_t\widehat{\lambda}_j) \leq \eta_t^2\mathcal{R}_K^2(1/\sqrt{\eta_t}) \ \leq \ \frac{1}{\sigma^2},$$

showing that $\mathbb{E}[B_t] \leq 1$.

Turning to the operator norm, we have

$$|||UQU^T|||_{\mathrm{op}} = \max_{j=1,\ldots,r}(\frac{(1-S_{jj}^t)^2}{\widehat{\lambda}_j}) \leq \max_{j=1,\ldots,r}\min(\widehat{\lambda}_j^{-1}, \eta_t^2\widehat{\lambda}_j) \ \leq \eta_t.$$

As for the Frobenius norm, we have

$$\frac{1}{n}|||UQU^T|||_{\mathrm{F}}^2 = \sum_{j=1}^r(\frac{(1-S_{jj}^t)^4}{\widehat{\lambda}_j^2}) \leq \frac{1}{n}\sum_{j=1}^r\min(\widehat{\lambda}_j^{-2}, \eta_t^4\widehat{\lambda}_j^2) \ \leq \frac{\eta_t^3}{n}\sum_{j=1}^r\min(\eta_t^{-3}\widehat{\lambda}_j^{-2}, \eta_t\widehat{\lambda}_j^2)$$

Using the definition of the empirical kernel complexity, we have

$$\frac{1}{n}|||UQU^T|||_{\mathrm{F}}^2 \leq \eta_t^3\mathcal{R}_K^2(1/\sqrt{\eta_t}) \ \leq \frac{\eta_t}{\sigma^2},$$

where the final inequality holds for $t \leq \widehat{T}$, using the definition of the stopping rule.

Putting together the pieces, we have shown that

$$\mathbb{P}[|B_t| \geq 2 \quad \text{or} \quad |A_t| \geq 1] \leq \exp(-cn/\eta_t)$$

for all $t \leq \widehat{T}$. Since $\frac{1}{\eta_t} \geq \widehat{\varepsilon}_n^2$ for any $t \leq \widehat{T}$, the claim follows.

E.2 Proof of Lemma 11

In this section, we need to show that $\widehat{\varepsilon}_n \leq \varepsilon_n$. Recall that $\widehat{\varepsilon}_n$ and ε_n satisfy

$$\widehat{\mathcal{R}}_K(\widehat{\varepsilon}_n) = \frac{\widehat{\varepsilon}_n^2}{2e\sigma} \ \text{ and } \ \mathcal{R}_{\mathbb{K}}(\varepsilon_n) = \frac{\varepsilon_n^2}{40\sigma}.$$

It suffices to prove that $\widehat{\mathcal{R}}_K(\varepsilon_n) \leq \frac{\varepsilon_n^2}{2e\sigma}$ using the definition of $\widehat{\varepsilon}_n$.

In order to prove the claim, we define the random variables

$$\widehat{Z}_n(w,t) := \sup_{\substack{\|g\|_{\mathcal{H}} \leq 1 \\ \|g\|_n \leq t}} \Big|\frac{1}{n}\sum_{i=1}^n w_i g(x_i)\Big|, \quad \text{and} \quad Z_n(w,t) := \mathbb{E}_x\Big[\sup_{\substack{\|g\|_{\mathcal{H}} \leq 1 \\ \|g\|_2 \leq t}} \Big|\frac{1}{n}\sum_{i=1}^n w_i g(x_i)\Big|\Big],$$

where $w_i \sim N(0,1)$ are i.i.d. standard normal, as well as the associated (deterministic) functions

$$\widehat{\mathcal{Q}}_n(t) \ := \mathbb{E}_w\big[\widehat{Z}_n(w;t)\big] \quad \text{and} \quad \mathcal{Q}_n(t) \ := \mathbb{E}_w\big[Z_n(w;t)\big].$$

By results of Mendelson (2002), there are universal constants $0 < c_\ell \leq c_u$ such that for all $t^2 \geq 1/n$, we have

$$c_\ell \mathcal{R}_{\mathbb{K}}(t) \leq \mathcal{Q}_n(t) \leq c_u \mathcal{R}_{\mathbb{K}}(t), \quad \text{and} \quad c_\ell \widehat{\mathcal{R}}_K(t) \leq \widehat{\mathcal{Q}}_n(t) \leq c_u \widehat{\mathcal{R}}_K(t).$$

We first appeal to the concentration of Lipschitz functions for Gaussian random variables to show that $\widehat{Z}_n(w,t)$ and $Z_n(w,t)$ are concentrated around their respective means. For any $t > 0$ and vectors $w, w' \in \mathbb{R}^n$, we have

$$|\widehat{Z}_n(w,t) - \widehat{Z}_n(w',t)| \leq \sup_{\substack{\|g\|_n \leq t \\ \|g\|_{\mathcal{H}} \leq 1}} \frac{1}{n}|\sum_{i=1}^n (w_i - w_i')g(x_i)| \leq \frac{t}{\sqrt{n}}\|w - w'\|_2,$$

showing that $w \mapsto \widehat{Z}_n(w,t)$ is $\frac{t}{\sqrt{n}}$-Lipschitz with respect to the ℓ_2 norm. A similar calculation for $w \mapsto Z_n(w,t)$ shows that

$$|\mathbb{E}_x[\widehat{Z}_n(w,t)] - \mathbb{E}_x[\widehat{Z}_n(w',t)]| \leq \mathbb{E}_x[\sup_{\substack{\|g\|_2 \leq t \\ \|g\|_{\mathcal{H}} \leq 1}} \frac{1}{n}|\sum_{i=1}^n (w_i - w_i')g(x_i)|] \leq \frac{t}{\sqrt{n}}\|w - w'\|_2,$$

so that it is also Lipschitz $\frac{t}{\sqrt{n}}$. Consequently, standard concentration results (Ledoux, 2001) imply that

$$\begin{aligned} \mathbb{P}\big[|\widehat{Z}_n(w,t) - \widehat{\mathcal{Q}}_n(t)| \geq t_0\big] &\leq 2\exp\Big(-\frac{nt_0^2}{2t^2}\Big), \quad \text{and} \\ \mathbb{P}\big[|Z_n(w,t) - \mathcal{Q}_n(t)| \geq t_0\big] &\leq 2\exp\Big(-\frac{nt_0^2}{2t^2}\Big). \end{aligned} \tag{31}$$

Now let us condition on the two events

$$\mathcal{A}(t,t_0) := \{|\widehat{Z}_n(w,t) - \widehat{\mathcal{Q}}_n(t)| \leq t_0\}, \quad \text{and} \quad \mathcal{A}'(t,t_0) := \{|Z_n(w,t) - \mathcal{Q}_n(t)| \leq t_0\}.$$

We then have

$$\widehat{\mathcal{R}}_K(\varepsilon_n) \overset{(a)}{\leq} \widehat{Z}_n(w,\varepsilon_n) + \frac{\varepsilon_n^2}{4e\sigma} \overset{(b)}{\leq} Z_n(w,2\varepsilon_n) + \frac{\varepsilon_n^2}{4e\sigma} \overset{(c)}{\leq} 2\mathcal{R}_{\mathbb{K}}(\varepsilon_n) + \frac{3\varepsilon_n^2}{8e\sigma} \overset{(d)}{\leq} \frac{\varepsilon_n^2}{2e\sigma},$$

where inequality (a) follows the first bound in Equation (31) with $t_0 = \frac{\varepsilon_n^2}{4e\sigma}$ and $t = \varepsilon_n^2$, inequality (b) follows from Lemma 10 with $t = \varepsilon_n$, inequality (c) follows from the second bound (31) with $t_0 = \frac{\varepsilon_n^2}{8e\sigma}$ and $t = \varepsilon_n^2$, and inequality (d) follows from the definition of ε_n. Since the events $\mathcal{A}(t,t_0)$ and $\mathcal{A}'(t,t_0)$ hold with the stated probability, the claim follows.

References

R. S. Anderssen and P. M. Prenter. A formal comparison of methods proposed for the numerical solution of first kind integral equations. *Jour. Australian Math. Soc. (Ser. B)*, 22:488–500, 1981.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

A. R. Barron, A. Cohen, W. Dahmen, and R. A. DeVore. Approximation and learning by greedy algorithms. *Annals of Statistics*, 36(1):64–94, 2008.

P. Bartlett and M. Traskin. Adaboost is consistent. *Journal of Machine Learning Research*, 8:2347–2368, 2007.

P. Bartlett, O. Bousquet, and S. Mendelson. Local Rademacher complexities. *Annals of Statistics*, 33(4):1497–1537, 2005.

F. Bauer, S. Pereverzev, and L. Rosasco. On regularization algorithms in learning theory. *J. Complexity*, 23:52–72, 2007.

M. S. Birman and M. Z. Solomjak. Piecewise-polynomial approximations of functions of the classes W_p^α. *Math. USSR-Sbornik*, 2(3):295–317, 1967.

G. Blanchard and M. Kramer. Optimal learning rates for kernel conjugate gradient regression. In *Proceedings of the NIPS Conference*, 2010.

P. Buhlmann and B. Yu. Boosting with L^2 loss: Regression and classification. *Journal of American Statistical Association*, 98:324–340, 2003.

A. Caponetto and Y. Yao. Adaptation for regularization operators in learning theory. Technical Report CBCL Paper #265/AI Technical Report #063, Massachusetts Institute of Technology, September 2006.

A. Caponetto and Y. Yao. Cross-validation based adaptation for regularization operators in learning theory. *Analysis and Applications*, 8(2):161–183, 2010.

A. Caponneto. Optimal rates for regularization operators in learning theory. Technical Report CBCL Paper #264/AI Technical Report #062, Massachusetts Institute of Technology, September 2006.

P. Drineas and M. W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

Y. Freund and R. 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.

J.H. Friedman and B. Popescu. Gradient directed regularization. Technical report, Stanford University, 2004.

C. Gu. *Smoothing Spline ANOVA Models*. Springer Series in Statistics. Springer, New York, NY, 2002.

M. G. Gu and H. T. Zhu. Maximum likelihood estimation by markov chain monte carlo approximation. *J. R. Statist. Soc. B*, 63:339–355, 2001.

P. Hall and J.S. Marron. On variance estimation in nonparametric regression. *Biometrika*, 77:415–419, 1990.

W. Jiang. Process consistency for adaboost. *Annals of Statistics*, 32:13–29, 2004.

G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Jour. Math. Anal. Appl.*, 33:82–95, 1971.

V. Koltchinskii. Local Rademacher complexities and oracle inequalities in risk minimization. *Annals of Statistics*, 34(6):2593–2656, 2006.

M. Ledoux. *The Concentration of Measure Phenomenon*. Mathematical Surveys and Monographs. American Mathematical Society, Providence, RI, 2001.

L. Mason, J. Baxter, P., and M. Frean. Boosting algorithms as gradient descent. In *Neural Information Processing Systems (NIPS)*, December 1999.

S. Mendelson. Geometric parameters of kernel machines. In *Proceedings of COLT*, pages 29–43, 2002.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society A*, 209:415–446, 1909.

N. Morgan and H. Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. In *Proceedings of Neural Information Processing Systems*, 1990.

L. Orecchia and M. W. Mahoney. Implementing regularization implicitly via approximate eigenvector computation. In *ICML '11*, 2011.

G. Raskutti, M. J. Wainwright, and B. Yu. Minimax-optimal rates for sparse additive models over kernel classes via convex programming. *Journal of Machine Learning Research*, 12: 389–427, March 2012.

S. Saitoh. *Theory of Reproducing Kernels and its Applications*. Longman Scientific & Technical, Harlow, UK, 1988.

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

C. M. Stein. Estimation of the mean of a multivariate normal distribution. *Annals of Statistics*, 9(6):1135–1151, 1981.

C. J. Stone. Additive regression and other nonparametric models. *Annals of Statistics*, 13 (2):689–705, 1985.

O. N. Strand. Theory and methods related to the singular value expansion and Landweber's iteration for integral equations of the first kind. *SIAM J. Numer. Anal.*, 11:798–825, 1974.

S. van de Geer. *Empirical Processes in M-Estimation*. Cambridge University Press, 2000.

E. De Vito, S. Pereverzyev, and L. Rosasco. Adaptive kernel methods using the balancing principle. *Foundations of Computational Mathematics*, 10(4):455–479, 2010.

G. Wahba. Three topics in ill-posed problems. In M. Engl and G. Groetsch, editors, *Inverse and ill-posed problems*, pages 37–50. Academic Press, 1987.

G. Wahba. *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia, PN, 1990.

H. L. Weinert, editor. *Reproducing Kernel Hilbert Spaces : Applications in Statistical Signal Processing*. Hutchinson Ross Publishing Co., Stroudsburg, PA, 1982.

F. T. Wright. A bound on tail probabilities for quadratic forms in independent random variables whose distributions are not necessarily symmetric. *Annals of Probability*, 1(6): 1068–1070, 1973.

Y. Yang and A. Barron. Information-theoretic determination of minimax rates of convergence. *Annals of Statistics*, 27(5):1564–1599, 1999.

Y. Yao, L. Rosasco, and A. Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26:289–315, 2007.

T. Zhang. Learning bounds for kernel regression using effective data dimensionality. *Neural Computation*, 17(9):2077–2098, 2005.

T. Zhang and B. Yu. Boosting with early stopping: Convergence and consistency. *Annal of Statistics*, 33:1538–1579, 2005.

Journal of Machine Learning Research 15 (2014) 367-443 Submitted 1/13; Revised 9/13; Published 2/14

Unbiased Generative Semi-Supervised Learning

Patrick Fox-Roberts* PFOXROBERTS@GMAIL.COM
Cambridge University Engineering Department
Trumpington Street
Cambridge, CB2 1PZ, UK

Edward Rosten ED@COMPUTERVISIONCONSULTING.COM
Computer Vision Consulting
7th floor
14 Bonhill Street
London, EC2A 4BX, UK

Editor: William Cohen

Abstract

Reliable semi-supervised learning, where a small amount of labelled data is complemented by a large body of unlabelled data, has been a long-standing goal of the machine learning community. However, while it seems intuitively obvious that unlabelled data can aid the learning process, in practise its performance has often been disappointing. We investigate this by examining generative maximum likelihood semi-supervised learning and derive novel upper and lower bounds on the degree of bias introduced by the unlabelled data. These bounds improve upon those provided in previous work, and are specifically applicable to the challenging case where the model is unable to exactly fit to the underlying distribution a situation which is common in practise, but for which fewer guarantees of semi-supervised performance have been found. Inspired by this new framework for analysing bounds, we propose a new, simple reweighing scheme which provides a provably unbiased estimator for arbitrary model/distribution pairs—an unusual property for a semi-supervised algorithm. This reweighing introduces no additional computational complexity and can be applied to very many models. Additionally, we provide specific conditions demonstrating the circumstance under which the unlabelled data will lower the estimator variance, thereby improving convergence.

Keywords: Kullback-Leibler, semi-supervised, asymptotic bounds, bias, generative model

1. Introduction

Reliable semi-supervised learning has been a long standing goal of the machine learning community. Its desirability is motivated by the observation that when collecting data sets, often each sample has two distinct parts: some feature X, collected from some real world population, often consisting of one or more basic measurements; and some label Y, assigned by the experimenter, representing a higher level concept. Furthermore the act of assigning this higher level label very often constitutes a major bottleneck in the data set creation process. It is perhaps expensive (requiring an expert's opinion), slow (requiring

*. PFR is currently affiliated with The Randal Division, Guy's Campus, King's College London, SE1 1UL

an investment of time or staff), or in some way destructive (requiring a component to be tested to destruction, or the death of a patient).

If we wish to fit a model parametrised by some set of parameters θ to this distribution, we will need some data set D_L consisting of N_L labelled samples, $D_L = (x_i, y_i)_{i=1,\dots,N_L}$, to train our model with; for example, if we are training using maximum likelihood, we must find the parameters which maximise $P(D_L|\theta)$, which for iid data is equivalent to finding

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{N_L} \log\left(P(x_i, y_i|\theta)\right). \tag{1}$$

In order to get a solution which generalises well to unseen data N_L may have to be quite large, especially if the model is rich or the feature space $\mathcal{X}$ high dimensional.

A far preferable situation would be to be able to utilise a smaller labelled data set D_L, augmented with an additional data set D_U consisting of N_U unlabelled samples, $D_U = (x_i)_{i=N_L+1,\dots,N_L+N_U}$, which consist only of their observed feature rather than a feature - label pair. In essence the unlabelled data is used to 'bootstrap' the labelled. Unlabelled samples tell us the shape of our distribution in the feature space, while labelled samples give us the classification information.

At first glance, utilising unlabelled data to aid in fitting the parameters of some model appears trivial. Inspired by the likelihood principal (Jaynes, 2003), it is tempting to simply augment the likelihood function of the parameters, $P(D_L|\theta)$, with the additional unlabelled data, $P(D_L, D_U|\theta)$, and proceed with training exactly as before, that is, find the parameters

$$\theta_S^\star = \arg\max_\theta \sum_{i=1}^{N_L} \log\left(P(x_i, y_i|\theta)\right) + \sum_{i=N_L}^{N_L+N_U} \log\left(P(x_i|\theta)\right). \tag{2}$$

In practice however this has proven to give mixed results, sometime improving model fitting, other times worsening it. This unpredictable of performance has formed a very major barrier to more widespread adoption of semi-supervised techniques. Many alternative algorithms have been developed to counter this. However, there still exists a need to better understand and quantify why more standard methods fail.

This paper examines the effect of including unlabelled data in a training set when performing maximum likelihood fitting of generative models. In particular, it is well known (see, for example, Bishop, 2006) that maximising the parameter likelihood for labelled data approximately minimises the Kullback Leibler divergence between the parametric distribution $P(X, Y|\theta)$ and the underlying distribution the data is sampled from, $P(X, Y)$. We show that maximising the likelihood of a data set containing unlabelled samples minimises a different divergence. We then show that the possible error between this and the correct divergence may grow rapidly with the proportion of unlabelled data, and will do so monotonically.

Out of necessity, the analysis presented shall only concern itself with generative models. This follows in the footsteps of numerous other pieces of work which have shed light on the generative semi-supervised learning problem, for example Castelli and Cover (1995), Castelli and Cover (1996), Dillon et al. (2010), Cozman et al. (2003) and Yang and Priebe (2011). Moreover, generative models are of interest in and of themselves. For example, they

are used in the fields of computer vision and text analysis, both of which could potentially benefit from better semi-supervised algorithms; recent examples of such work include that of Rauschert and Collins (2012), Beecks et al. (2011), Lücke and Eggert (2010), Kang et al. (2012) and Zhuang et al. (2012). In the general case there is also evidence that generative models can converge faster than discriminative, as shown by Ng and Jordan (2002), and so are valuable when dealing with small data sets.

2. Previous Work

A great deal of work has been done proposing algorithms designed to take advantage of semi-supervised data. Here we shall concern ourselves instead with examining the work done on finding general bounds on performance.

We begin by considering the highly influential work by Castelli and Cover (1995, 1996). This looks not at a particular semi-supervised algorithm, but rather at a slightly more general question of when unlabelled samples can be of value. They conclude that for an identifiable (as defined in the paper) binary decision problem, using a generative model, the misclassification risk decreases exponentially fast towards the Bayes error as the number of labelled samples increases. This result is encouraging. However, the requirement of identifiability is a strict one. In practise it cannot often be guaranteed, and may even be flatly contradicted.

The work of Dillon et al. (2010) builds upon this. Amongst other things they confirm that provided a data set is generated from $P(X, Y|\theta_0)$ where $\theta_0 \in \Omega$, the estimator

$$\hat{\theta}_N = \arg\max_{\theta\in\Omega} \sum_{i=1}^{N_L} \log(P(x_i, y_i|\theta)) + \sum_{i=N_L+1}^{N} \log(P(x_i|\theta))$$

is consistent. As such, in cases where there is good reason to believe the true distribution is drawn from the same family as our parametric model, we can expect consistent convergence. They also provide one of few examinations of the associated variance of an estimator, though again under the assumption of an identifiable model.

In a similar vein Zhang (2000) examines the fisher information matrix when learning parameters for semi-supervised learning, and conclude that even when their true distribution can not be expressed by the model parameters being fitted, unlabelled samples always aid in learning in that they reduce the variance of the estimator. From this work we can conclude that adding unlabelled samples is not preventing consistent convergence. As such, if performance is observed to often worsen instead of improve as the number of unlabelled samples increases, the fault must lie elsewhere.

The asymptotic behaviours of semi-supervised learning where the model is mis-specified has been further studied by Cozman et al. (2003); Cozman and Cohen (2006, 2002), where no assumptions are made about the parametric model being close to the underlying distribution. In particular, they show that the limiting value of the optimum parameters $\theta^\star$ when performing ML semi-supervised learning in such a scenario is

$$\arg\max_{\theta} \left((1-\lambda)E_{P(X,Y)}\left(\log P(x, y|\theta)\right) + \lambda E_{P(x)}\left(\log(P(x|\theta))\right)\right)$$

where λ is the probability of a sample being unlabelled. If λ varies (say by adding unlabelled samples) then this will likely change the optimal parameters $\theta^\star$, and so the associated error

rate. In the limit, as $\lambda \to 1$, we will tend towards the solution found training entirely on unlabelled data. They argue that with a few assumptions on the modelling densities, $\theta^\star$ is a continuous function of λ. They also show that an instance where the asymptotically optimal parameters are not changed by λ comes, as might be expected, when the model is "correct" and can be fitted exactly to the underlying distribution (i.e., the true distribution $P(x, y)$ is a member of the family of distributions that can be modelled by $P(x, y|\theta)$).

The relative value of labelled/unlabelled samples was also investigated in Ratsaby and Venkatesh (1995) for the case of classifying between two multivariate gaussian distributions of unknown class prior and position parameters. As in the work by Castelli and Cover, an exponential decrease in error rate with the number of labelled samples is shown, and an only polynomial decrease in the same with the number of unlabelled samples. However they also demonstrate a deleterious effect in the *dimensionality* of the space, indicating unlabelled samples are likely to be less useful in high dimensional spaces. Separately, the work of Shahshahani and Landgrebe (1994) examines learning the parameters of both a single gaussian and a GMM when labels are missing. They too note an interesting effect of dimensionality on semi-supervised learning, in particular from the point of view of the Hughes phenomenon (Hughes, 1968). This is the observation that, in theory, increasing the dimensionality of a classification problem by taking new measurements should never increase the Bayes error; yet in practise, if we are learning from sampled data we find performance will after a while degrade due to the larger number of parameters that must be estimated (this is very closely linked to the perhaps more familiar *Curse of Dimensionality*, see Bishop, 2006). They propose that semi-supervised learning can help mitigate this, but only if the rate of introduction of bias due to the unlabelled samples is lower than the decrease in variance of the estimator.

Recently, Yang and Priebe (2011) has provided an investigation of semi-supervised generative learning that builds upon these conclusions. The key parameters they identify are the asymptotic optima achieved when performing fully supervised learning, θ^*_{sup}, and those achieved from entirely un-supervised learning, θ^*_{unsup}. Provided that the ratio of N_L to N tends towards 0 as N tends towards infinity (where $N = N_L + N_U$) we have the scenario where we are moving from a high-variance, unbiased estimate, towards a low variance, biased estimate. Interestingly, the KL divergences between the distributions defined by θ^*_{sup} and θ^*_{unsup}, and between these distributions and a given estimate based on a data set (either fully labelled or a mixture of labelled and unlabelled), are identified as providing bounds on the probability that classification performance will improve/worsen as unlabelled data is added. Intuitively, if the divergence between the models specified by θ^*_{sup} and θ^*_{unsup} is small, then adding unlabelled data is less likely to significantly worsen results. They also show for a particular model that the point at which this occurs can be quite sharp. However, as it is likely to be different for different models and distributions, it still remains an open question how it can be best estimated.

Additionally, theoretic examinations of expected performance for other semi supervised learning situations, such as transductive learning (for example, Wang et al., 2007; Vapnik, 1998), PAC learning (Balcan and Blum, 2005; Blum and Balcan, 2010), and for generic loss functions (Syed and Taskar, 2010), have also been carried out. However, as our purpose here is to examine what can be said about generative semi-supervised learning we shall not discuss these further.

2.1 Non-ML Algorithms

Given the problems associated with standard ML semi-supervised learning, as well as the desire to utilise unlabelled samples in non-generative models, a large number of alternative objective functions have been proposed to take advantage of unlabelled data. Notable examples include Multi Conditional Learning (introduced by McCallum et al. 2006 and applied to semi-supervised learning by Druck et al. 2007) and the hybrid Bayesian approach of Lasserre et al. (2006), both of which utilise mixtures of generative and discriminative models; information theory based approaches, which consider the similarity of class predictions across the kNN graph such as Subramanya and Bilmes (2009, 2008), the mutual information of samples within local clusters (Szummer and Jaakkola, 2002), or the conditional entropy of class predictions across the unlabelled samples (Grandvalet and Bengio, 2006); Expectation Regularisation (Mann and McCallum, 2007), which seeks to enforce class proportion constraints; Co-training, (Blum and Mitchell, 1998), which makes use of situations where data is known to be separable in two different 'views'; transduction, (Vapnik, 1998), and the transductive support vector machine; kernel methods, such as those investigated by Krishnapuram et al. (2005) and Jaakkola and Haussler (1999), which seek to use unlabelled samples to build better kernel functions; and many others. A thorough literature review was carried out by Zhu (2005).

3. Local And Global Bounds On Semi-Supervised Divergences

We now present a number of theorems, showing the asymptotic limits of the performance of models trained on semi-supervised data using the standard technique Equation (2). While it has been previously noted in the literature that ML semi-supervised learning introduces bias when the model and underlying distributions do not match, we provide new bounds on the degree of this bias as a function of the proportion of unlabelled data, and the best case performance of our model if it were to be trained on a large labelled data set, giving new insight into the reason behind these bounds.

3.1 Notation And Conditions

A semi-supervised data set consists of two types of data - labelled samples drawn from $P(X, Y)$, and unlabelled drawn from $P(X)$. To allow us to deal with both of these within a single framework we shall introduce a new variable Z, and consider our *entire* data set to be drawn from $P(X, Z)$, $\{x_i, z_i\}_{i=1,\ldots N}$ in the space $\mathcal{X} \times \mathcal{Z}$. We shall allow the 'labelling' Z to take on the same set of values as Y, plus one extra, U, and therefore $\mathcal{Z} = \mathcal{Y} \cup U$. For every "labelled" sample, $z_i = y_i$, and for every "unlabelled" sample $z_i = U$. As such we now have the data set $\{x_i, z_i\}_{i=1,\ldots N}$. Similarly, we shall consider our parameters θ to specify a distribution $P(X, Z|\theta)$ rather than $P(X, Y|\theta)$, in a manner which will become clear as we proceed.

We shall now apply several conditions to $P(X, Z|\theta)$ and $P(X, Z)$ to allow them to reflect what we consider the typical maximum likelihood generative semi-supervised learning problem.

Condition 1 *X is conditionally independent of Z given Y — if we know the class y of sample x, z gives us no more information, that is,*

$$P(x|y,z,\theta) = P(x|y,\theta), \quad P(x|y,z) = P(x|y)$$

This first condition represents the fact that z_i can be considered a noisy estimate of y_i - in as much as it will either be equal to y_i, or it will take on the value U to indicate y_i is unknown. In either case, if we had access to the true value of y_i, then z_i would be irrelevant as it can give us no useful information. This condition is similar to the "missing at random" assumption discussed by Grandvalet and Bengio (2006).

Condition 2 *The labelled samples have been drawn randomly and labelled correctly. The unlabelled samples are similarly drawn randomly, with no class bias. As such,*

$$P(z|y,\theta) = \begin{cases} P(U|\theta), & z = U \\ \delta_k(z,y)P(\bar{U}|\theta), & z \neq U \end{cases} \qquad P(z|y) = \begin{cases} P(U), & z = U \\ \delta_k(z,y)P(\bar{U}), & z \neq U \end{cases}$$

where δ_k indicates the Kronecker delta function, and we have denoted $1 - P(U|\theta)$ as $P(\bar{U}|\theta)$ and $1 - P(U)$ as $P(\bar{U})$

This second condition specifies our labelling process. It is imagined that a 'bag' full of unlabelled samples initially exists, and individual ones are then drawn from it and the correct label associated with them by some expensive labelling process[1] to form the labelled set.

In practise truly drawing samples completely at random runs with risk of certain classes having zero labelled samples, which is likely to cause highly undesirable behaviour of the algorithm. We do not foresee this as a problem for two reasons. Firstly, in the asymptotic limit (which is what most of our work will be concerned with in this section) we will almost surely achieve labelled samples being drawn from all classes. Secondly, in practise, the individuals running the experiment are likely to ensure that all classes have some representative samples. This breaks the assumption of iid data; however, provided the class priors are respected when choosing how many samples to label from each class (or suitable weighting applied) we can still attain an asymptotically unbiased estimate of the expectation term in the divergence.

Condition 3 *The proportion of labelled data is known, letting us set $P(\bar{U}|\theta) = P(\bar{U}) = 1 - P(U)$*

We assume this as matching labels to samples is a process controlled entirely by the user, and that they use this knowledge to set $P(U|\theta)$ rather than having to infer it from the data.

1. This can be considered a somewhat simpler model to that proposed by Rosset et al. (2005), where the labelling also depended on the feature vector x. This work however is interested in the case of biased semi-supervised learning, which we assume here to not be the case.

3.1.1 Divergences

The KL divergence is a widely used method of measuring the similarity between two distributions, and one which shall be made extensive use of in this article. For distributions $P(X,Y)$ and $P(X,Y|\theta)$ where X is a continuous random variable and Y is discrete, it is defined as

$$KL(P(X,Y)||P(X,Y|\theta)) = \int_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} P(x,y) \log\left(\frac{P(x,y)}{P(x,y|\theta)}\right)$$

It is perhaps most widely used as a justification of maximum likelihood methods, as it is a standard proof that the parameters

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{N_L} \log\left(p\left(x_i, y_i|\theta\right)\right)$$

are an asymptotically unbiased minimiser of $KL(P(X,Y)||P(X,Y|\theta))$, for example, see Bishop (2006).

For brevity and to make subsequent equations more readable we shall introduce a more concise notation to refer to the KL divergence. For random variables A and B and parameters θ, the full divergence shall be denoted as

$$D(P(A,B),\theta) \equiv KL(P(A,B)||P(A,B|\theta))$$

and the conditional divergence as

$$D(P(A|B),\theta) \equiv KL(P(A|B)||P(A|B,\theta)).$$

3.2 Standard ML Semi-Supervised Learning Expressed As A Divergence

Our first step is to demonstrate that when a proportion of a data set whose likelihood we are maximising is lacking labels, in the asymptotic limit we will minimise a different divergence to that we might wish - specifically, we minimise $D(P(X,Z)|\theta)$ rather than $D(P(X,Y)|\theta)$.

Theorem 4 *Subject to the conditions in 3.1, maximising*

$$\prod_{i=1}^{N_L} P(x_i, y_i|\theta) \prod_{i=n_L+1}^{N_L+N_U} P(x_i|\theta) \tag{3}$$

w.r.t. θ minimises an asymptotically unbiased estimate of a term directly proportional to $D(P(X,Z),\theta)$, not $D(P(X,Y),\theta)$

Proof Given a set of N samples $\{x_i, z_i\}_{i=1,\ldots N}$ drawn from $P(X,Z)$, we can approximate the expectation term in $D(P(X,Z)|\theta)$ with an arithmetic mean over our samples (for example, see MacKay, 2003). Ignoring terms which are not a function of θ, and taking the antilog, we attain

$$\arg\min_\theta D(P(X,Z),\theta) \approx \arg\max_\theta \prod_{i=1}^{N} P(x_i, z_i|\theta). \tag{4}$$

Examining the above, and making use of Condition 1 to simplify $P(x_i|z_i, y, \theta)$ into $P(x_i|y, \theta)$, we can rewrite the likelihood contribution of a single sample i, $P(x_i, z_i|\theta)$ as follows

$$\begin{aligned} P(x_i, z_i|\theta) &= \sum_{y\in\mathcal{Y}} P(x_i, z_i, y|\theta) \\ &= \sum_{y\in\mathcal{Y}} P(x_i|z_i, y, \theta)P(z_i|y, \theta)P(y|\theta) \\ &= \sum_{y\in\mathcal{Y}} P(x_i|y, \theta)P(z_i|y, \theta)P(y|\theta) \\ &= \sum_{y\in\mathcal{Y}} P(x_i, y|\theta)P(z_i|y, \theta). \end{aligned}$$

This can be simplified further using Conditions 2 and 3, depending on the value of z_i. First consider the case where $z_i = U$

$$P(x_i, z_i|\theta)|_{z_i=U} = \sum_{y\in\mathcal{Y}} P(x_i, y|\theta)P(U|\theta) = P(x_i|\theta)P(U). \tag{5}$$

Thus, a sample whose labelling z_i indicates it is unlabelled contributes a quantity proportional to $P(x_i|\theta)$ to our likelihood expression. Now consider a single labelled example (i.e., where $z_i \neq U$),

$$\begin{aligned} P(x_i, z_i|\theta)|_{z_i\neq U} &= \sum_{y\in\mathcal{Y}} P(x_i, y|\theta)\delta_k(y, z_i)P(\bar{U}|\theta) \\ &= P(x_i, y|\theta)|_{y=z_i}P(\bar{U}) \\ &= P(x_i, y_i|\theta)P(\bar{U}) \end{aligned} \tag{6}$$

where we have made a slight change of notation in the last term to represent that if $z_i \neq U$, then y_i is known. This contributes a term proportional to $P(x_i, y_i|\theta)$ to the likelihood. If we substitute these results back into Equation (4), our final likelihood expression is

$$\arg\max_\theta \prod_{i,z_i\neq U} P(\bar{U})P(x_i, y_i|\theta) \prod_{j,z_j=U} P(U)P(x_j|\theta)$$

which is equivalent to maximising Equation (3). ■

The form of our final likelihood function is the same as that found by Cozman and Cohen (2006). The difference is in the interpretation of this. While Cozman and Cohen (2006) considers it simply as a biased approximate minimiser of $D(P(X, Y), \theta)$, we consider it an *unbiased* minimiser of a *new divergence* $D(P(X, Z), \theta)$. The utility of this is that we can investigate the 'bias' introduced by considering the relationship between this semi-supervised divergence and the original fully supervised one, and the properties of KL divergences.

3.3 Bounding $D\big(P(X,Y),\theta\big)$ With $D\big(P(X,Z),\theta\big)$

Maximising the likelihood of a partially labelled data set corresponds to approximately minimising $D\left(P(X,Z),\theta\right)$. We now examine how $D\left(P(X,Z),\theta\right)$ is related to $D\left(P(X,Y),\theta\right)$, and show that a set of upper and lower bounds can be formed using it.

Theorem 5 *Subject to the conditions in 3.1, for a given set of parameters θ, $D\big(P(X,Z),\theta\big)$ defines an upper and lower set of bounds on $D\big(P(X,Y),\theta\big)$ as follows:*

$$D\big(P(X,Z),\theta\big) \leq D\big(P(X,Y),\theta\big) \leq \frac{D\big(P(X,Z),\theta\big)}{P(\bar{U})}. \tag{7}$$

Remark 6 *These bounds imply that, for a given $D\big(P(X,Z),\theta\big)$ that we are optimising, the divergence of interest $D\big(P(X,Y),\theta\big)$ could vary by up to a factor $P(\bar{U})^{-1}$. In situations where $P(\bar{U})^{-1}$ is large, this uncertainty may become the dominant factor in determining the quality of our result.*

Proof Consider the KL divergence $D(P(X,Z),\theta)$. We shall take the summation over Z and split out the term $z=U$, noting that $\mathcal{Z}-U=\mathcal{Y}$

$$\begin{aligned} D(P(X,Z),\theta) = &\int_{x\in\mathcal{X}} \sum_{z\in\mathcal{Y}} P(x,z)\log\Big(\frac{P(x,z)}{P(x,z|\theta)}\Big)dx \\ &+\int_{x\in\mathcal{X}} P(x,z)|_{z=U}\log\Big(\frac{P(x,z)|_{z=U}}{P(x,z|\theta)|_{z=U}}\Big)dx. \end{aligned}$$

Using Equation (5) and Equation (6), and their corresponding counterparts when not conditioned on θ, we can simplify the terms within our logarithms

$$\frac{P(x,z)|_{z=U}}{P(x,z|\theta)|_{z=U}} = \frac{P(x)P(U)}{P(x|\theta)P(U)} = \frac{P(x)}{P(x|\theta)},$$

$$\frac{P(x,z)|_{z\neq U}}{P(x,z|\theta)|_{z\neq U}} = \frac{P(x,y)|_{y=z}}{P(x,y|\theta)|_{y=z}}.$$

Using these identities, and Equation (5) and Equation (6) this allows us to rewrite the divergence $D(P(X,Z),\theta)$

$$\begin{aligned} D(P(X,Z),\theta) = P(\bar{U})\int\limits_{x\in\mathcal{X}} \sum_{z\in\mathcal{Y}} P(x,y)|_{y=z}\log\Big(\frac{P(x,y)|_{y=z}}{P(x,y|\theta)|_{y=z}}\Big)dx \\ +P(U)\int\limits_{x\in\mathcal{X}} P(x)\log\Big(\frac{P(x)}{P(x|\theta)}\Big)dx. \end{aligned}$$

which is exactly equivalent to

$$D(P(X,Z),\theta) = P(\bar{U})D\big(P(X,Y),\theta\big) + P(U)D\big(P(X),\theta\big). \tag{8}$$

From this we can find a set of upper and lower bounds on $D(P(X,Y),\theta)$ in terms of $D(P(X,Z),\theta)$ alone. The upper bound can be found by noting $D(P(X),\theta) \geq 0$, which given Equation (8) implies

$$D(P(X,Z),\theta) \geq P(\bar{U})D\big(P(X,Y),\theta\big) \tag{9}$$

which when rearranged gives the upper bound in Equation (7). The lower bound follows similarly, by noting that $D(P(X,Y),\theta) = D(P(Y|X),\theta) + D(P(X),\theta) \geq D(P(X),\theta)$. Again using Equation (8) this gives

$$\begin{aligned} D(P(X,Z),\theta) &\leq P(\bar{U})D\big(P(X,Y),\theta\big) + P(U)D\big(P(X,Y),\theta\big) \\ &= \big(P(\bar{U}) + P(U)\big)D\big(P(X,Y),\theta\big) \\ &= D\big(P(X,Y),\theta\big) \end{aligned} \tag{10}$$

which gives us our lower bound. Combining Equation (10) and Equation (9) gives us Equation (7). ■

It is notable that in deriving these bounds we have treated $D(P(X),\theta))$ (or, equivalently, $D(P(Y|X),\theta)))$ simply as a value in the range 0 to $D(P(X,Y),\theta)$. As we wished to find general bounds that would hold for any combination of $P(X,Y)$ and $P(X,Y|\theta)$ we feel that this is an entirely justifiable method of proceeding.

In practice, however, we will not be dealing with arbitrary distributions for $P(X,Y)$ and $P(X,Y|\theta)$; rather, $P(X,Y)$ will usually represent some measurements of a real world phenomenon that we believe to be learnable in (hopefully) some well chosen space. Similarly our model may have been selected from a pool of potential models are that which is considered most likely (according to some prior beliefs) to be able to fit to the distribution of interest acceptably well, and will also often be smoothly varying with non-negligible correlations between $P(Y|X,\theta)$ and $P(X|\theta)$. As such, with additional problem specific knowledge, we suspect that tighter bounds on $D(P(X,Y),\theta)$ will tend to exist.

Another question one might raise is whether the lower bound can become tight even in instances where there is a mismatch between the model and true distribution - that is, given $\min_\theta D(P(X,Y),\theta) > 0$, can we have the situation where $D(P(X,Z),\theta) = D(P(X,Y),\theta)$? To answer this, consider $D(P(X,Z),\theta)$ as written in Equation (8). This can be re-written as follows

$$D(P(X,Z),\theta) = D\big(P(X,Y),\theta\big) - P(U)D\big(P(Y|X),\theta\big).$$

As such, in order to achieve the situation where $D(P(X,Z),\theta) = D\big(P(X,Y),\theta\big)$, it must be the case that $P(U)D\big(P(Y|X),\theta\big) = 0$. Assuming $P(U) > 0$ (as otherwise we are dealing with the trivial case of utilising no labelled data) then this must mean $D(P(Y|X),\theta) = 0$, that is, the conditional distribution specified by the model perfectly matches the true distribution. This observation seems to match intuition - if the model can correctly predict the class of unlabelled data, then its divergence estimate will not be biased by utilising these samples.

3.4 Global Bounds

The results in 3.3 give us bounds on $D(P(X,Y),\theta)$ in terms of $D(P(X,Z),\theta)$ for a given θ. It is of more interest however to characterise the global minimisers of these two expressions.

That is, if we make use of our unlabelled data to minimise $D(P(X,Z),\theta)$ with respect to θ, what can be inferred about the value of $D(P(X,Y),\theta)$ evaluated at this minimum?

Theorem 7 *Define the optimum parameters for the supervised and ML semi-supervised learning problems as*

$$\theta^\star = \arg\min_\theta D(P(X,Y),\theta),$$
$$\theta^\star_S = \arg\min_\theta D(P(X,Z),\theta).$$

Subject to the conditions in 3.1, it can be shown that

$$D(P(X,Y),\theta^\star) \leq D(P(X,Y),\theta^\star_S) \leq \frac{D(P(X,Y),\theta^\star)}{P(\bar{U})}. \tag{11}$$

That is, the divergence minimised by supervised learning, $D(P(X,Y),\theta)$, evaluated at the parameters which minimise the semi-supervised divergence, $\theta^\star_S$, can be upper and lower bounded as a function of said divergence evaluated at its own optima, $\theta^\star$.

Proof The lower bound

$$D(P(X,Y),\theta^\star) \leq D(P(X,Y),\theta^\star_S)$$

is true by the definition of θ^* - it is the minimiser of $D(P(X,Y),\theta)$, and so any other value of θ must result in a greater than or equal divergence.

The upper bound can be derived as follows. Consider the term $D(P(X,Y),\theta^\star_S)P(\bar{U})$. Using Equation (9) evaluated at $\theta = \theta^\star_S$ we can see the following,

$$D(P(X,Y),\theta^\star_S)P(\bar{U}) \leq D(P(X,Z),\theta^\star_S).$$

Given the definition of θ^*_S we can further see that

$$D(P(X,Z),\theta^\star_S) \leq D(P(X,Z),\theta^\star).$$

And using Equation (10) evaluated at $\theta = \theta^\star$,

$$D(P(X,Z),\theta^\star) \leq D(P(X,Y),\theta^\star).$$

Hence, utilising all three of these inequalities in that order,

$$\begin{aligned} D(P(X,Y),\theta^\star_S)P(\bar{U}) &\leq D(P(X,Z),\theta^\star_S) \\ &\leq D(P(X,Z),\theta^\star) \\ &\leq D(P(X,Y),\theta^\star) \end{aligned}$$

we see that

$$D(P(X,Y),\theta^\star_S)P(\bar{U}) \leq D(P(X,Y),\theta^\star).$$

By dividing through by $P(\bar{U})$ we achieve our upper bound in Equation (11), that is,

$$D(P(X,Y),\theta^\star_S) \leq \frac{D(P(X,Y),\theta^\star)}{P(\bar{U})}.$$

■

Thus, we can place bounds on divergence $D(P(X,Y),\theta)$ evaluated at θ^*_S in terms of the proportion of $P(\bar{U})$, and $D(P(X,Y),\theta^*)$. One immediate observation is that if $D\big(P(X,Y,\theta^\star\big) = 0$, then $D\big(P(X,Y),\theta^\star_S\big) = 0$. Thus, if the true distribution lies within the family of distributions expressible by our model, then the optima intersect regardless of $P(U)$, as confirmed by Cozman et al. (2003). Conversely, if $D\big(P(X,Y),\theta^\star\big) > 0$ then our bounds loosen as $P(U)$ grows, and the rate of this depends on how well matched our model is to the data - if they are very similar then the bound grows slowly, whereas if they are different it may grow much faster. This confirms earlier results (see Section 2.2 in Zhu, 2005, for a summary), and builds on them by providing explicit bounds on how rapidly performance may degrade.

The overall conclusion is that performing ML semi-supervised learning in the manner of Equation (3) forces us to make a trade off. We can rarely evaluate KL divergences directly, and must use estimators whose variance is inversely proportional to N (MacKay, 2003). By including unlabelled data we can decrease this source of uncertainty. However in doing so we weaken our bounds, introducing a new source of error. This provides a complementary reinterpretation of the results noted by Cozman et al. (2003).

As our bounds weaken then, how does our solution degrade? We now show that the supervised divergence, evaluated at the ML semi-supervised optima, grows monotonically with the proportion of unlabelled samples.

Theorem 8 *Subject to the conditions in 3.1, let us define two distributions $P_1(X,Z)$ and $P_2(X,Z)$, and corresponding models $P_1(X,Z|\theta)$ and $P_2(X,Z|\theta)$. These distributions shall differ from one another only in terms of the probability that $Z = U$; that is, $P_1(X,Y) = P_2(X,Y)$ and $P_1(X,Y|\theta) = P_2(X,Y|\theta)$ (which in turn implies $P_1(X) = P_2(X)$ and $P_1(X|\theta) = P_2(X|\theta)$). We shall assume that distribution $P_2(X,Z)$ has a greater chance of an unlabelled sample, and so $P_2(U) > P_1(U)$.*

Define the optima $\theta^\star_{S1}$ and $\theta^\star_{S2}$ as

$$\theta^\star_{S1} = \arg\min_\theta D(P_1(X,Z),\theta), \quad \theta^\star_{S2} = \arg\min_\theta D(P_2(X,Z),\theta). \tag{12}$$

It follows that

$$D(P(X,Y),\theta^\star_{S1}) \leq D(P(X,Y),\theta^\star_{S2}) \tag{13}$$

and

$$D(P(Y|X),\theta^\star_{S1}) \leq D(P(Y|X),\theta^\star_{S2}). \tag{14}$$

Proof By definition,

$$D(P_1(X,Z),\theta^*_{S1}) \leq D(P_1(X,Z),\theta^*_{S2}).$$

We can expand both these divergences to rewrite this expression as follows;

$$\begin{aligned} &P_1(\bar{U})D(P(X,Y),\theta^\star_{S1}) + P_1(U)D(P(X),\theta^\star_{S1}) \\ &\leq P_1(\bar{U})D(P(X,Y),\theta^\star_{S2}) + P_1(U)D(P(X),\theta^\star_{S2}) \end{aligned}$$

Rearranging this expression to isolate $D(P(X), \theta^\star_{S1}) - D(P(X), \theta^\star_{S2})$ gives us

$$D(P(X), \theta^\star_{S1}) - D(P(X), \theta^\star_{S2}) \leq \frac{P_1(\bar{U})}{P_1(U)} \left(D(P(X,Y), \theta^\star_{S2}) - D(P(X,Y), \theta^\star_{S1}) \right). \tag{15}$$

We shall utilise this term later.

Now examining the divergences associated with the distribution $P_2(X, Z)$, by the definition given in Equation (12) we see that

$$D(P_2(X,Z), \theta^*_{S2}) \leq D(P_2(X,Z), \theta^*_{S1}).$$

This can be expanded as before,

$$\begin{aligned} &P_2(\bar{U}) D(P(X,Y), \theta^\star_{S2}) + P_2(U) D(P(X), \theta^\star_{S2}) \\ &\leq P_2(\bar{U}) D(P(X,Y), \theta^\star_{S1}) + P_2(U) D(P(X), \theta^\star_{S1}), \end{aligned}$$

and $D(P(X), \theta^\star_{S1}) - D(P(X), \theta^\star_{S2})$ once again isolated,

$$\frac{P_2(\bar{U})}{P_2(U)} \left(D(P(X,Y), \theta^\star_{S2}) - D(P(X,Y), \theta^\star_{S1}) \right) \leq D(P(X), \theta^\star_{S1}) - D(P(X), \theta^\star_{S2}). \tag{16}$$

Combining Equation (15) and Equation (16) to eliminate $D(P(X), \theta^\star_{S1}) - D(P(X), \theta^\star_{S2})$ gives us

$$\begin{aligned} &\frac{P_2(\bar{U})}{P_2(U)} \left(D(P(X,Y), \theta^\star_{S2}) - D(P(X,Y), \theta^\star_{S1}) \right) \\ &\leq \frac{P_1(\bar{U})}{P_1(U)} \left(D(P(X,Y), \theta^\star_{S2}) - D(P(X,Y), \theta^\star_{S1}) \right). \end{aligned}$$

Gathering together similar divergences, this implies that

$$\left(\frac{P_1(\bar{U})}{P_1(U)} - \frac{P_2(\bar{U})}{P_2(U)} \right) D(P(X,Y), \theta^\star_{S1}) \leq \left(\frac{P_1(\bar{U})}{P_1(U)} - \frac{P_2(\bar{U})}{P_2(U)} \right) D(P(X,Y), \theta^\star_{S2})$$

As we know that $P_2(U) > P_1(U)$, and so $P_2(\bar{U}) < P_1(\bar{U})$, it follows that $P_2(U)P_1(\bar{U}) > P_1(U)P_2(\bar{U})$, which in turn implies

$$\frac{P_1(\bar{U})}{P_1(U)} - \frac{P_2(\bar{U})}{P_2(U)} > 0.$$

As it is positive we may cancel this term out without altering the inequality, indicating that

$$D(P(X,Y), \theta^\star_{S1}) \leq D(P(X,Y), \theta^\star_{S2})$$

proving Equation (13).

To prove Equation (14), note that if we take Equation (15), multiply though by $P_1(U)$, and then use some simple algebra to gather all terms relating to the marginal divergence together, it is equivalent to stating

$$D(P(X), \theta^\star_{S1}) - D(P(X), \theta^\star_{S2}) \leq P_1(\bar{U}) \left(D(P(Y|X), \theta^\star_{S2}) - D(P(Y|X), \theta^\star_{S1}) \right).$$

Similarly, Equation (16) can be rearranged as

$$P_2(\bar{U})\left(D(P(Y|X),\theta^\star_{S2}) - D(P(Y|X),\theta^\star_{S1})\right) \leq D(P(X),\theta^\star_{S1}) - D(P(X),\theta^\star_{S2}).$$

Combining these two, we see that

$$\begin{aligned}&P_2(\bar{U})\left(D(P(Y|X),\theta^\star_{S2}) - D(P(Y|X),\theta^\star_{S1})\right)\\ &\leq P_1(\bar{U})\left(D(P(Y|X),\theta^\star_{S2}) - D(P(Y|X),\theta^\star_{S1})\right).\end{aligned}$$

Gathering together terms, this rearranges to

$$\left(P_1(\bar{U}) - P_2(\bar{U})\right) D(P(Y|X),\theta^\star_{S1}) \leq \left(P_1(\bar{U}) - P_2(\bar{U})\right) D(P(Y|X),\theta^\star_{S2})$$

which, given $P_2(U) > P_1(U)$, and hence $P_2(\bar{U}) < P_1(\bar{U})$, implies

$$D(P(Y|X),\theta^\star_{S2}) \geq D(P(Y|X),\theta^\star_{S1})$$

proving Equation (14). ■

This observation seems intuitively reasonable. As $P(U)$ grows the model is increasingly penalised by large values of $D(P(X),\theta)$, and so seeks to minimise this at the expense of letting $D(P(Y|X),\theta)$ get larger. However, if our end goal is to create a classifier then this result may give us cause to reconsider - adding unlabelled data not only weakens our bounds on the joint divergence, but asymptotically can only worsen (or at best leave unchanged) the conditional divergence.

Thus, we can now conclude several things about the asymptotic optimum of the ML semi-supervised learning problem. Firstly, due to the observation of monotonicity, the divergence $D(P(X,Y),\theta^*_S)$ is upper bounded by $D(P(X,Y),\theta^*_U)$, confirming Yang and Priebe (2011). Secondly, that if we were to increase the quantity of unlabelled data, it will tend towards this approaching equality as $P(U)$ tends towards 1. Finally, it will do so monotonically - raising the proportion of unlabelled data will never decrease $D(P(X,Y),\theta^*_S)$ or $D(P(Y|X),\theta^*_S)$.

This result initially seems to contradict that of Cozman and Cohen (2006), where they gave an example of a ML semi-supervised learning process where despite the model not fitting the underlying distribution, adding unlabelled data asymptotically improved the decision boundary. We point out though that their measure of how well the boundary fits is based on the error rate, not the KL divergence. while it is true that minimising the conditional KL divergence will typically reduce the error rate this is not an absolute rule (and indeed forms a set of bounds). We would postulate that this is an example of a case where the divergence rises but the classification rate improves.

Finally, it makes sense to more closely examine the final solution arrived at as $P(U) \to 1$, in a similar manner to that discussed by Yang and Priebe (2011). In particular, we wish to confirm that their result extend beyond identifiable models, and shall show that where there is a choice between multiple sets of parameters which minimise the unsupervised divergence $D(P(X),\theta)$, the semi-supervised divergence minimised by ML learning is upper bounded by the one set of these parameters which best minimises $D(P(Y|X),\theta)$. This proof is largely similar to that showing monotonicity but is included here for completeness.

Theorem 9 *Subject to the conditions in 3.1, define the optimum unsupervised parameters $\theta^\star_U$ to be any parameters which meet these requirements:*

$$\theta^\star_U = \arg\min_\theta D(P(Y|X),\theta) \quad \textit{subject to} \quad D(P(X),\theta^\star_U) = \min_{\theta'} D(P(X),\theta')$$

It can be shown that provided $P(\bar{U}) \neq 0$,

$$D(P(X,Y),\theta^\star_S) \leq D(P(X,Y),\theta^\star_U) \tag{17}$$

and

$$D(P(Y|X),\theta^\star_S) \leq D(P(Y|X),\theta^\star_U) \tag{18}$$

Proof By definition,

$$D(P(X,Z),\theta^\star_S) \leq D(P(X,Z),\theta^\star_U). \tag{19}$$

The standard semi-supervised divergence can be expanded as follows,

$$D(P(X,Z)|\theta) = P(\bar{U})D\big(P(X,Y),\theta\big) + P(U)D\big(P(X),\theta\big).$$

As such, we can rewrite Equation (19) as follows

$$\begin{aligned}&P(\bar{U})D(P(X,Y),\theta^\star_S) + P(U)D(P(X),\theta^\star_S)\\ &\leq P(\bar{U})D(P(X,Y),\theta^\star_U) + P(U)D(P(X),\theta^\star_U).\end{aligned}$$

If we subtract $P(U)D(P(X),\theta^\star_U)$ from both sides this becomes

$$\begin{aligned}&P(\bar{U})D(P(X,Y),\theta^\star_S) + P(U)\left(D(P(X),\theta^\star_S) - D(P(X),\theta^\star_U)\right)\\ &\leq P(\bar{U})D(P(X,Y),\theta^\star_U).\end{aligned}$$

However, by definition, $D(P(X),\theta^\star_S) \geq D(P(X),\theta^\star_U)$, implying

$$P(\bar{U})D(P(X,Y),\theta^\star_S) \leq P(\bar{U})D(P(X,Y),\theta^\star_U)$$

and hence Equation (17) directly follows by dividing through by $P(\bar{U})$. Equation (18) follows similarly by noting that Equation (17) implies

$$D(P(Y|X),\theta^\star_S) + D(P(X),\theta^\star_S) \leq D(P(Y|X),\theta^\star_U) + D(P(X),\theta^\star_U).$$

If we subtract $D(P(X),\theta^\star_U)$ from both sides we find that

$$D(P(Y|X),\theta^\star_S) + D(P(X),\theta^\star_S) - D(P(X),\theta^\star_U) \leq D(P(Y|X),\theta^\star_U)$$

and again note that by definition, $D(P(X),\theta^\star_S) \geq D(P(X),\theta^\star_U)$, and hence

$$D(P(Y|X),\theta^\star_S) \leq D(P(Y|X),\theta^\star_U)$$

directly follows, proving Equation (18).

■

Note that the above derivation could have proceeded in exactly the same manner with $\theta^\star_U$ chosen to be any parameters for which $D(P(X), \theta^\star_U) = \min_{\theta'} D(P(X), \theta')$. However, by choosing $\theta^\star_U$ to be the parameters which also minimised $D(P(Y|X), \theta^\star_U)$ we attain as tight a bound as possible.

For many models there will be only one set of parameters which minimise $D(P(X), \theta)$, and so this is not an issue. However for others this will not be the case. For example, many mixture models contain mixture components which are identical save for the class they are assigned to. In these cases, specifying that $\theta^\star_U$ have the lowest conditional divergence amongst those set of parameters which have the minimum marginal divergence allows us to choose the best combination of class assignment for each mixture component given their other parameters, strengthening slightly the conclusions of Yang and Priebe (2011).

We can now rewrite our global bounds as follows:

$$D(P(X,Y), \theta^\star) \leq D(P(X,Y), \theta^\star_S) \leq \min\left(\frac{D(P(X,Y), \theta^\star)}{P(\bar{U})}, D(P(X,Y), \theta^\star_U)\right).$$

Assuming that $D(P(Y|X), \theta^\star_U) < \infty$, which will be the case provided $P(Y|X, \theta^\star_U)$ does not assign zero probability to any Y given any X, this gives tighter performance bounds as $P(\bar{U}) \to 0$.

3.5 Summary

This ends our theoretical examination of performing ML learning on a partially labelled data set. Overall we can conclude the following;

- When we introduce unlabelled data into our likelihood expression, we change the divergence being minimised, from $D(P(X,Y), \theta)$ to $D(P(X,Z), \theta)$.
- We can form a set of upper and lower bounds on $D(P(X,Y), \theta)$ using $D(P(X,Z), \theta)$ for a given θ, namely

  $$D(P(X,Z), \theta) \leq D(P(X,Y), \theta) \leq \frac{D(P(X,Z), \theta)}{P(\bar{U})}.$$

  The lower bound becomes tight if $D(P(Y|X), \theta)$ is equal to 0, that is, if our model is fits to the conditional distribution well.
- If we find the parameters $\theta^\star_S$ which minimise the standard semi-supervised divergence $D(P(X,Z), \theta)$, then these are linked to the parameters $\theta^\star$ which minimise $D(P(X,Y), \theta)$ using the expression

  $$D(P(X,Y), \theta^\star) \leq D(P(X,Y), \theta^\star_S) \leq \frac{D(P(X,Y), \theta^\star)}{P(\bar{U})},$$

  that is, our supervised divergence evaluated at the standard semi-supervised minima may exceed the supervised minima by a factor of $1/(P(\bar{U}))$. Where there is a large quantity of unlabelled data this factor may be very high.
- $D(P(X,Y), \theta^\star_S)$ grows monotonically with the proportion of unlabelled data $P(U)$. Moreover, it is the term $D(P(Y|X), \theta^\star_S)$ which grows, indicating that we can expect classification results to remain steady or worsen.

Taken together, this gives a clear indication of the problem we face conducting generative semi-supervised ML learning, and gives novel bounds on the asymptotic performance achievable.

4. Unbiased Generative Semi-Supervised Learning

Having investigated the properties of $D(P(X,Z),\theta)$, it is clear that if we wish to minimise $D(P(X,Y),\theta)$, it is better we find an unbiased likelihood estimator. From examination of the form of the supervised divergence, we propose the following.

Theorem 10 *Subject to the conditions in 3.1, and provided $P(\bar{U}) > 0$, the expression*

$$\arg\max_{\theta} \prod_{i=1}^{N_L} P(y_i|x_i,\theta)\Big(\prod_{i=1}^{N} P(x_i|\theta)\Big)^{N_L/N} \tag{20}$$

returns a set of parameters which minimise an asymptotically unbiased estimator of the divergence $D(P(X,Y),\theta)$.

Proof The divergence $D(P(X,Y),\theta)$ is exactly equivalent to the following:

$$D(P(Y|X),\theta) + D(P(X),\theta). \tag{21}$$

We draw samples $(x_i,y_i)_{i=1,\ldots,N_L}$ and $(x_i)_{i=N_L+1,\ldots,N}$. Assuming that as $N\to\infty$, $N_L/N \to P(\bar{U})$, we can use these to construct an asymptotically unbiased estimator of the divergence Equation (21),

$$\left(\frac{1}{N_L}\sum_{i=1}^{N_L}\log\left(\frac{P(y_i|x_i)}{P(y_i|x_i,\theta)}\right) + \frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{P(x_i)}{P(x_i|\theta)}\right)\right). \tag{22}$$

Disregarding all terms which are not a function of θ gives the expression

$$\left(\frac{-1}{N_L}\sum_{i=1}^{N_L}\log\left(P(y_i|x_i,\theta)\right) + \frac{-1}{N}\sum_{i=1}^{N}\log\left(P(x_i|\theta)\right)\right). \tag{23}$$

Multiplying this by $-N_L$ and taking the antilog yields

$$\prod_{i=1}^{N_L} P(y_i|x_i,\theta)\prod_{i=1}^{N} P(x_i|\theta)^{N_L/N}.$$

This is the quantity which is maximised in Equation (20). As the log function is monotonic, the parameters which maximise this will minimise Equation (23) (due to the multiplication by $-N_L$). Minimising Equation (23) is equivalent to minimising Equation (22) as they only differ by terms that are constant with respect to the parameters. And Equation (22) is an unbiased estimator of Equation (21). Hence, the parameters returned by Equation (20) are equivalent to those which minimise an unbiased estimator of $D(P(X,Y),\theta)$. ■

A special case occurs when $P(\bar{U}) = 0$. This corresponds to Equation (22) where N_L is fixed while $N_U \to \infty$,

$$\arg\min_{\theta} \frac{1}{N_L} \sum_{i=1}^{N_L} \log\left(\frac{P(y_i|x_i)}{P(y_i|x_i,\theta)}\right) + D(P(X),\theta) \tag{24}$$

which estimates the marginal component of the divergence exactly while using the available labelled data to estimate the conditional component as best possible.

Our term Equation (20) is somewhat similar to the form of Equation 3 presented by McCallum et al. (2006), which was further investigated by Druck et al. (2007), but with the exponents of the conditional and generative components of the equation set by the ratio of labelled to unlabelled data, rather than being found by cross validation. Moreover, our purpose in using an equation of this form is different; we wish to fit a generative model, not a classifier. Rosset et al. (2005) has also previously noted that performance can be improved by requiring certain expectations in the labelled and unlabelled data set match. However, they enforced this as a strict requirement, rather than using it to find an unbiased likelihood estimate as we do. Nigam et al. (2000) implement down-weighting of the log likelihood of all unlabelled elements, by a factor which is set using cross validation. As the marginal likelihood of the labelled samples is not re-weighed this also produces a biased estimator of the joint likelihood.

An argument might be made that a biased estimator which is tuned using cross validation has the potential to outperform the proposed unbiased objective function. While there is certainly merit to this, we would respond that the parameter tuning inherent in cross validation can increase the amount of time spent training dramatically, and that it requires a large enough corpus of labelled data that a holdout set can be safely put aside to validate with. Our objective function provides a simple, principled alternative, applicable in cases where such restrictions prevent cross validation, as well as others.

4.1 Estimator Variance

Note that we can already generate an asymptotically unbiased estimate by using the labelled data alone. Unlabelled samples are only of value if they make this process more reliable, so it is worth investigating the uncertainty of this estimator. Consider the variance V of Equation (22), where the variance is taken w.r.t. the probability of the possible data sets we may have observed

$$V = \mathrm{Var}\left(\frac{1}{N_L}\sum_{i=1}^{N_L} \log\left(\frac{P(y_i|x_i)}{P(y_i|x_i,\theta)}\right) + \frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{P(x_i)}{P(x_i|\theta)}\right)\right). \tag{25}$$

We can expand Equation (25) as

$$\begin{aligned} V \;&= \mathrm{Var}\left(\frac{1}{N_L}\sum_{i=1}^{N_L} L_{y|x}(i)\right) + \mathrm{Var}\left(\frac{1}{N}\sum_{i=1}^{N} L_x(i)\right) \\ &\quad + 2\,\mathrm{Cov}\left(\frac{1}{N_L}\sum_{i=1}^{N_L} L_{y|x}(i), \frac{1}{N}\sum_{i=1}^{N} L_x(i)\right) \end{aligned} \tag{26}$$

where for ease of notation we have defined

$$L_{y|x}(i) \equiv \log\left(\frac{P(y_i|x_i)}{P(y_i|x_i,\theta)}\right), \quad L_x(i) \equiv \log\left(\frac{P(x_i)}{P(x_i|\theta)}\right).$$

Using standard identities for variances and covariances,[2] and taking advantage of our samples being iid, we can expand Equation (26) as follows

$$V = \frac{1}{N_L}\operatorname{Var}\left(L_{Y|X}\right) + \frac{1}{N}\operatorname{Var}\left(L_X\right) + \frac{2}{N}\operatorname{Cov}\left(L_{Y|X}, L_X\right) \tag{27}$$

where we have now defined

$$L_{Y|X} \equiv \log\left(\frac{P(Y|X)}{P(Y|X,\theta)}\right), \quad L_X \equiv \log\left(\frac{P(X)}{P(X|\theta)}\right).$$

As such Equation (27) gives us the variance of Equation (22) in terms of a relationship between the distributions $P(X,Y)$, $P(X,Y|\theta)$, N_L and N_U. Clearly $V \to 0$ as $N_L \to \infty$. However we are also interested in the case where we increase N_U while holding N_L steady, corresponding to $P(\bar{U}) = 0$. By inspection, as $N_U \to \infty$, $V \to \frac{1}{N_L}\operatorname{Var}\left(L_{Y|X}\right)$ (as one might expect from examination of Equation (24)), so the question becomes whether this reduces V. Remembering that $N = N_L + N_U$, the derivative[3] of Equation (27) with respect to N (and hence N_U since N_L is fixed) is

$$\frac{dV}{dN} = \frac{-1}{N^2}\operatorname{Var}\left(L_X\right) - \frac{2}{N^2}\operatorname{Cov}\left(L_{Y|X}, L_X\right).$$

2. The simplification of the variance terms is intuitively obvious and a standard result - for any random variable, we expect the variance of the arithmetic mean of a set of observations to be the variance of the variable itself, divided by the number of observations made (see MacKay, 2003). However the covariance term is perhaps a little more surprising, as it has no dependence on N_L. This is due to the iid nature of the data, which implies a covariance of zero between different samples. As such we can derive the following:

$$\begin{aligned}
&\operatorname{Cov}\left(\frac{1}{N_L}\sum_{i=1}^{N_L} L_{y|x}(i), \frac{1}{N}\sum_{i=1}^{N} L_x(i)\right) \\
&= \frac{1}{NN_L}\sum_{i=1}^{N_L}\sum_{j=1}^{N}\operatorname{Cov}\left(L_{y|x}(i), L_x(j)\right) \\
&= \frac{1}{NN_L}\sum_{i=1}^{N_L}\sum_{\substack{j=1\\ j\neq i}}^{N}\operatorname{Cov}\left(L_{y|x}(i), L_x(j)\right) + \frac{1}{NN_L}\sum_{i=1}^{N_L}\operatorname{Cov}\left(L_{y|x}(i), L_x(i)\right) \\
&= 0 + \frac{1}{NN_L}\sum_{i=1}^{N_L}\operatorname{Cov}\left(L_{y|x}(i), L_x(i)\right) \\
&= \frac{1}{NN_L}\sum_{i=1}^{N_L}\operatorname{Cov}\left(L_{Y|X}, L_X\right) \\
&= \frac{1}{N}\operatorname{Cov}\left(L_{Y|X}, L_X\right)
\end{aligned}$$

which eliminates N_L and so gives us the stated form of the covariance.

3. Strictly N_U is discrete, and does not have a derivative. However the expression is monotonic in N_U, and so the overall trend will be the same if this constraint is relaxed.

For Equation (25) to decrease as N_U increases this quantity must be negative. This is the case iff

$$\mathrm{Cov}\left(L_{Y|X}, L_X\right) \geq \frac{-\mathrm{Var}\left(L_X\right)}{2}.$$

The conclusion is that even if N_L is fixed our method of including unlabelled data reduces the variance of our estimator provided $\mathrm{Cov}\left(L_{Y|X}, L_X\right)$ is above a lower bound, proportional to $\mathrm{Var}\left(L_X\right)$. Perhaps surprisingly this bound is negative, indicating they may be slightly anti-correlated. We feel this is a sufficiently weak criteria for our scheme to find application across a variety of data sets.

5. Empirical Demonstration

We now examine the performance of the objective function given in Section 4 on real world data sets, compared to the standard semi-supervised learning, supervised learning, and several other alternative semi-supervised techniques. To maximally highlight the effect of mismatch between the model and true distribution, a simple marginal distribution consisting of a single axis aligned Gaussian was chosen to model each class.

The following six learning schemes were tested with this model: our unbiased semi-supervised expression (**SSunb**), that is, the natural log of Equation (20); the log likelihood of the labelled data (**LL**), that is, Equation (1); the log likelihood of the standard (biased) semi-supervised expression (**SSb**), that is, the natural log of Equation (3); the log likelihood of the standard semi-supervised expression plus an Entropy Regularisation term (Grandvalet and Bengio, 2006) with the parameter λ set by 5 fold cross validation, selecting the λ with the lowest holdout set error rate (**ERer**); Entropy Regularisation as before, except cross validation is carried out on the log likelihood of the holdout set (**ERnll**); the semi-supervised equivalent of Multi Conditional learning (as investigated in Druck et al., 2007), again cross validating hyper parameters once on error rate (**MCer**) and once on log likelihood (**MCnll**); and the log likelihood of the standard semi-supervised expression plus an Expectation Regularisation (Mann and McCallum, 2007) term (**XR**), with the trade off parameter set (after some experimentation) as in the original paper to the equivalent of 10 times the number of labelled samples; Additionally, for the position parameter μ of each Gaussian a penalty term $-C||\mu||^2$ was added onto each objective function with C set to a small constant ($\approx 10^{-5}$).

We would point out that many of these learning schemes were originally designed for use with a discriminative model. Here we are using them in a different manner, to augment the objective function during the learning of a generative model. They have been selected due to their reported good performance in improving discriminative learning, in the hope that this will counteract the bias introduced by the missing class information in the likelihood of the unlabelled samples.

We chose 7 data sets from the UCI repository (Frank and Asuncion, 2010); **Diabetes**, **Wine**, glass identification (**Glass**), blood transfusion (**Blood**) (Yeh et al., 2009), **Ecoli**, Haberman survival (**Haber**), and Pima Indian diabetes (**Pima**); and 2 from libsvm: SVM guide 1 (**SVMg**) (Hsu et al., 2003) and fourclass (**Four**) (Ho and Kleinberg, 1996). Due to computational constraints, data sets with > 3 classes had one or more merged to create 3 approximately equally sized groupings. Each axis of the data was transformed to lie in

data	SSunb	LL	SSb	MCer	MCnll	ERer	ERnll	XR
Diabetes	**3.36**	4.12	3.90	144	<u>3.58</u>	3.90	3.97	3.74
SVMg	<u>0.379</u>	0.417	1.18	99.4	**0.376**	1.23	1.19	1.16
Wine	<u>19.4</u>	58.4	23.0	67.2	21.4	24.7	24.7	**12.5**
Glass	23.7	40.9	23.5	213	26.3	22.7	**21.5**	<u>21.5</u>
Blood	**1.78**	2.27	3.01	77.2	<u>2.08</u>	3.06	3.06	2.65
ecoli	**8.80**	13.7	10.0	68.6	10.3	9.97	10.0	<u>9.63</u>
Haber	<u>4.75</u>	7.30	5.02	79.8	5.10	4.98	4.90	**4.59**
Pima	**3.60**	4.30	4.24	136	<u>3.80</u>	4.26	4.25	3.87
Four	**2.17**	2.22	2.25	37.3	<u>2.19</u>	2.33	2.32	2.23

Table 1: Overall mean negative log likelihood - best result for each data set shown in **bold**, second best <u>underlined</u>

the range $[-1, 1]$. Samples with missing attributes were excluded. Where a data set had a dedicated test set, this was used; otherwise, one fifth of the data was randomly separated a priori for this purpose.

A range of values of N_L and N_U were trialled. As a proportion of the total available training data, N_L varied from $[0.025, 0.05, 0.1, 0.2]$, and N_U from $[0.025, 0.05, 0.1, 0.2, 0.4, 0.8]$, with N_U being formed by discarding labels prior to training (for example, a test where $N_L = 0.05$ and $N_U = 0.4$ would indicate 0.45 of the available data was used for training, of which one ninth was labelled). For each repetition a random set of parameters was generated and used as the starting point for each of the above learning schemes. Each model was optimised by repeatedly alternating between a small number of iterations of downhill simplex search (Lagarias et al., 1998), followed by a large numbers of iterations of BFGS search (Nocedal and Wright, 1999), until convergence. This process was repeated 100 times for each combination of N_L and N_U values. The error rate and negative log likelihood of the test set was found for each solution. A selection of these results are shown here. Full results over all test sets are included in the appendix.

Note that we have purposefully used the same optimisation scheme for all objective functions - including **LL**, which has a closed form solution, and **SSb**, which can be optimised using expectation maximisation. Also note that for each repetition a single set of starting parameters was randomly generated, and then used to initialise every learning scheme investigated. The intent of this was to ensure that all variability encountered was solely due to the choice of objective function.

Table 1 shows the mean negative log likelihood for each data set, that is, the negative log likelihood averaged over all all repetitions of all values of N_L and N_U. For each data set, the minimum negative log likelihood is shown in bold, and the second smallest underlined. Our method **SSunb** achieved the lowest mean for 5 of the 9 data sets, and the second lowest for a further 3. **XR** proved the best for two data sets, and **MCnll** and **ERnll** for one each.

Mean error rates are shown in in Table 2. Our method performed best for 3 data sets and second best in a further 4, roughly equivalent to **LL** or **MCer** (without the need for the

data	SSunb	LL	SSb	MCer	MCnll	ERer	ERnll	XR
Diabetes	**0.283**	0.284	0.332	0.299	0.294	0.337	0.346	0.312
SVMg	0.0597	0.0597	0.189	**0.0572**	0.0678	0.193	0.175	0.171
Wine	**0.144**	0.163	0.190	0.238	0.162	0.230	0.259	0.218
Glass	0.483	**0.459**	0.539	0.470	0.501	0.545	0.555	0.525
Blood	0.277	0.277	0.367	**0.273**	0.292	0.372	0.369	0.309
ecoli	0.121	**0.104**	0.266	0.140	0.147	0.276	0.280	0.184
Haber	**0.298**	0.298	0.371	0.337	0.319	0.379	0.369	0.301
Pima	0.296	**0.292**	0.348	0.302	0.306	0.355	0.358	0.327
Four	0.249	0.250	0.260	**0.245**	0.250	0.281	0.274	0.259

Table 2: Overall mean error rate - best result for each data set shown in **bold**, second best underlined

latter's expensive cross validation). We point out that we are training a simple generative model, and so error rates reported are not directly comparable to previous work using more powerful / conditional models.

Figure 1 consists of four plots, showing how the mean negative log likelihood of the **Blood** data set varies as N_U is increased, for all four values of N_L tested. Error bars indicate a single standard deviation. Note how for small values of N_U all methods perform similarly, with some benefit from using unlabelled data. As N_U increases and the upper bound weakens, the methods begin to diverge - the **ER** methods, along with **SSb** and **XR** worsen consistently. **LL** remains approximately constant (as expected) and slightly larger than **SSunb**. **MCnll** sits somewhere between **LL** and **SSunb**, worsening a little as the proportion of unlabelled data grows. This qualitative description of the observed behaviour applies to a significant proportion of the results. The main exceptions to this trend were in the **Glass**, **Wine** and **Haber** data sets for small values of N_L, where competing methods (noticeably **XR**) performed better though this advantage tended to tail off as N_L grew - for example see Figure 3, which shows the Haberman data set performing very well under **XR** training.

Figure 2 shows how the mean error rate of the same data set varies. For small quantities of labelled data SSunb tends to tie with **LL**. However as the quantity of labelled data grows competing methods begin to out perform it. In general we found that the proposed unbiased method was not always best (most commonly being out performed by **XR**), but often very competitive. It also rarely showed degradation in behaviour as the quantity of unlabelled data was increased - as we would expect, given the manner in which it automatically downgrades the influence of additional unlabelled samples.

As well as looking at the mean log likelihood and error rates though, we believe another informative measure of the success of a semi-supervised algorithm is the raw frequency with which it out performs alternate methods. This gives an estimate of the probability that, should you include unlabelled data in your training data set, the performance of the algorithm will improve.

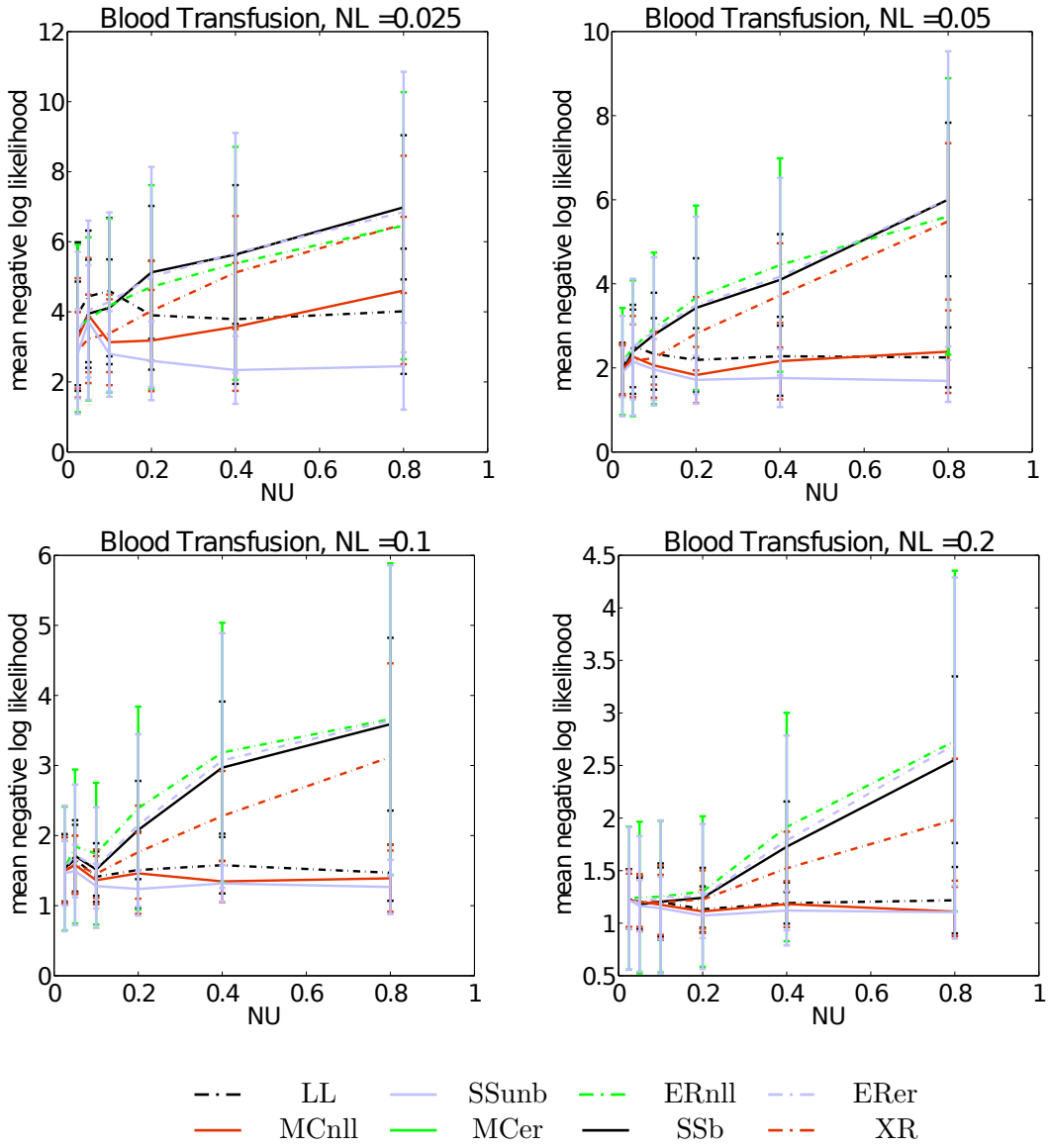


Figure 1: Four sample plots of the mean negative log likelihood of the **Blood** data set for a variety of values of N_L, as N_U grows. Note that *MCer* is excluded, as it significantly underperformed and caused unfavourable axis scaling.

An example of this is shown in Figure 4, which shows the proportion of occasions each semi-supervised algorithm out performs supervised learning, where performance is measured in terms of the negative log likelihood of the test set. For this particular case our proposed unbiased estimator is consistently the superior one - on only one occasion does another algorithm (**MCnll**) outperform supervised learning with greater frequency. In general it was found that only when N_U is small that we typically saw other methods performing

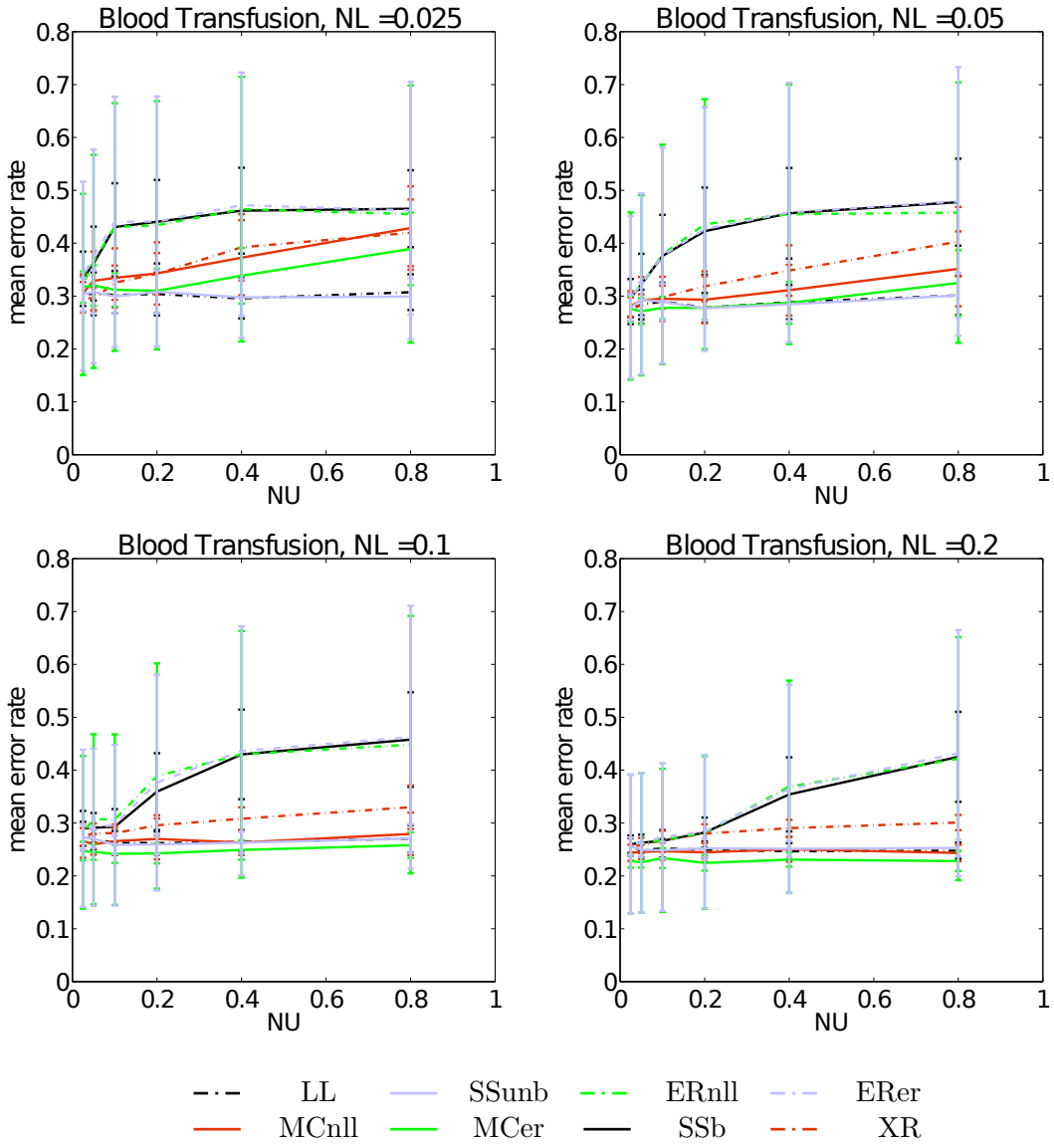


Figure 2: Four sample plots of the mean error rate of the **Blood** data set for a variety of values of N_L, as N_U grows.

better. What is also notable is how, while several other methods initially provide a bonus when N_U is small (where the proportion rises above 0.5, indicating that they were more likely than not to improve learning), they tend to degrade quite rapidly as unlabelled data is added, often making it more probable that they will worsen performance by the time $N_U = 0.8$. It was much rarer that our algorithm did this (one example occurring in the **Ecoli** data set with $N_L = 0.2$).

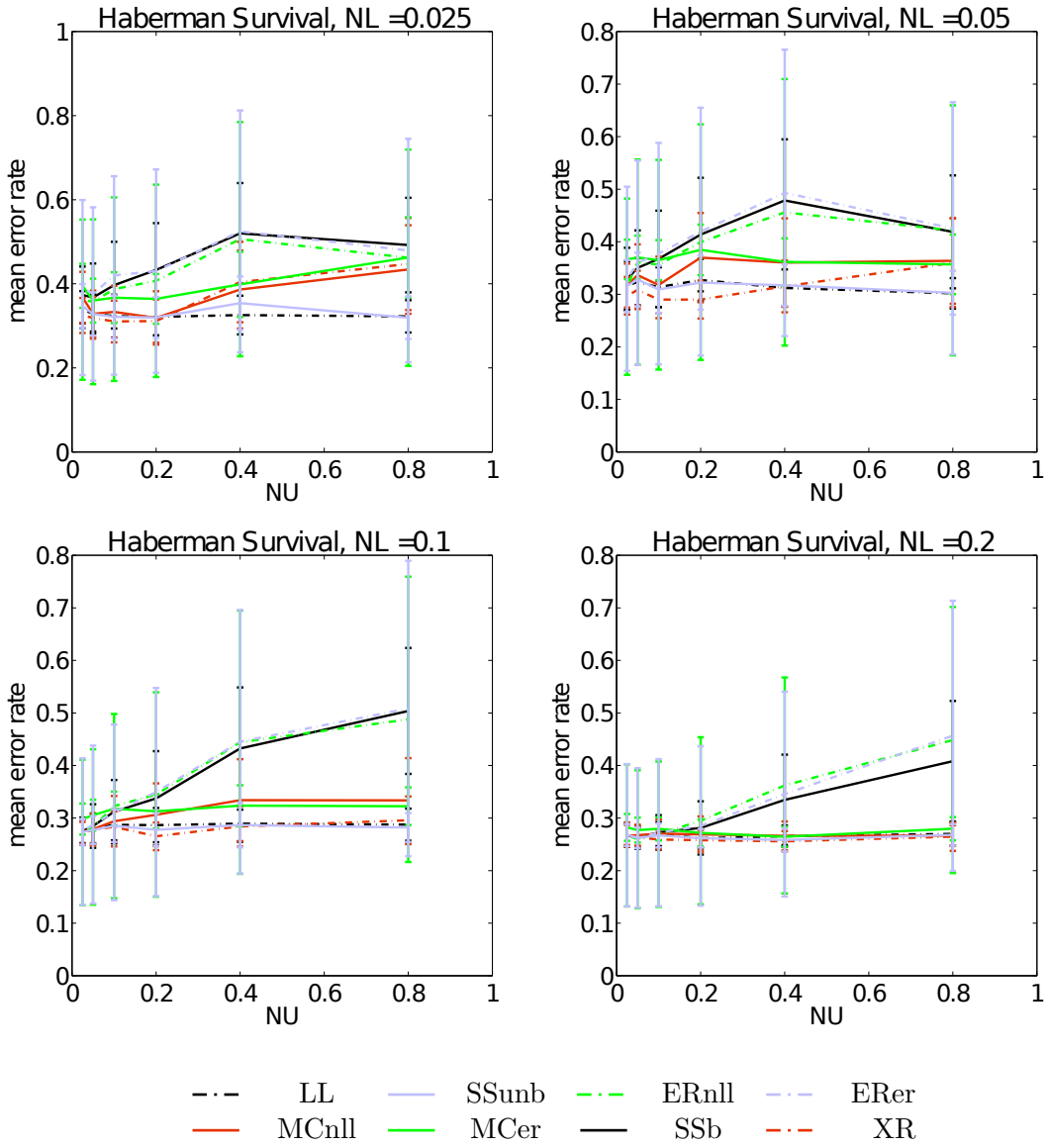


Figure 3: Four sample plots of the proportion of tests in which each semi-supervised learning scheme outperformed learning on the labelled data alone as measured by the negative log likelihood on the **Haberman** data set for a variety of values of N_L, as N_U grows.

Finally, Figure 5 shows the proportion of repetitions for which each semi-supervised algorithm reduced the error compared to supervised learning alone. Here our algorithm behaves much less impressively. With N_L set to its lowest value 0.025 it tends to be the better of the algorithms as N_U grows, but the proportion of occasions it provides a benefit is barely above 0.5. As the number of labelled data samples grows the two multi conditional

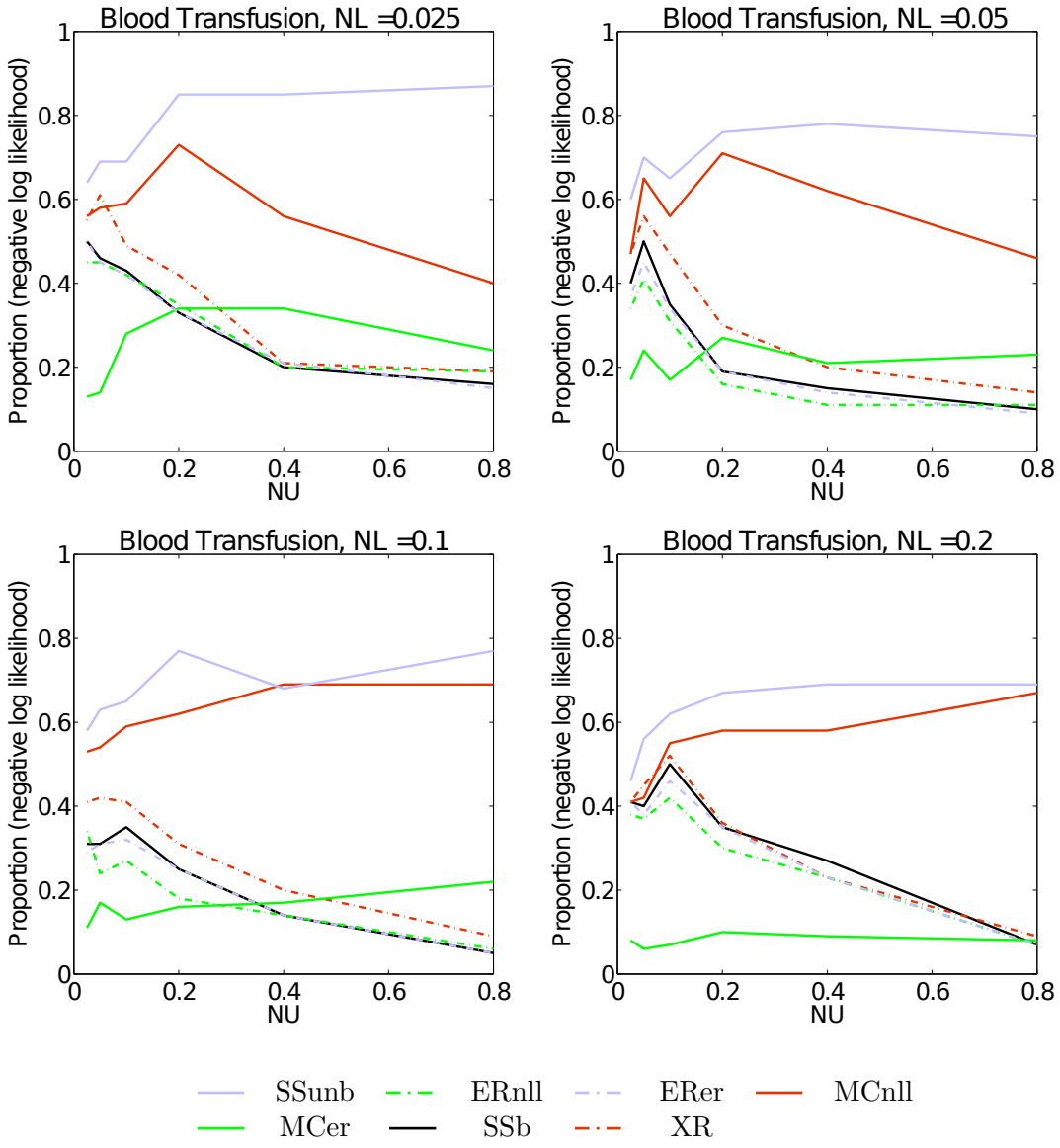


Figure 4: Four sample plots of the proportion of tests in which each semi-supervised learning scheme outperformed learning on the labelled data alone as measured by the negative log likelihood on the **Blood** data set for a variety of values of N_L, as N_U grows.

learning algorithms begin to out perform all others, especially when cross validated to reduce error.

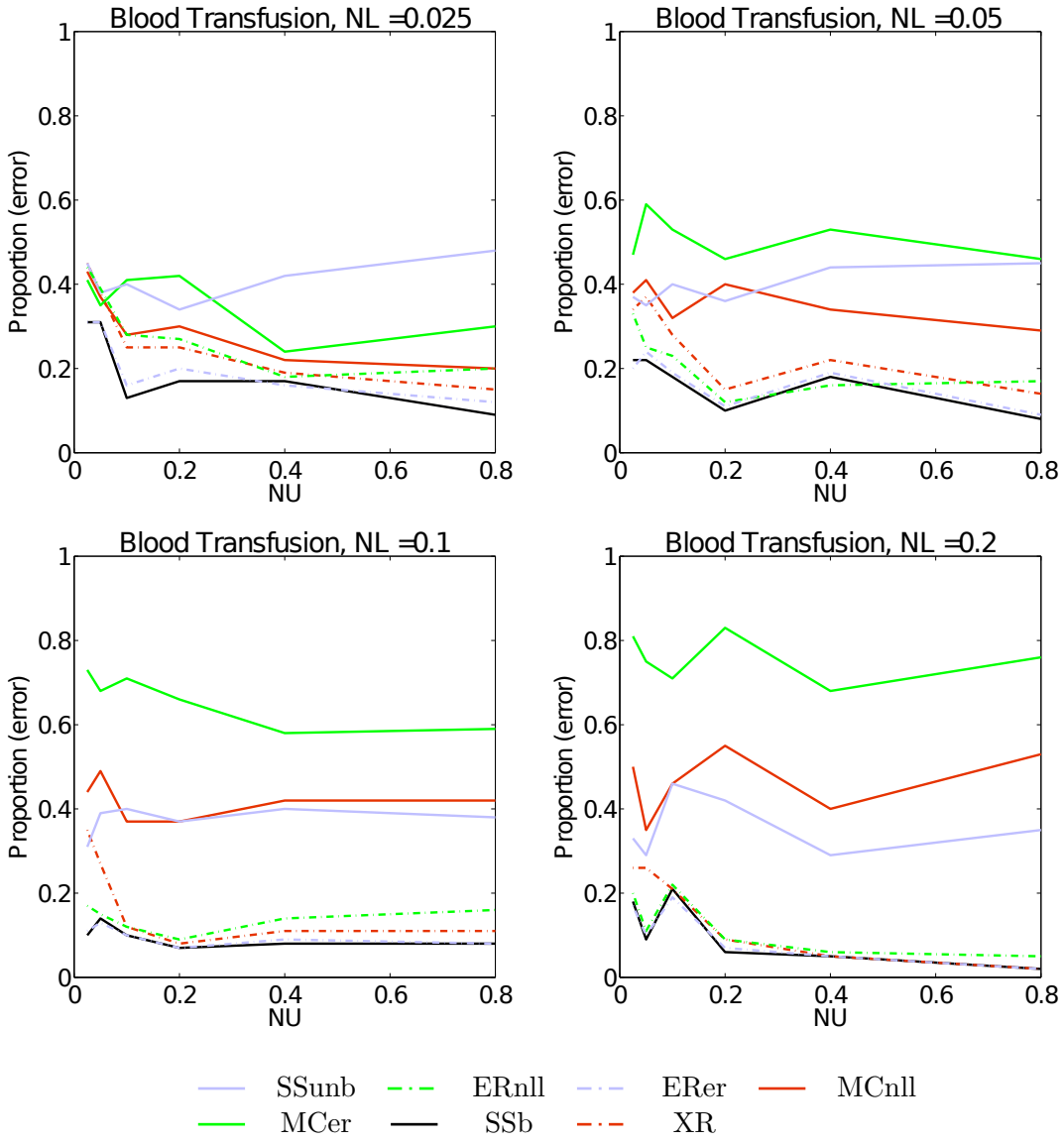


Figure 5: Four sample plots of the proportion of tests in which each semi-supervised learning scheme outperformed learning on the labelled data alone as measured by the error rate on the **Blood** data set for a variety of values of N_L, as N_U grows.

6. Conclusions

We have presented tighter bounds on the bias introduced when performing semi-supervised (as opposed to full supervised) maximum likelihood learning with a generative model using the standard technique. We also provide a new interpretation which gives an intuitive explanation for why the results are often poor with large amounts of unlabelled data.

Additionally, we have demonstrated a simple example of a new unbiased objective function which approximately minimises $KL\left(P(X,Y)||P(X,Y|\theta)\right)$. This method is no more computationally complex than simply augmenting the likelihood, demonstrates very good behaviour with even very large quantities of unlabelled data, and requires quite weak conditions on the correlation between the conditional and generative components of the likelihood to reduce the variance of our estimator.

Although not covered, much of the analysis presented here likely to be applicable to regression problems as well as classification ones. We leave this as an avenue for possible future work.

Acknowledgments

This work was supported by the EPSRC.

Appendix A. Supplementary Results Of Unbiased Semi-Supervised Training

There are two measures of performance we examine, evaluated over a holdout set:

- The error rate
- The negative log likelihood

For each of these measures two statistics are calculated:

- The mean performance over all repetitions, and the associated variance.
- The proportion of repetitions in which the performance was better than that achieved using the labelled data alone.

The former gives an approximate measure of 'average risk', according to whether we consider risk in terms of misclassification rate (for example when designing a classification algorithm) or negative log likelihood (such as when building a compression algorithm, say). The latter tells us, for each measure of 'risk', whether or not including unlabelled data has improved or worsened our performance.

Multi conditional learning, when cross validated according to error rate, often gave extremely bad negative log likelihood results. This caused unfavourable scaling of the axis, making other results indistinguishable. As such, the mean negative log likelihood results of **MCer** have been separated out and plotted alone.

See the main body of the text for such as abbreviations and data sets.

A.1 Mean Errors And Negative Log Likelihood

This section shows the mean error and negative log likelihood results.

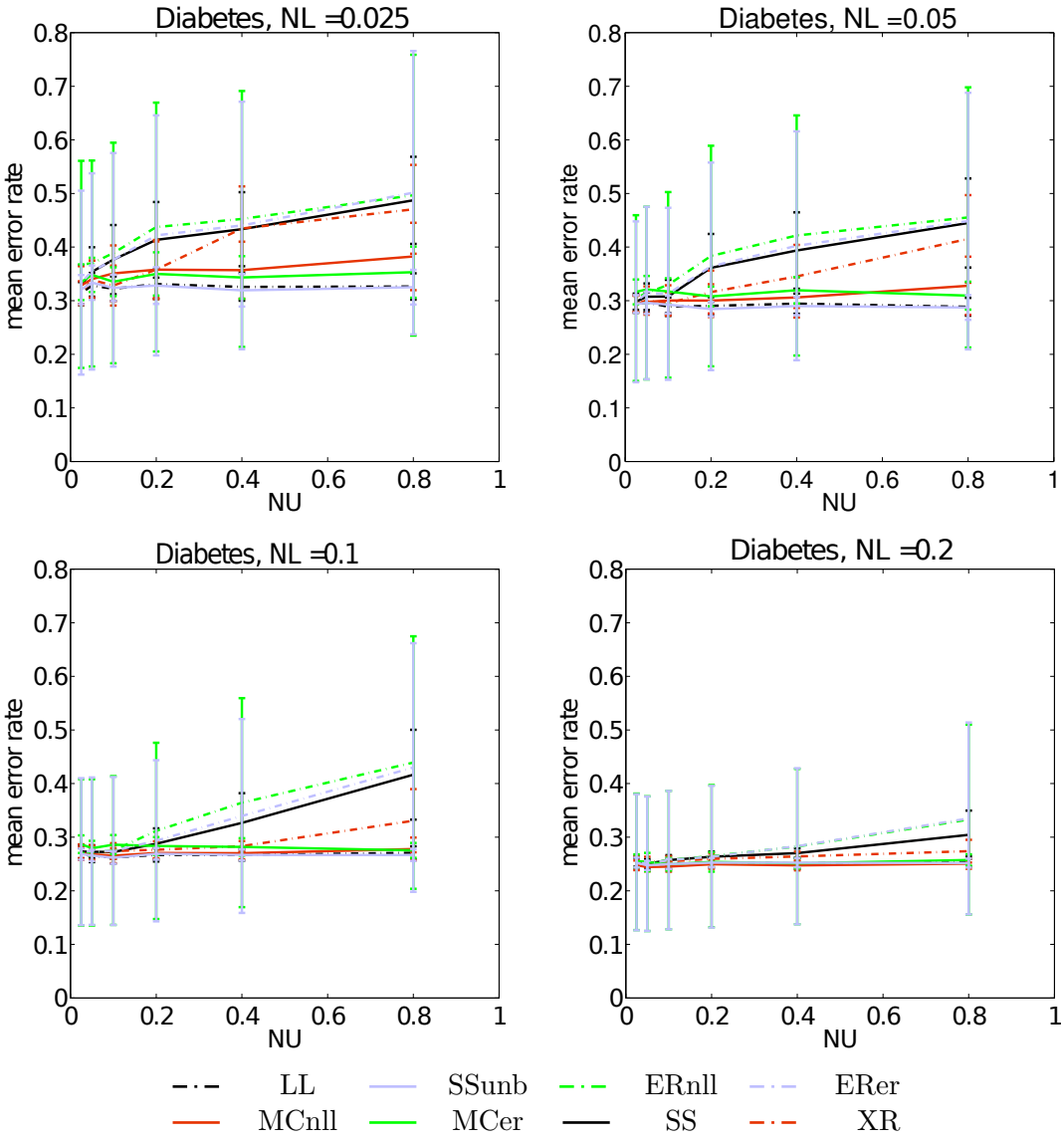


Figure 6: Mean error results of Diabetes data set

A.2 Proportion In Which Performance Improves

This section shows the proportion of test in which the error / negative log likelihood was improved by the addition of unlabelled data. That is, for every data set, the performance of the model was evaluated for each training scheme, and compared to the performance when only the labelled data was used. The frequency with which each semi-supervised scheme outperforms supervised learning was recorded and normalised. This gives an estimate of the probability that including unlabelled data will improve performance compared to supervised learning alone. A value close to 1 indicates reliable improvement when unlabelled data is added, whereas one close to 0 shows reliable worsening of results.

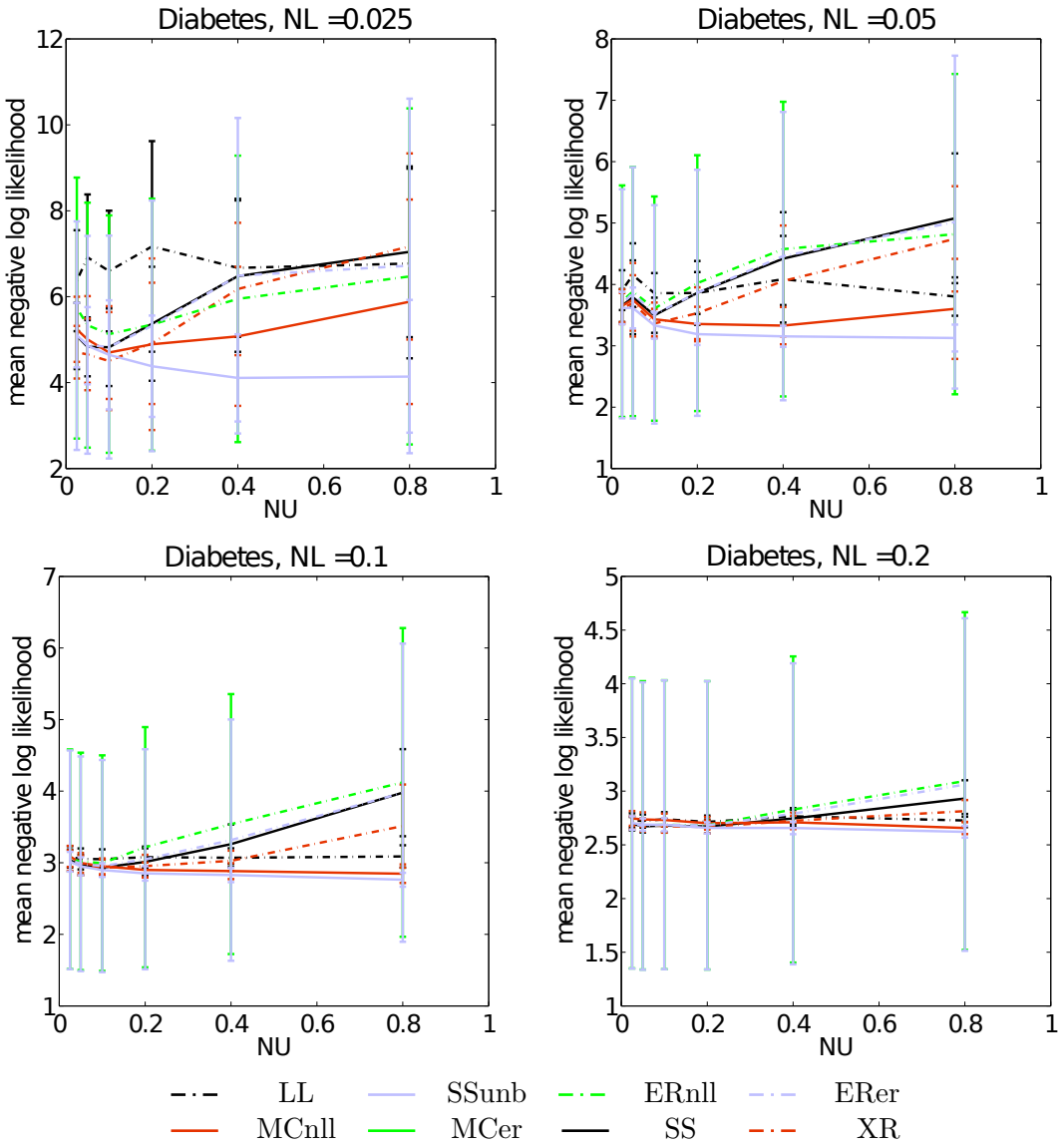


Figure 7: Mean log likelihood results of Diabetes data set

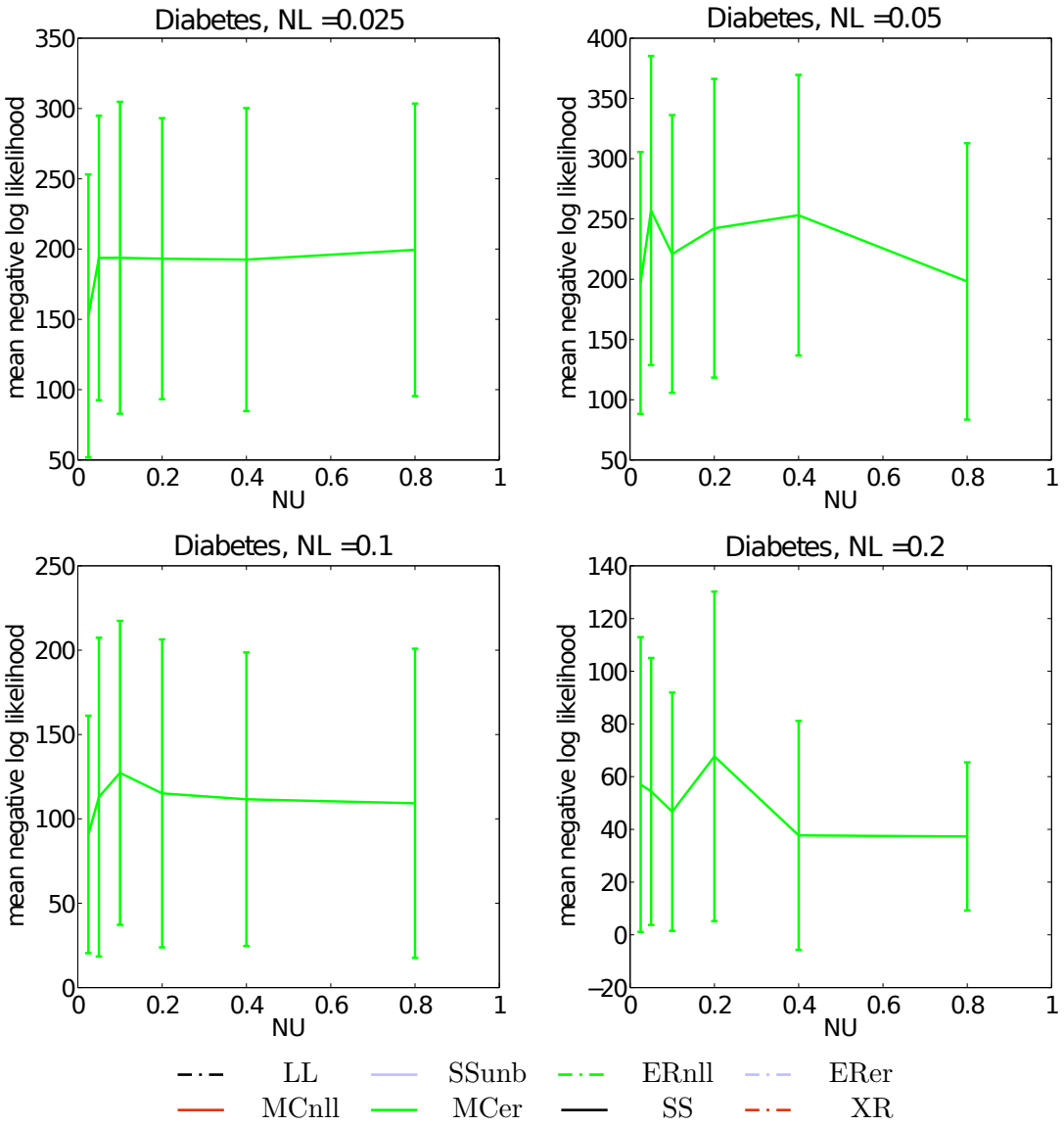


Figure 8: Mean log likelihood results of Diabetes data set

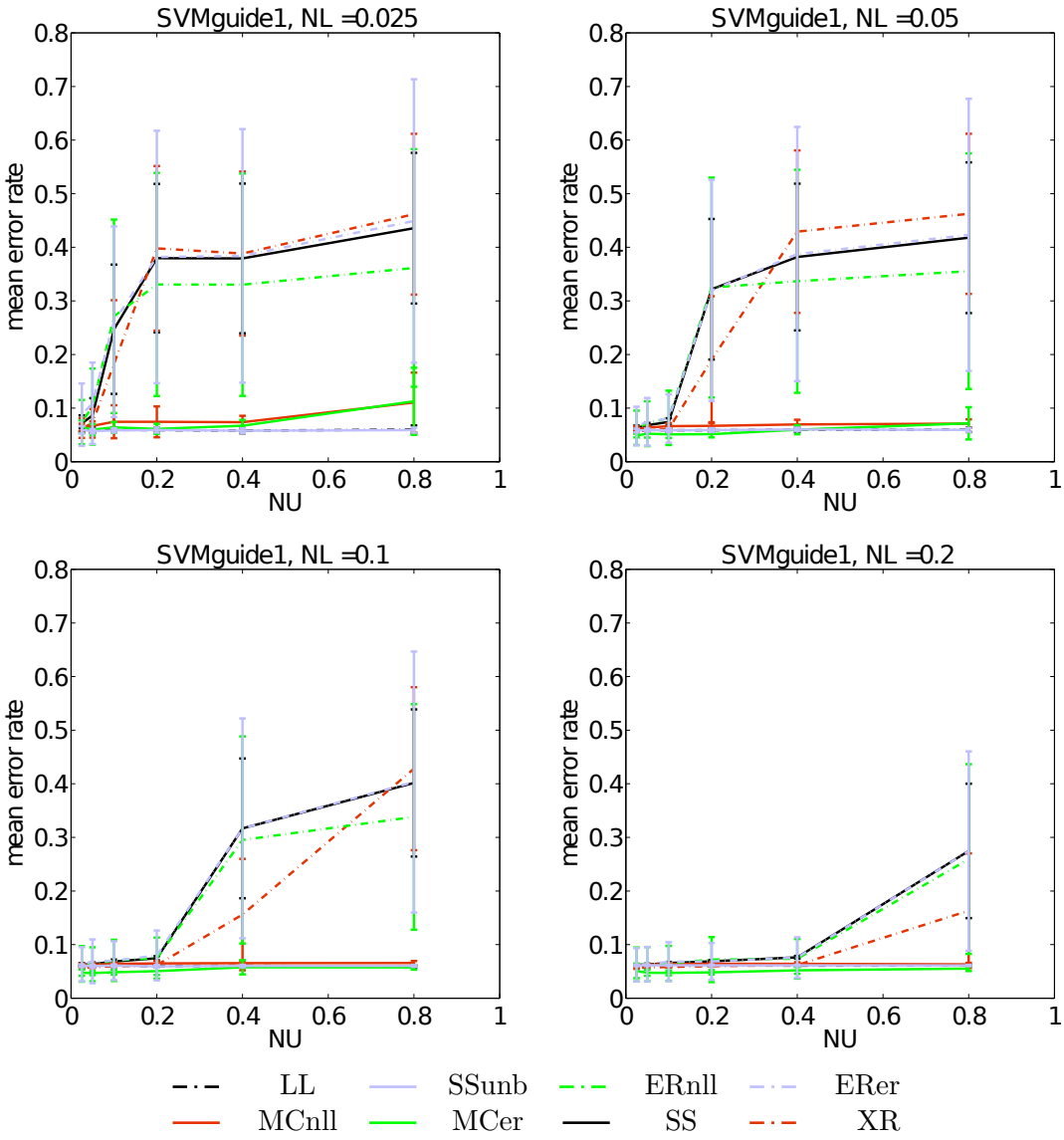


Figure 9: Mean error results of the SVMguide data set

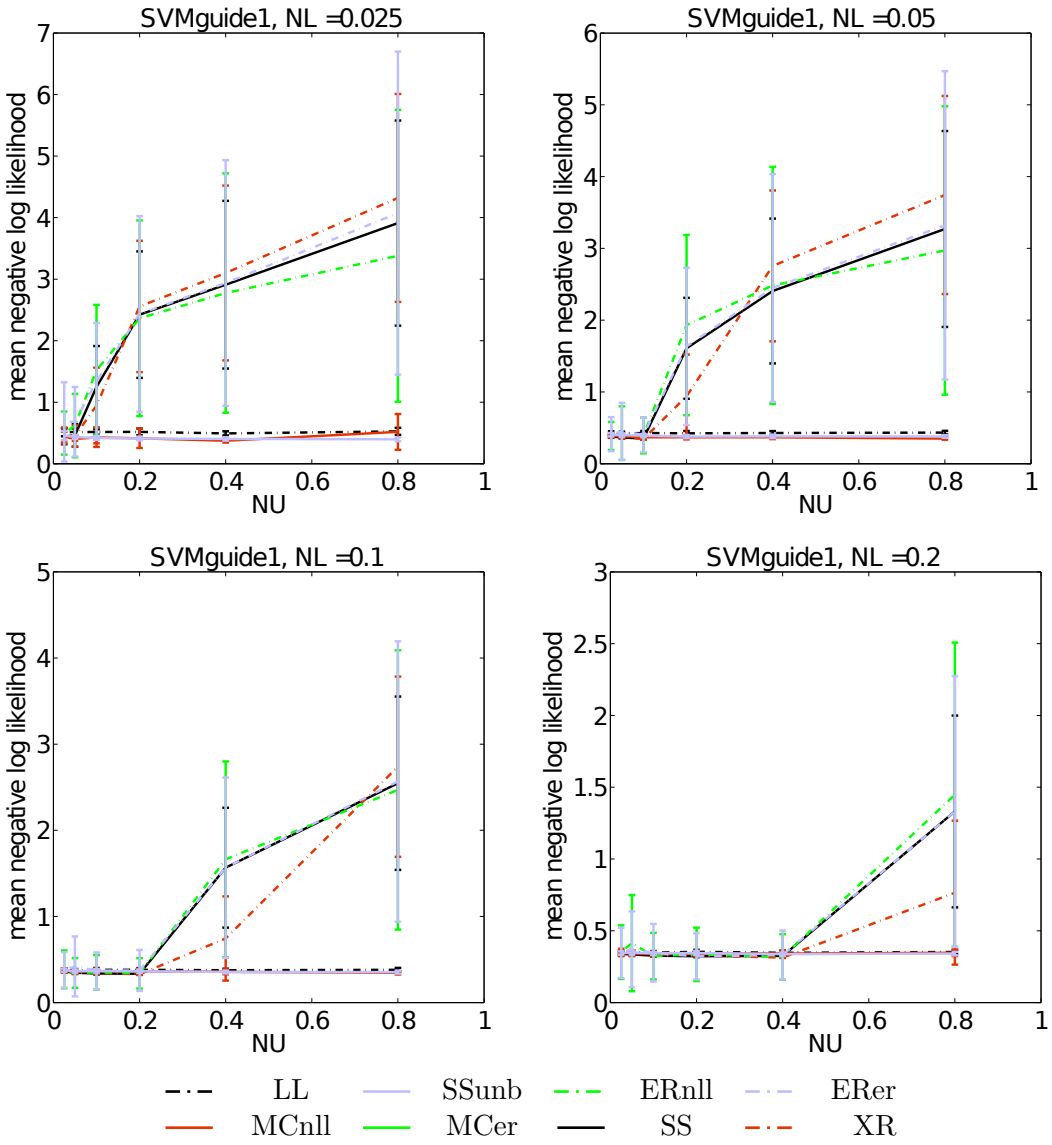


Figure 10: Mean log likelihood results of the SVMguide data set

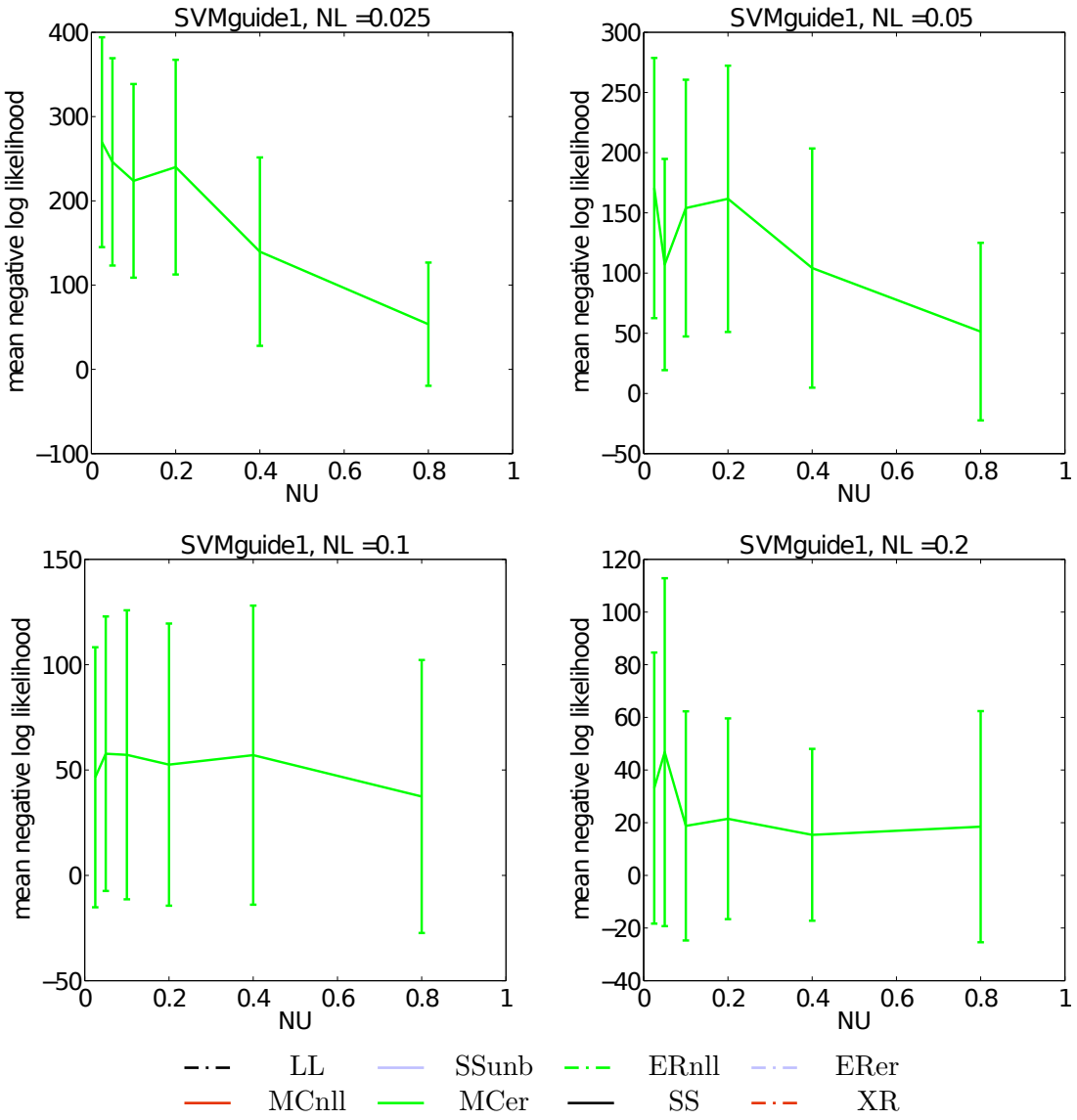


Figure 11: Mean log likelihood results of the SVMguide data set

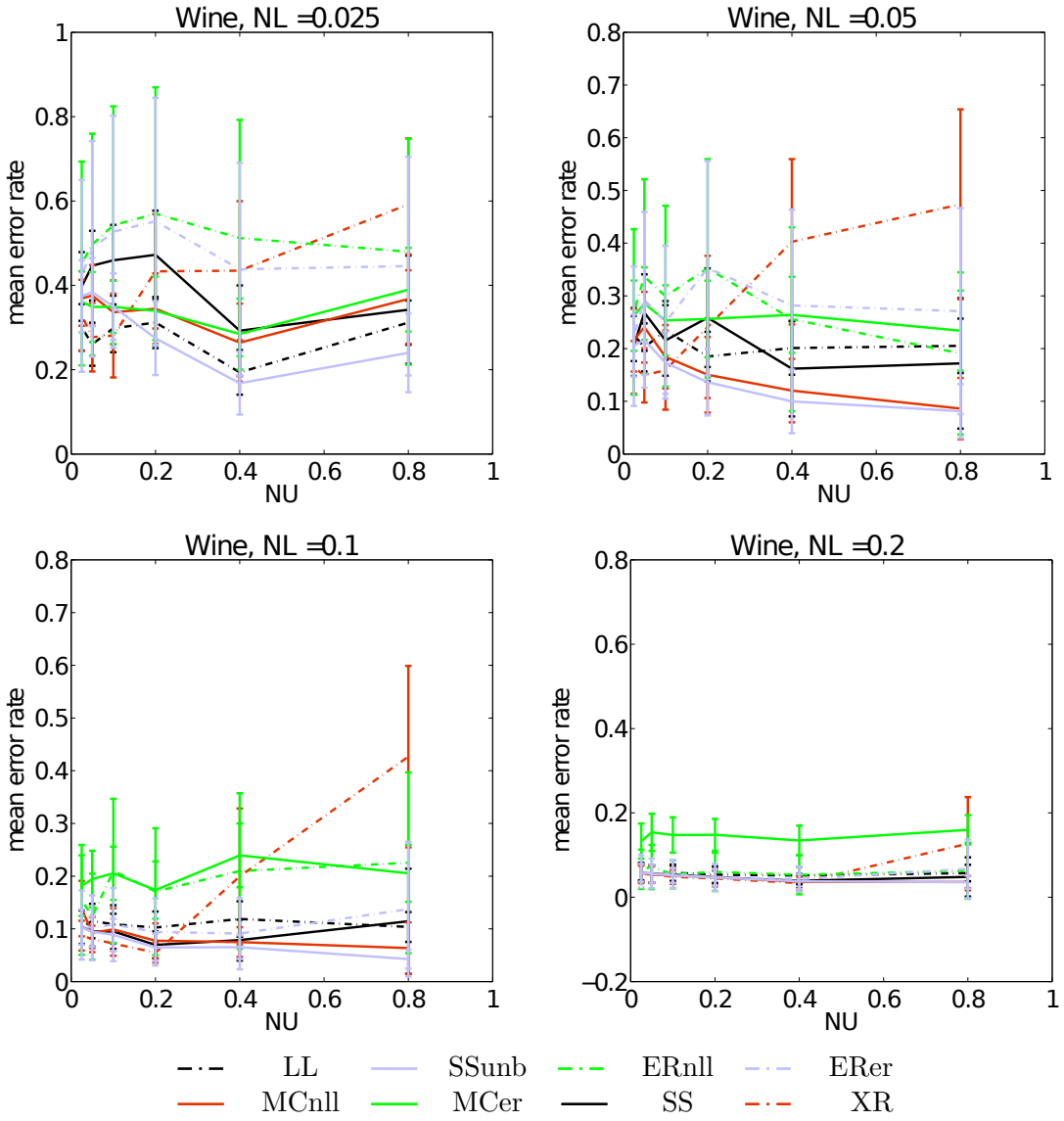


Figure 12: Mean error results of the Wine data set

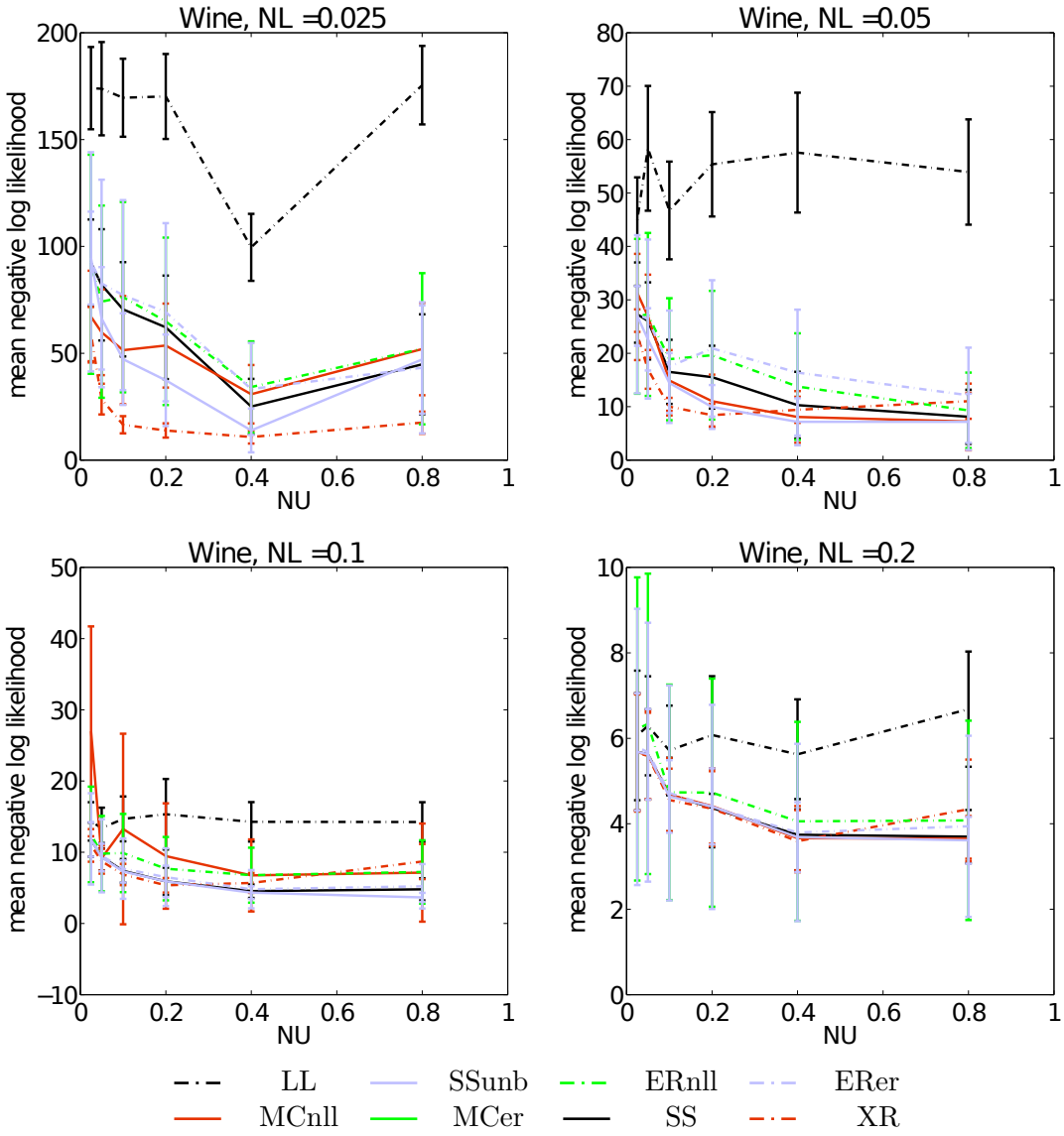


Figure 13: Mean log likelihood results of the Wine data set

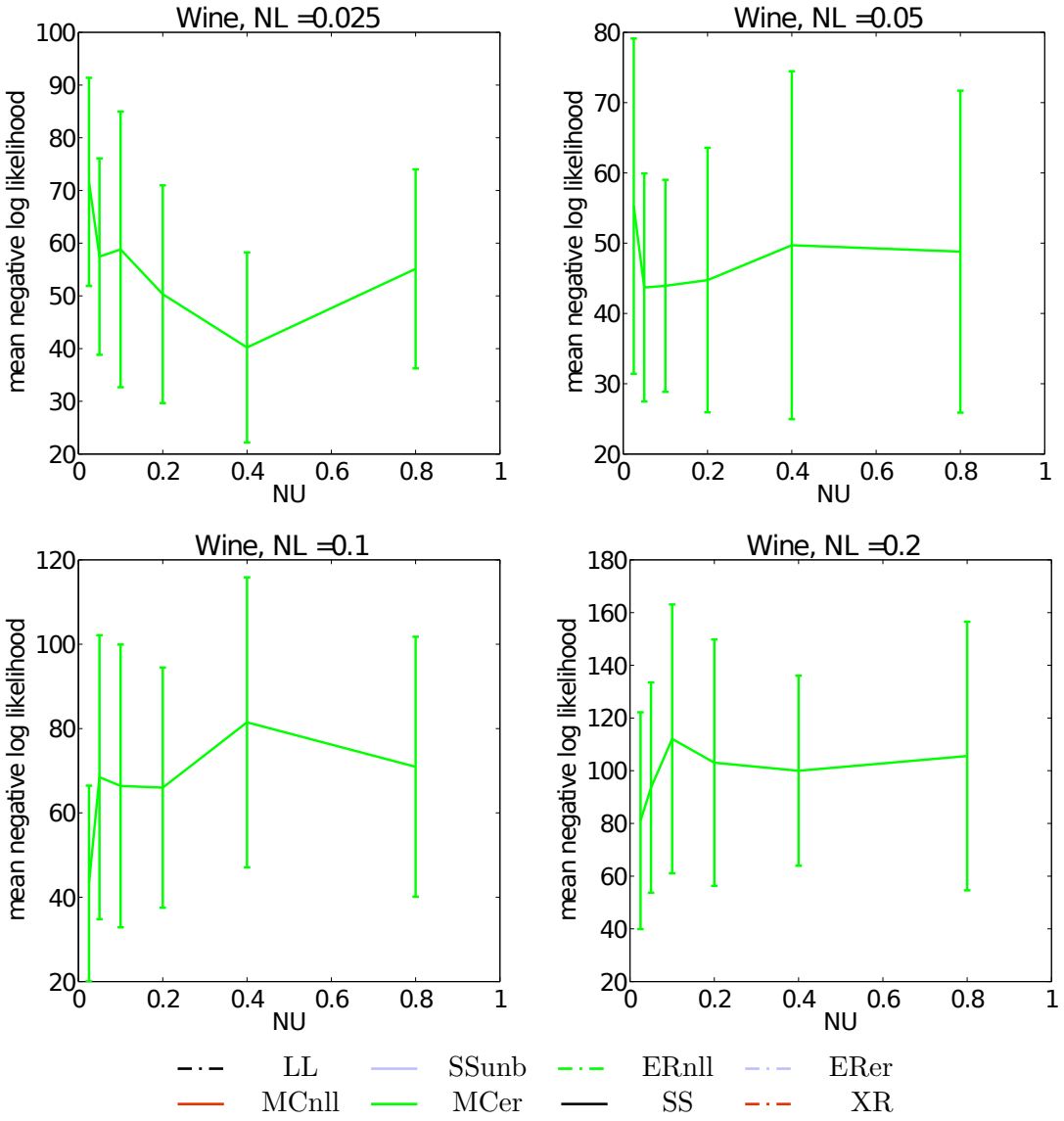


Figure 14: Mean log likelihood results of the Wine data set

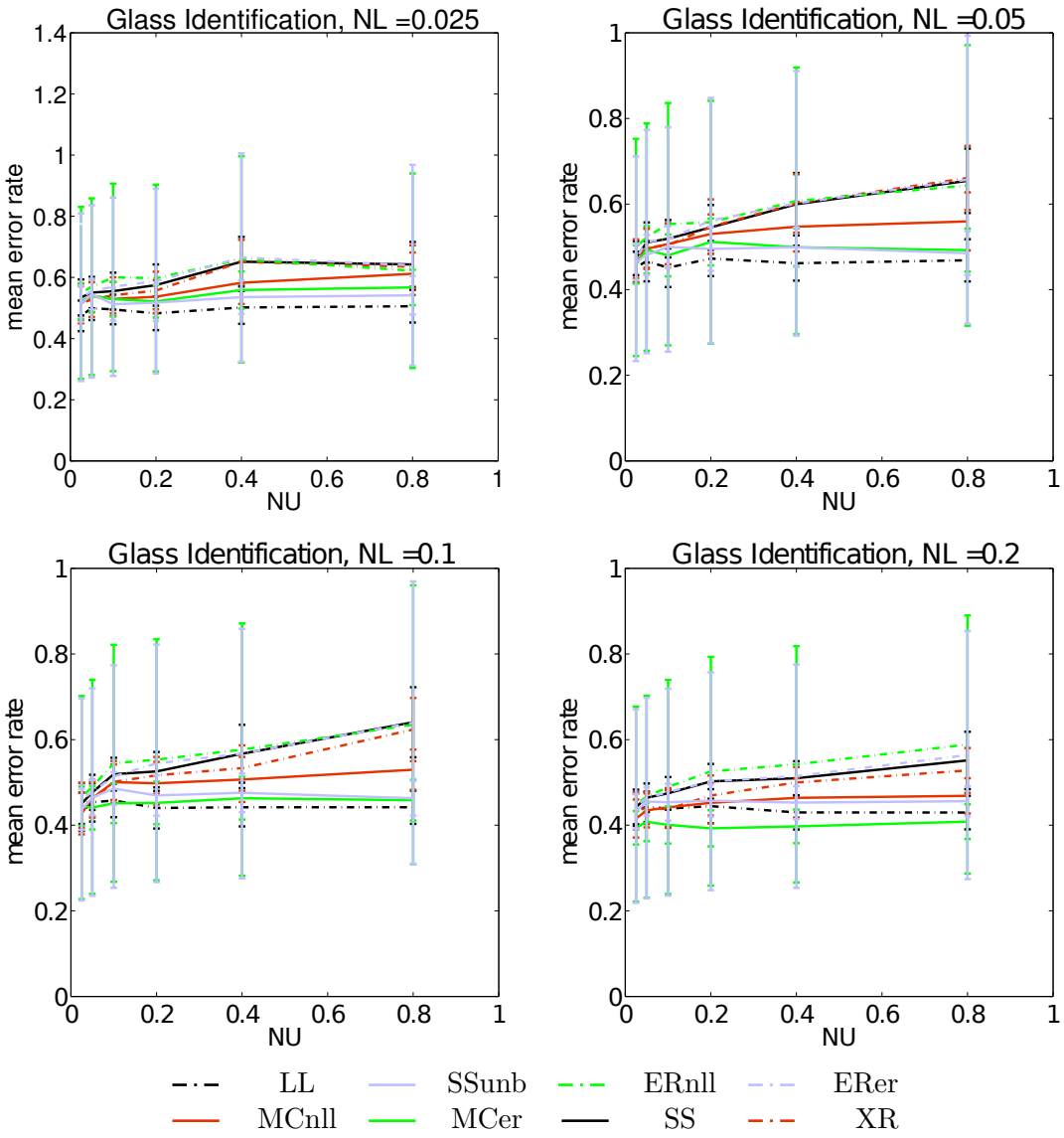


Figure 15: Mean error results of the Glass data set

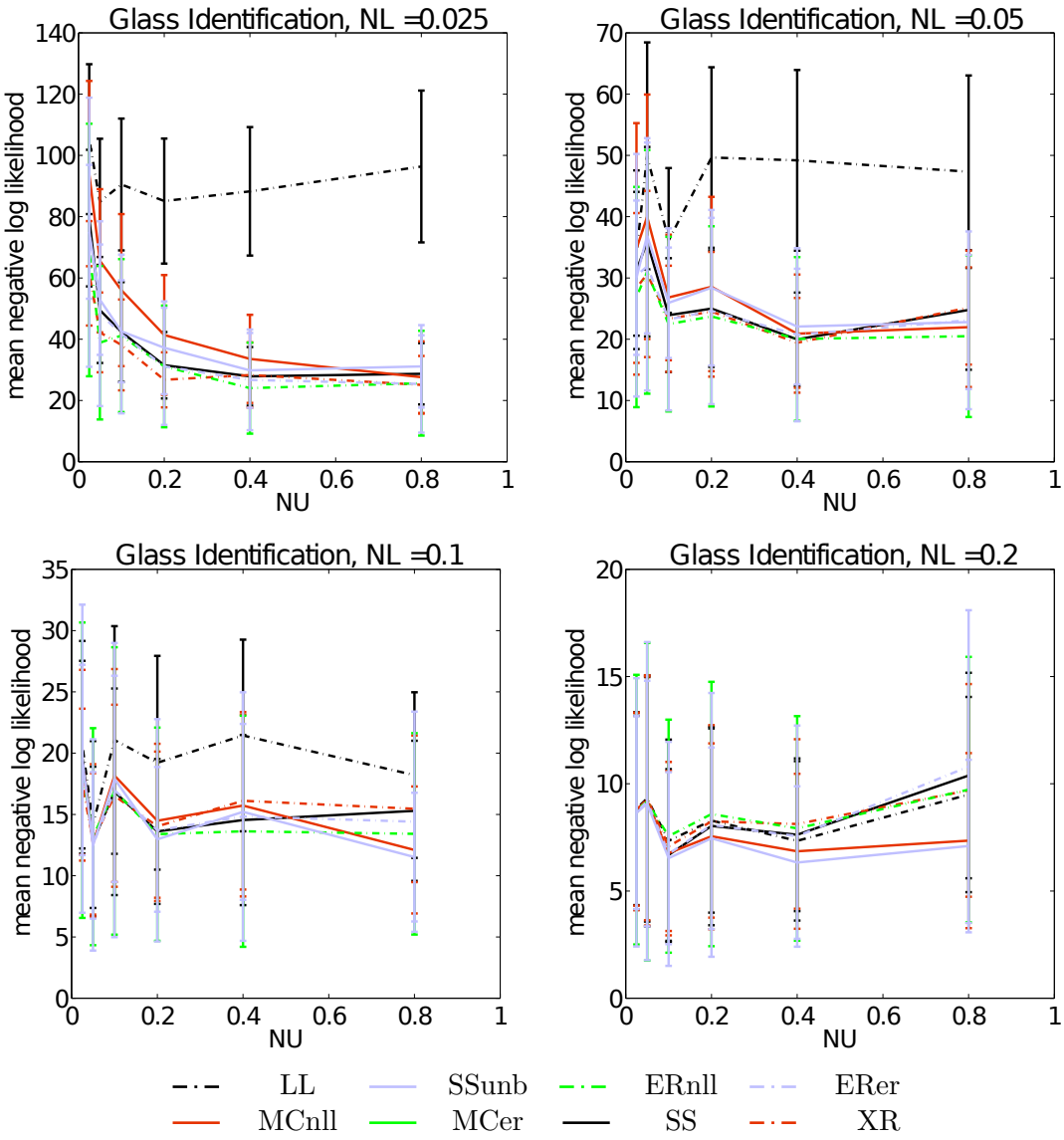


Figure 16: Mean log likelihood results of the Glass data set

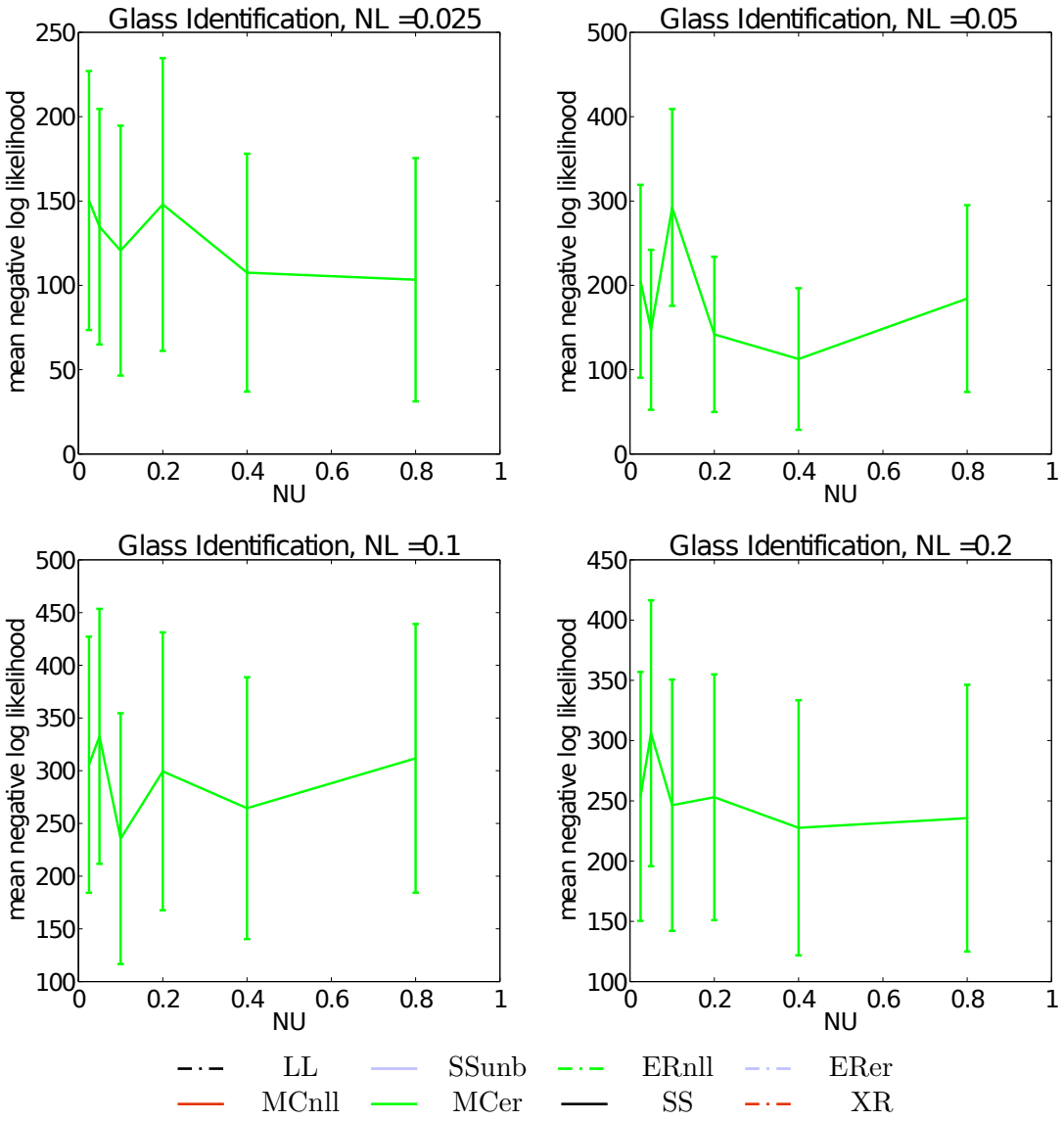


Figure 17: Mean log likelihood results of the Glass data set

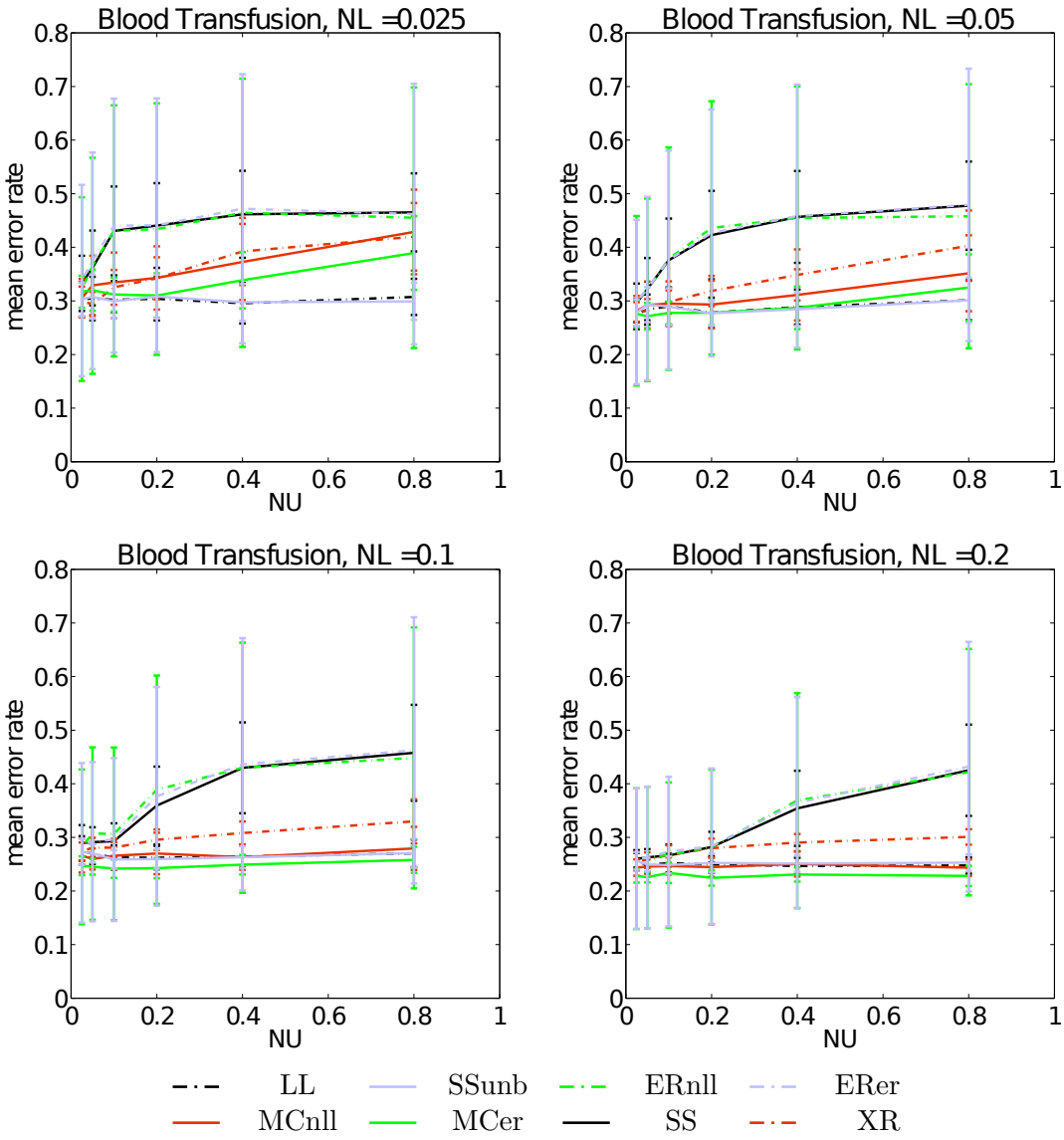


Figure 18: Mean error results of the Blood data set

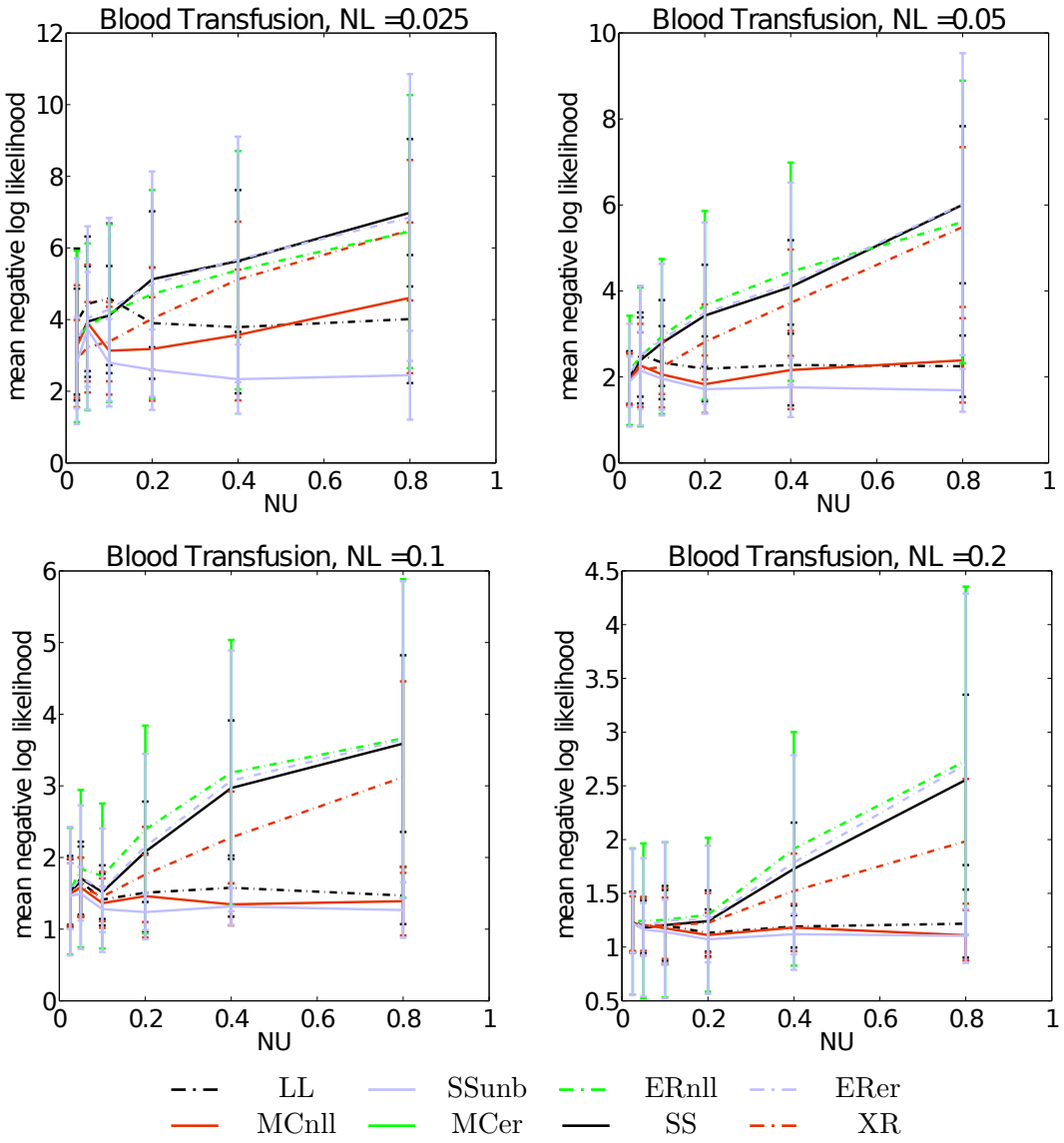


Figure 19: Mean log likelihood results of the Blood data set

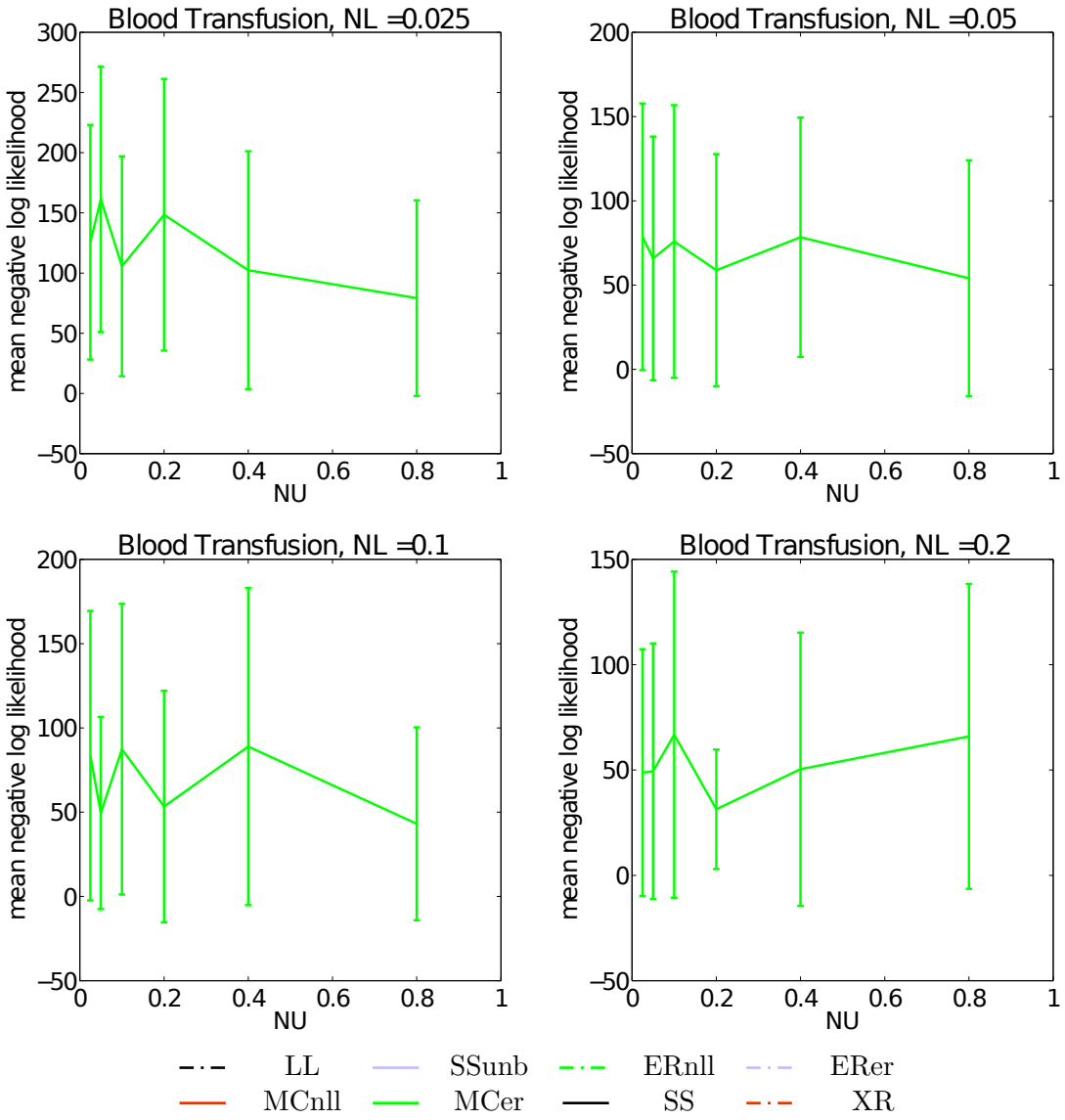


Figure 20: Mean log likelihood results of the Blood data set

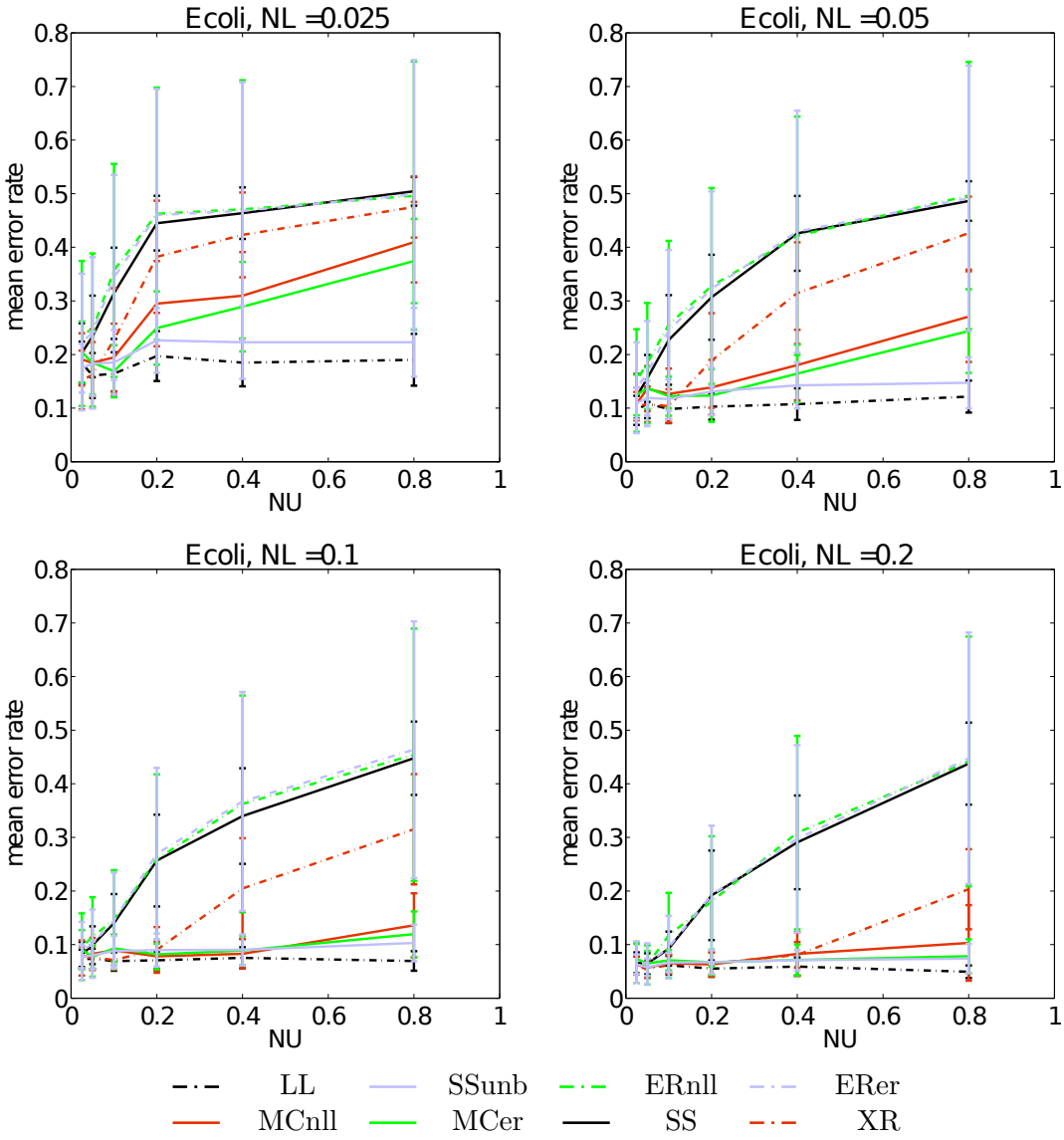


Figure 21: Mean error results of the Ecoil data set

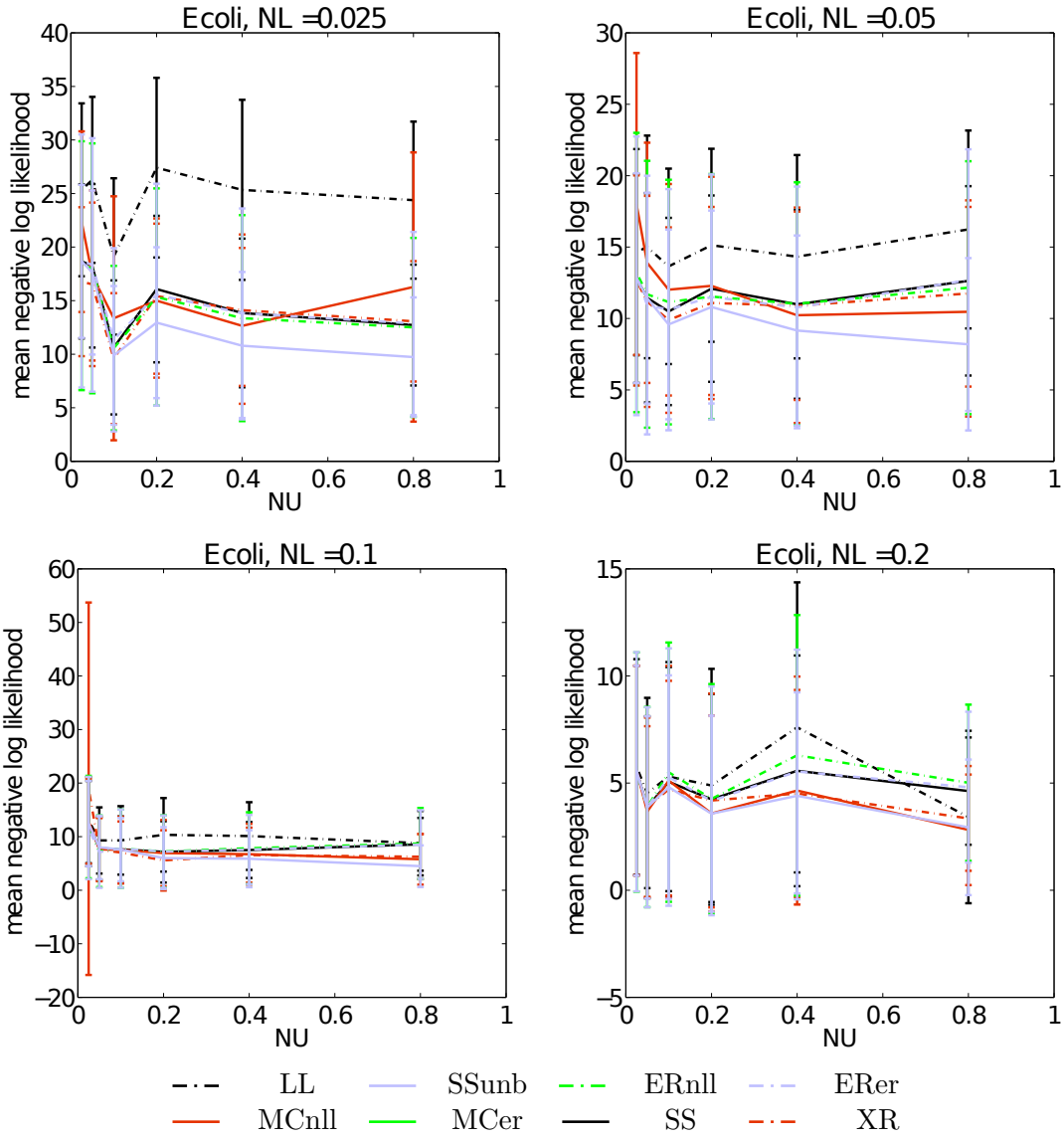


Figure 22: Mean log likelihood results of the Ecoil data set

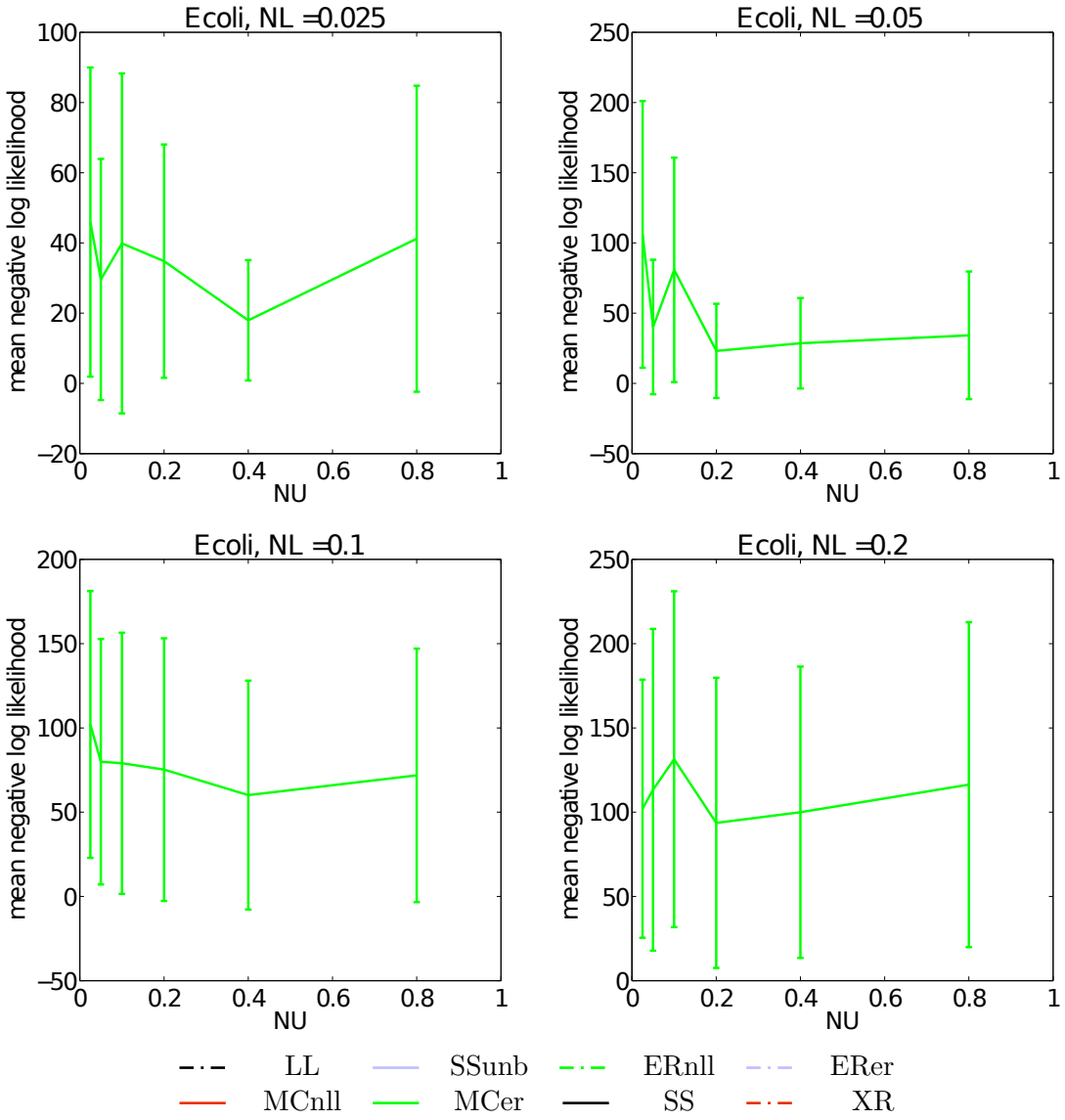


Figure 23: Mean log likelihood results of the Ecoli data set

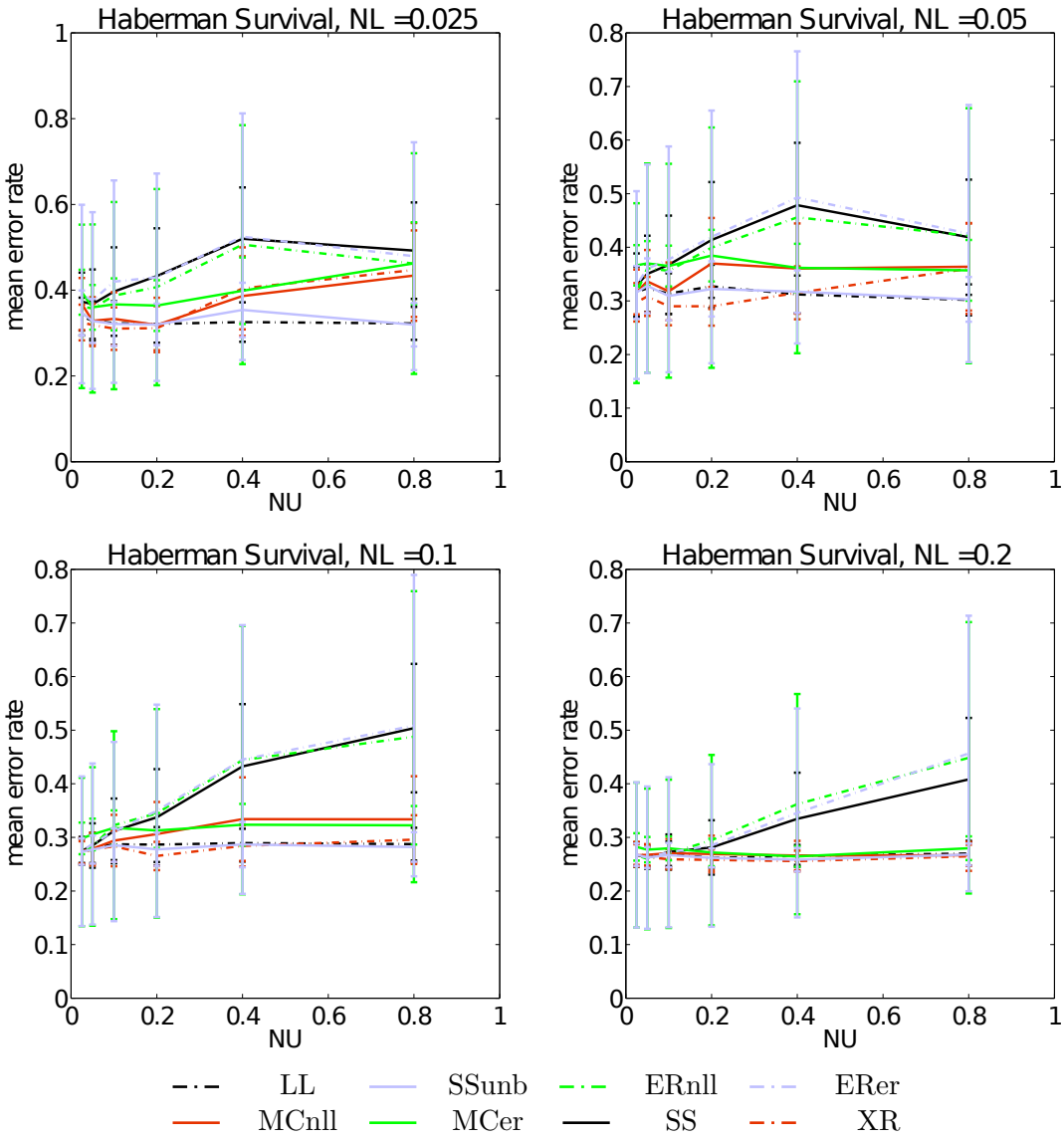


Figure 24: Mean error results of the Haberman data set

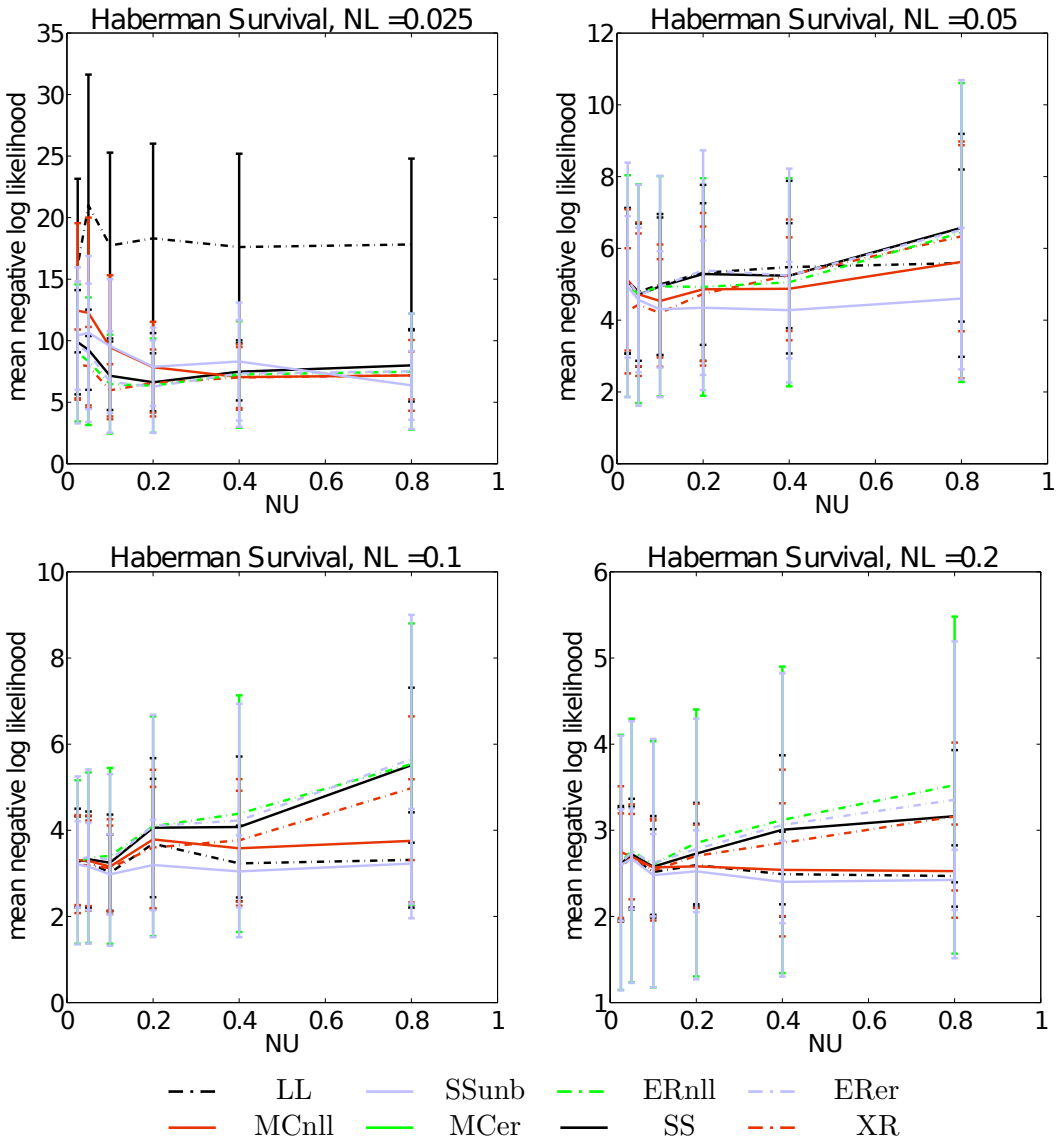


Figure 25: Mean log likelihood results of the Haberman data set

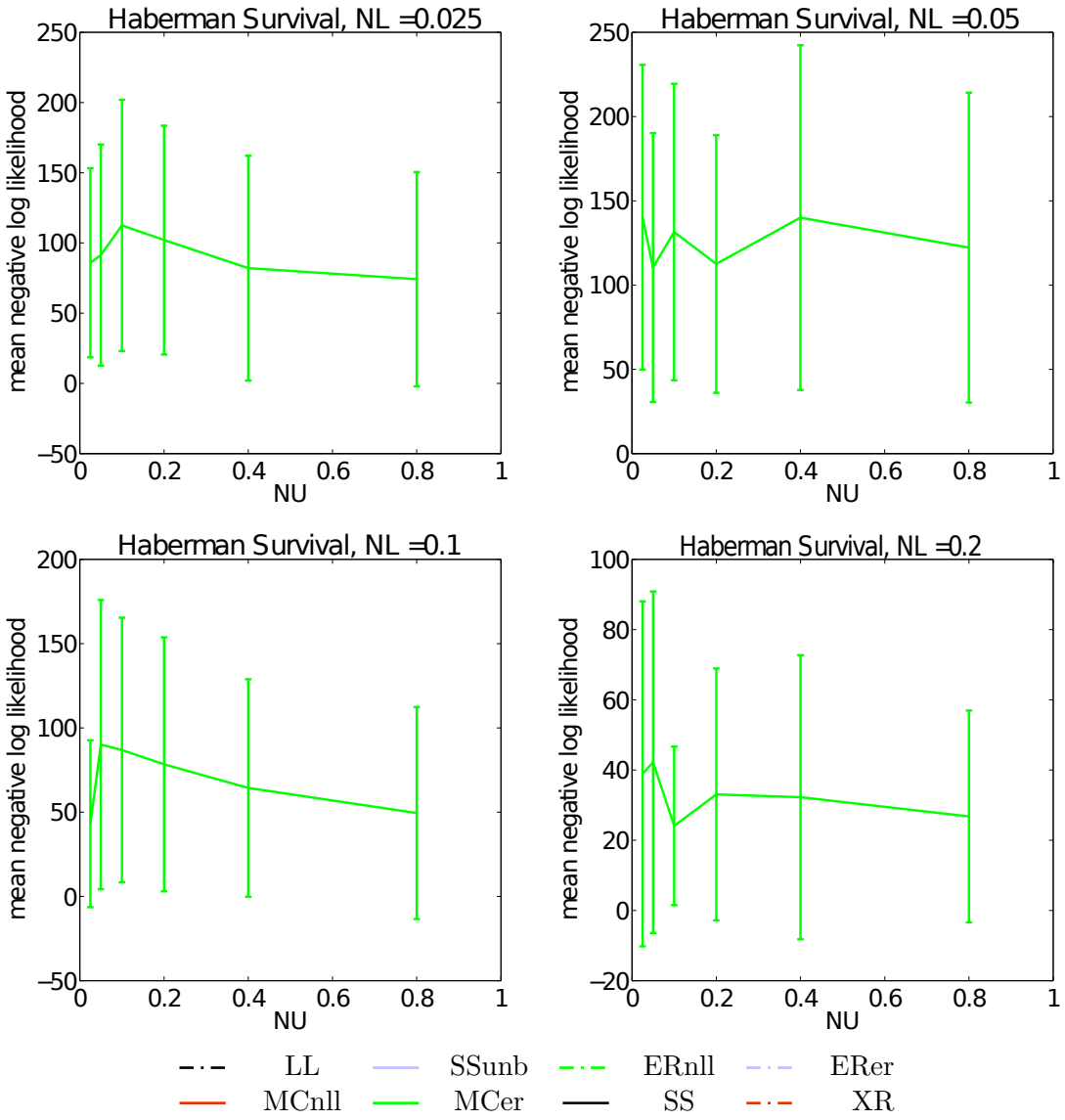


Figure 26: Mean log likelihood results of the Haberman data set

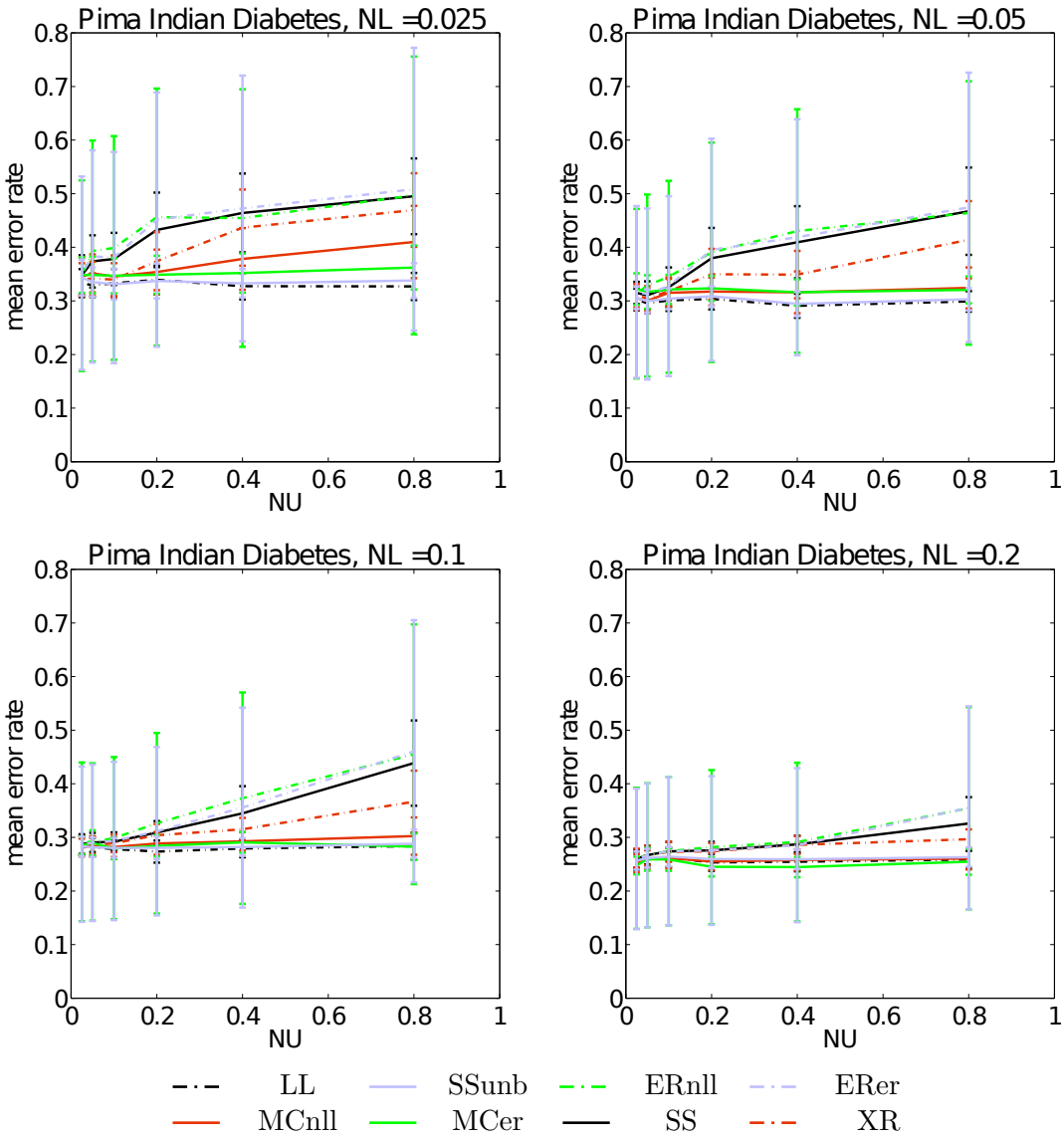


Figure 27: Mean error results of the Pima Indian data set

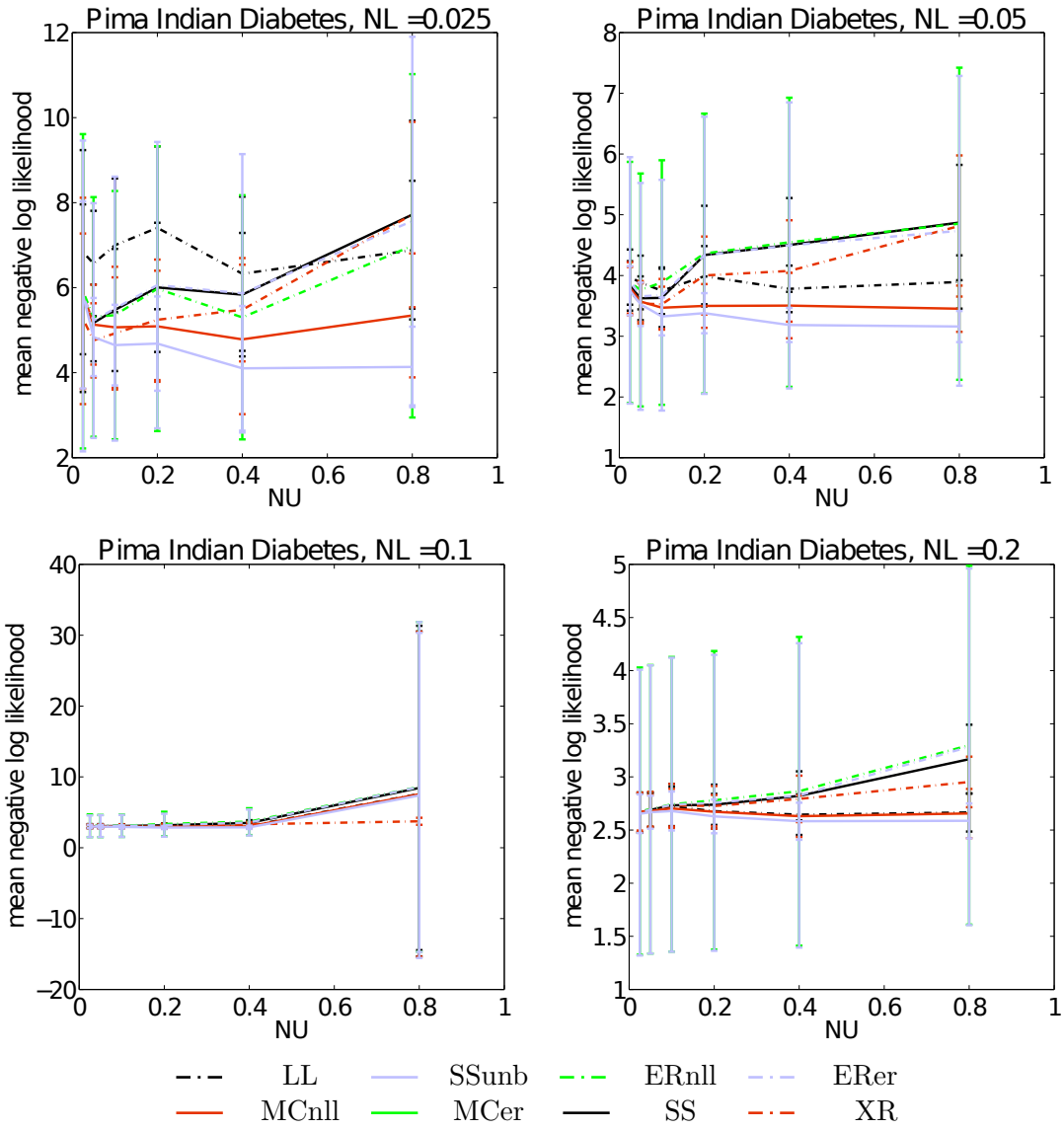


Figure 28: Mean log likelihood results of the Pima Indian data set

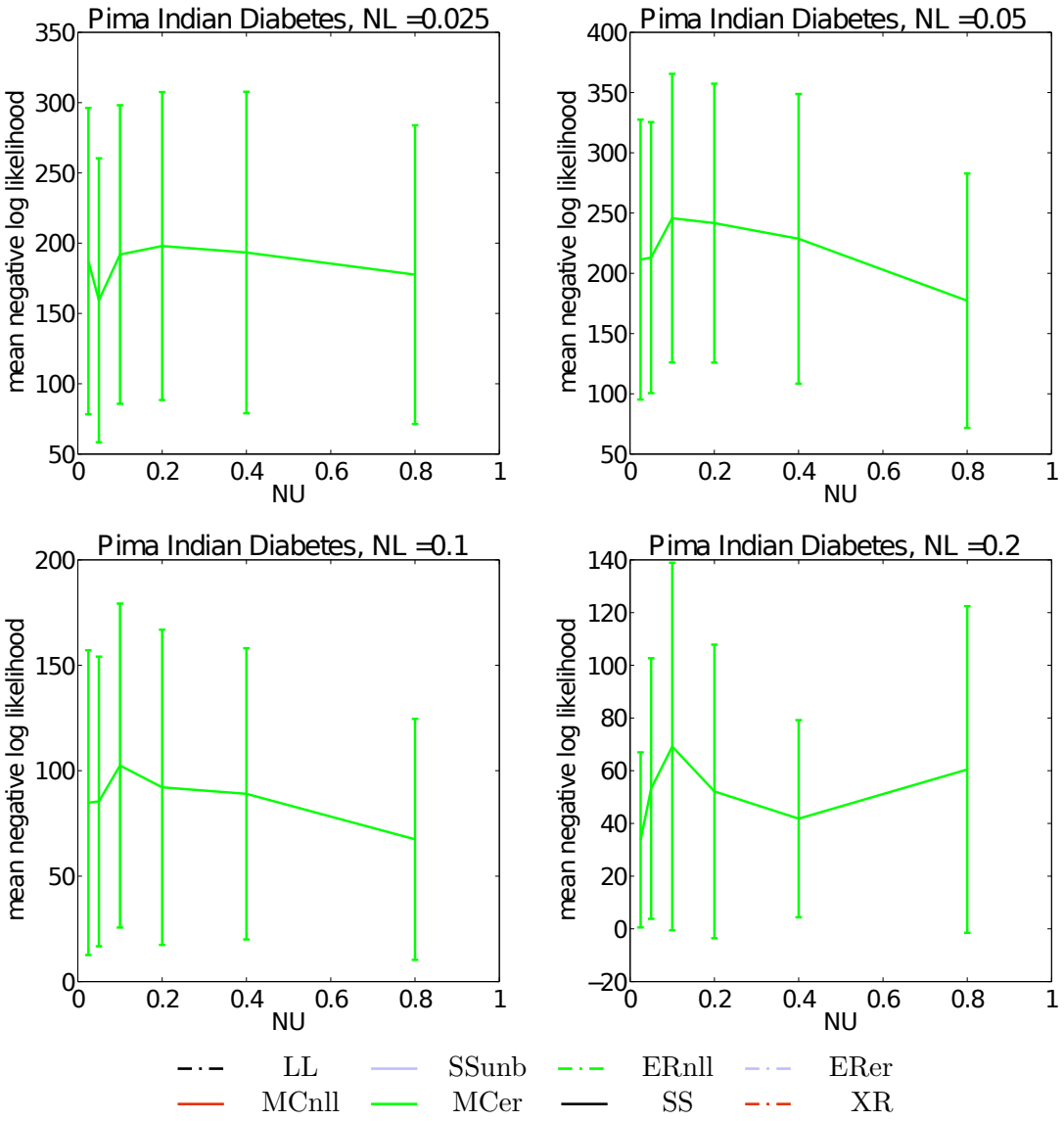


Figure 29: Mean log likelihood results of the Pima Indian data set

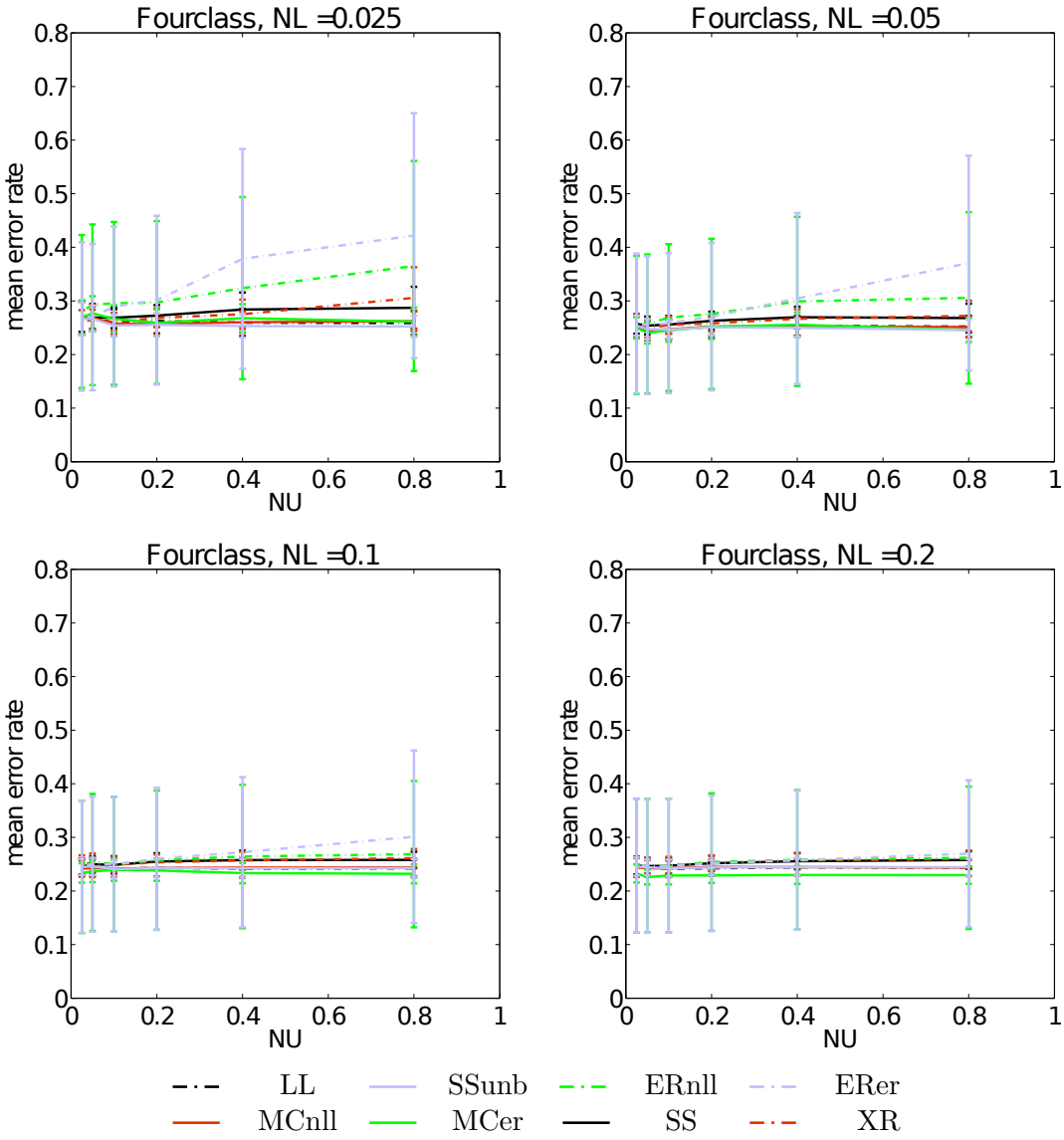


Figure 30: Mean error results of the Fourclass data set

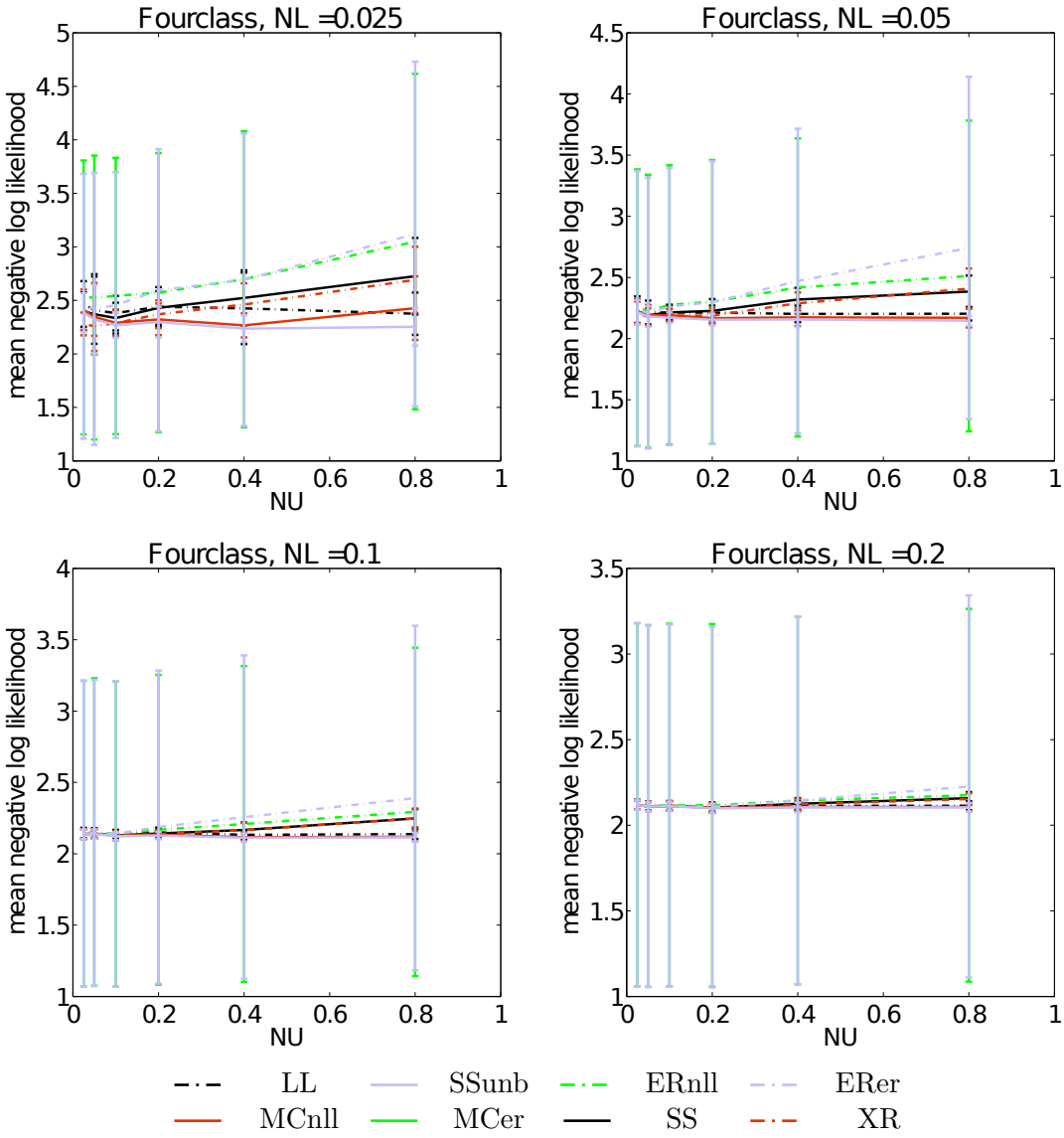


Figure 31: Mean log likelihood results of the Fourclass data set

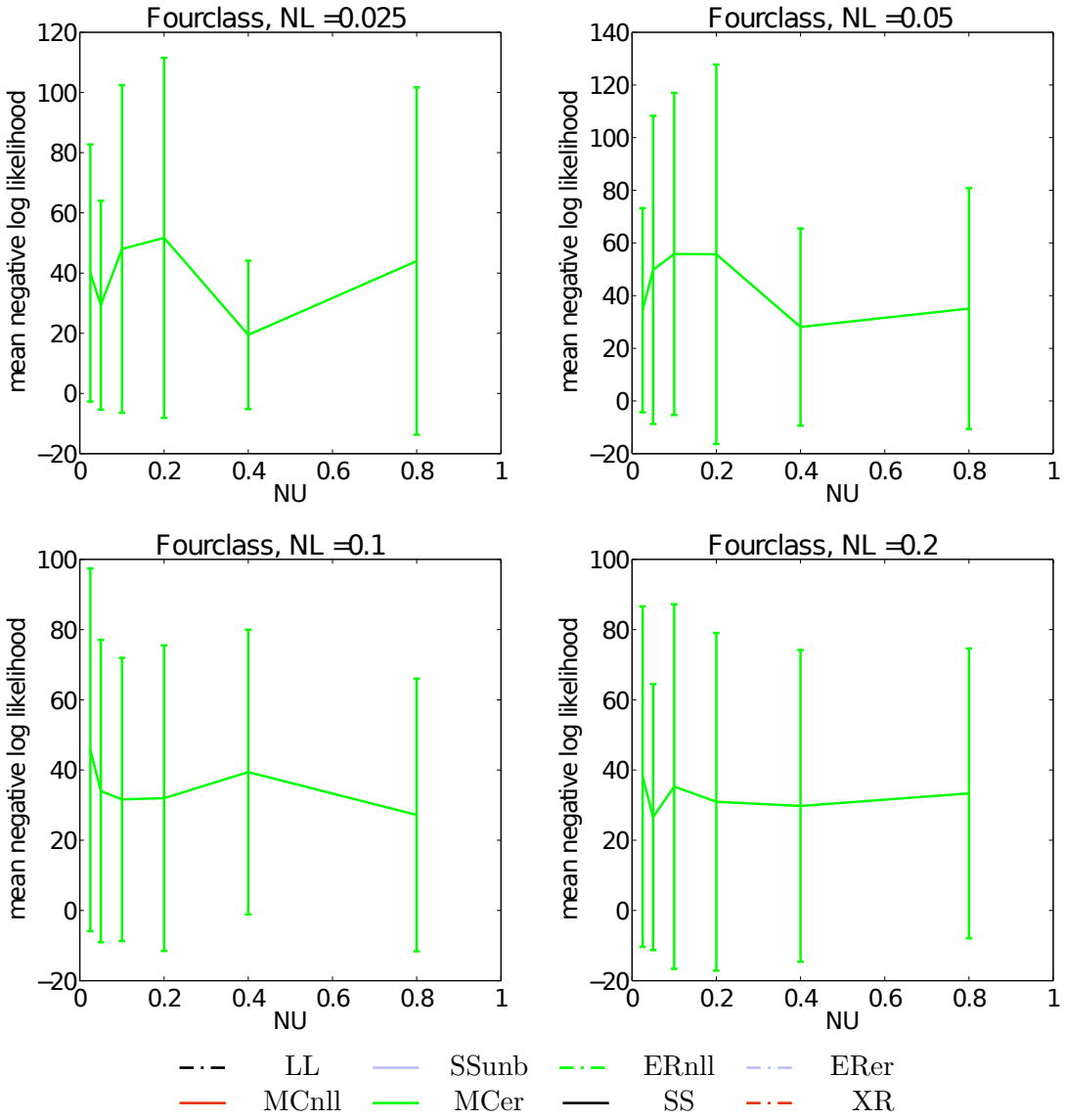


Figure 32: Mean log likelihood results of the Fourclass data set

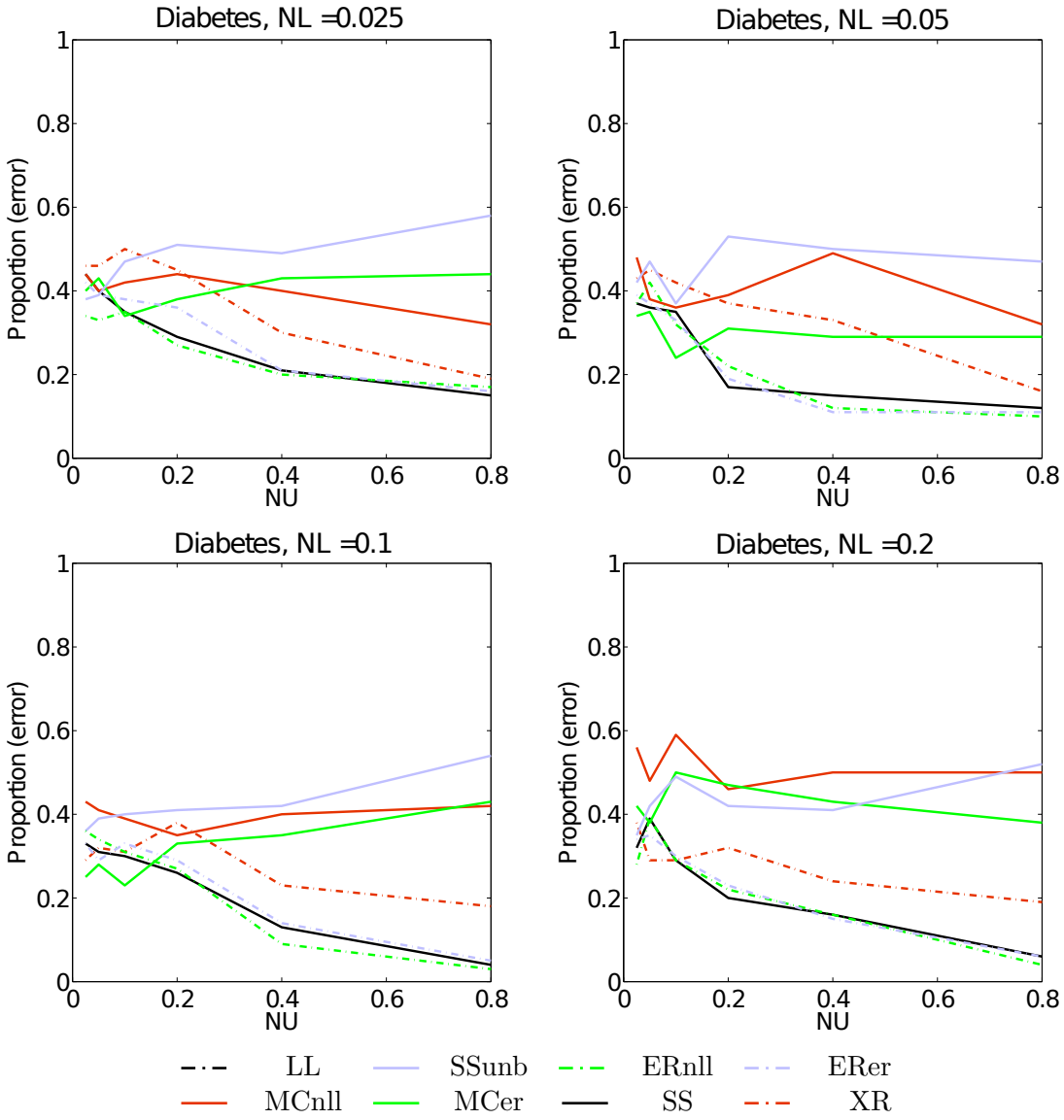


Figure 33: Chance of error rate improvement VS labelled data alone - Diabetes data set

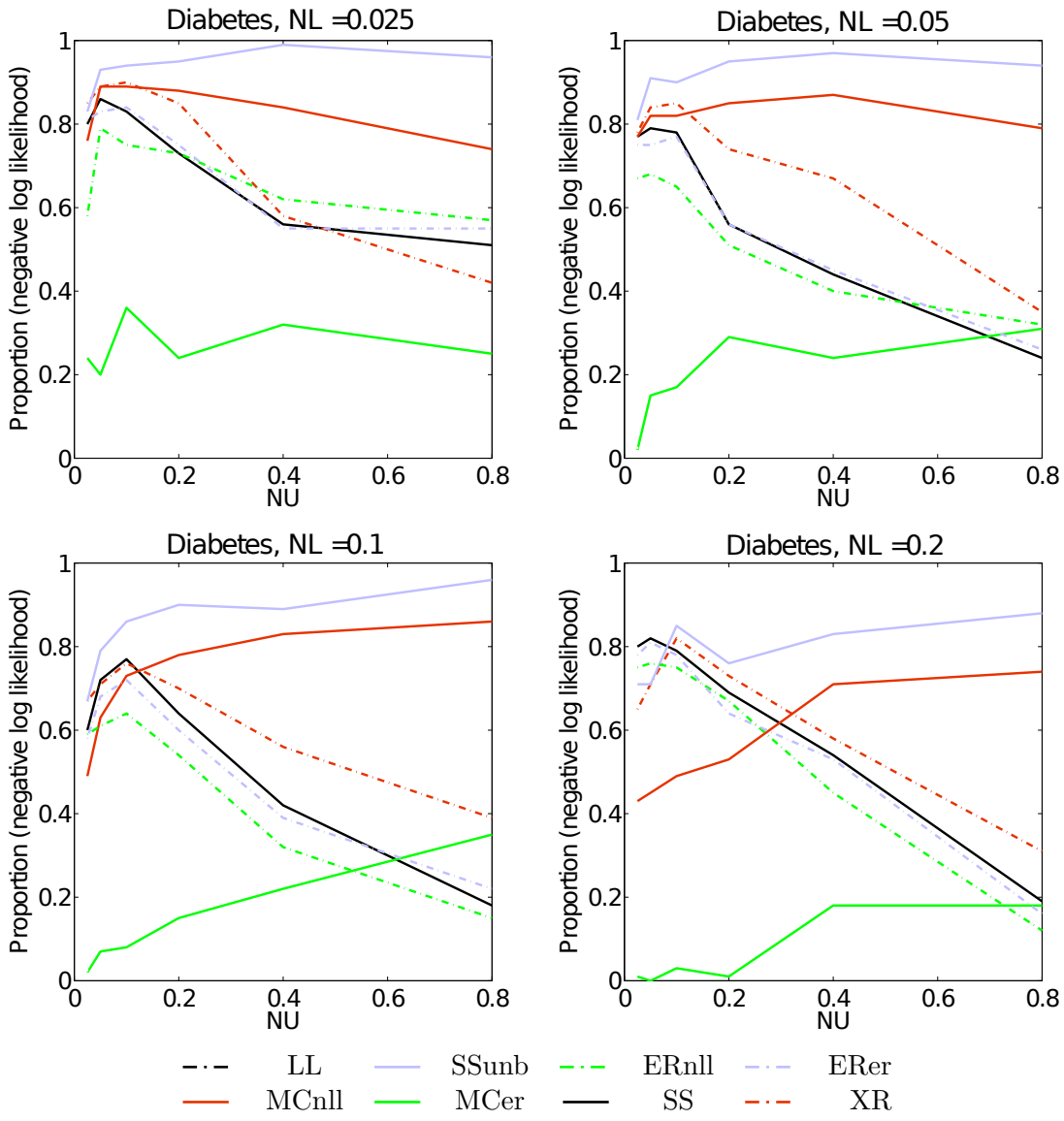


Figure 34: Chance of log likelihood improvement VS labelled data alone - Diabetes data set

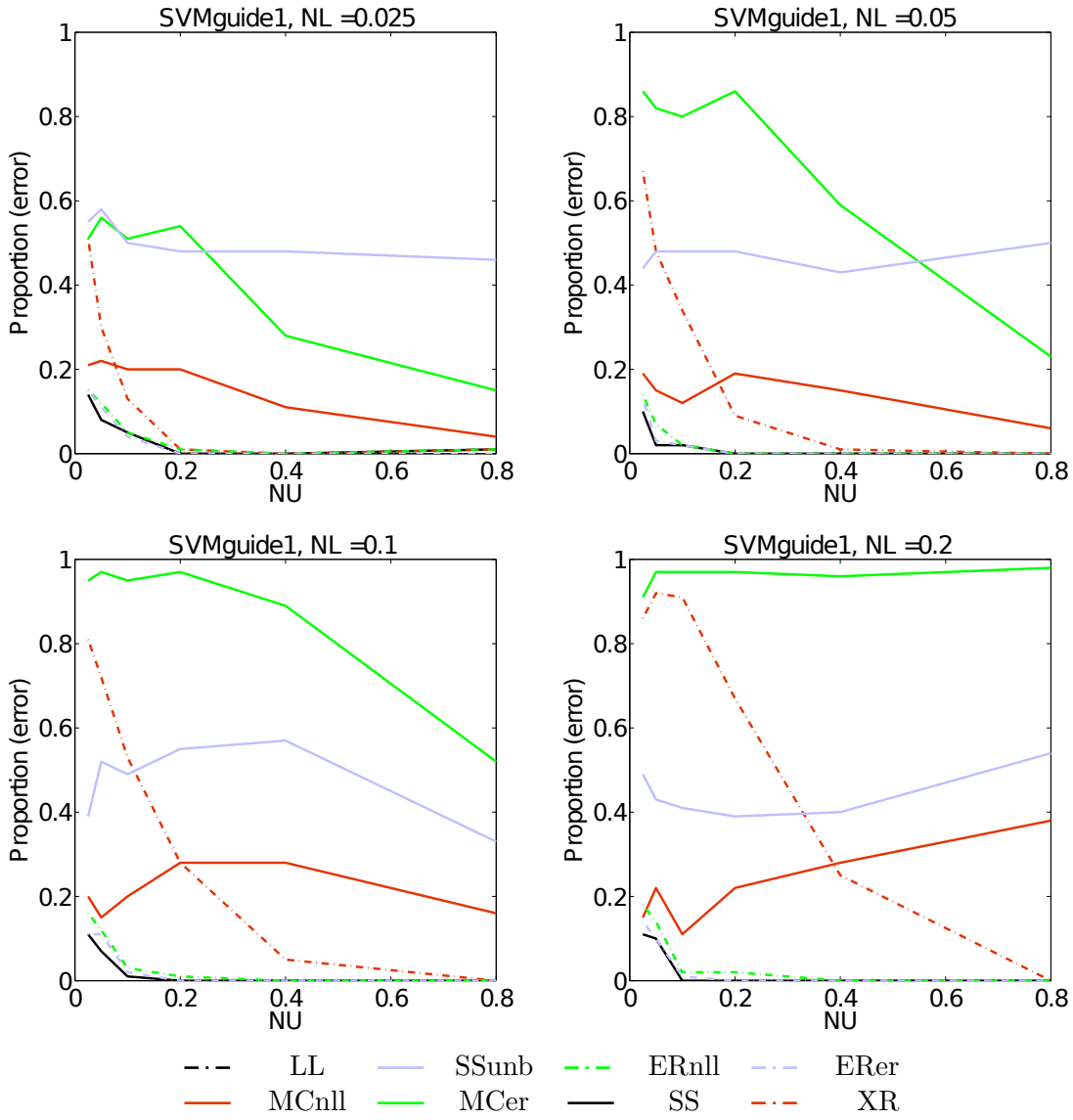


Figure 35: Chance of error rate improvement VS labelled data alone - SVMguide data set

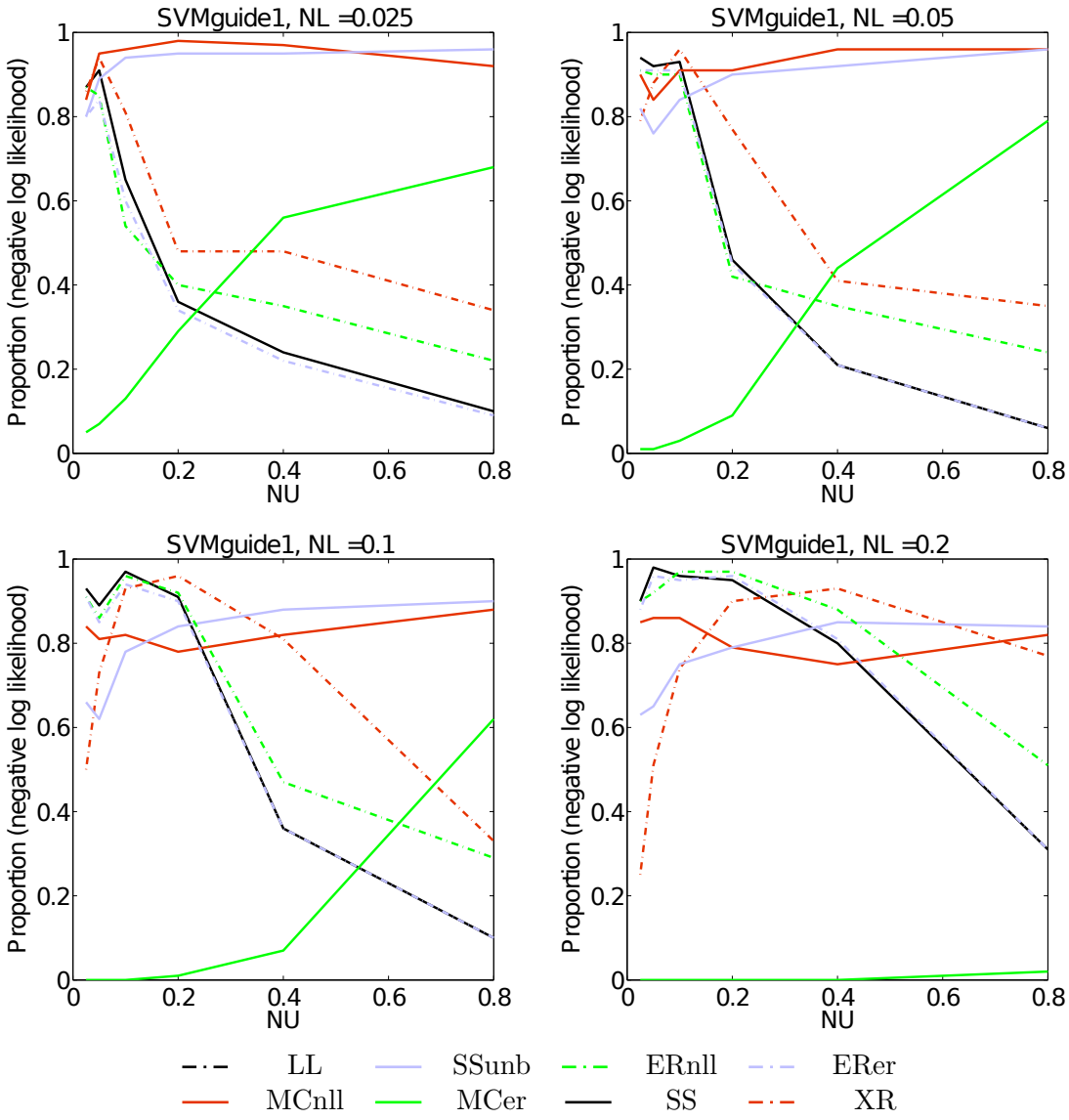


Figure 36: Chance of log likelihood improvement VS labelled data alone - SVMguide data set

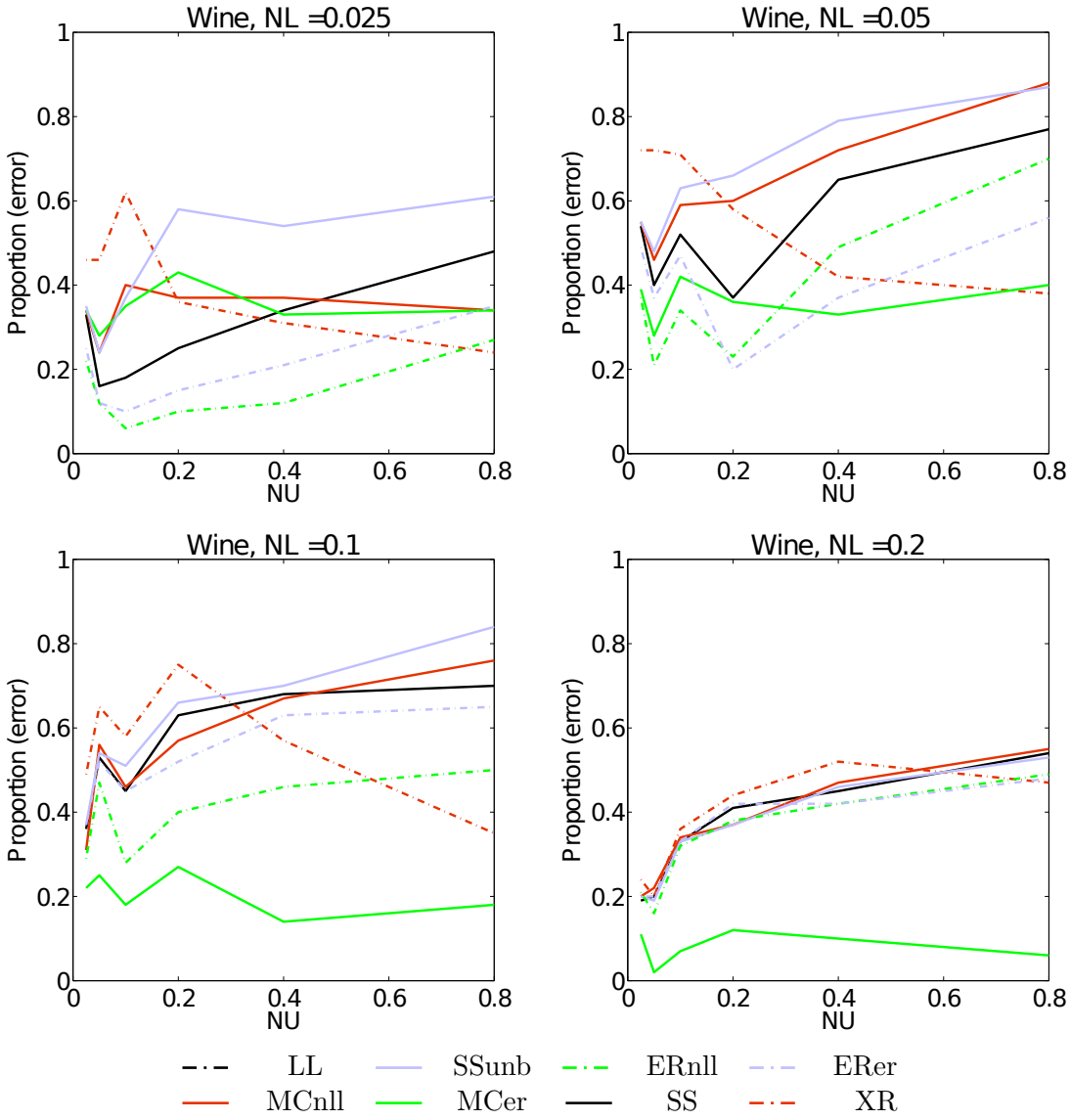


Figure 37: Chance of error rate improvement VS labelled data alone - Wine data set

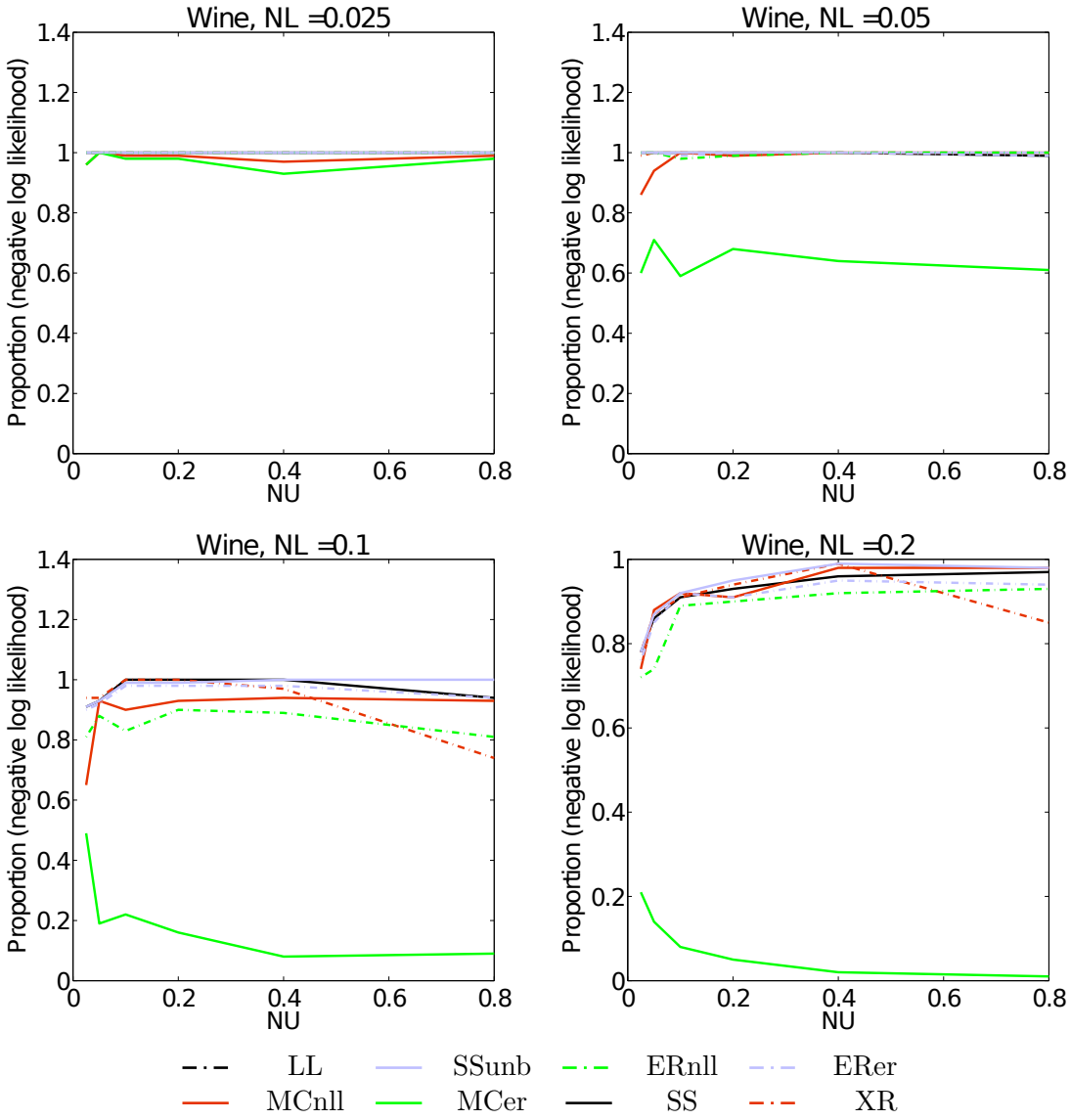


Figure 38: Chance of log likelihood improvement VS labelled data alone - Wine data set

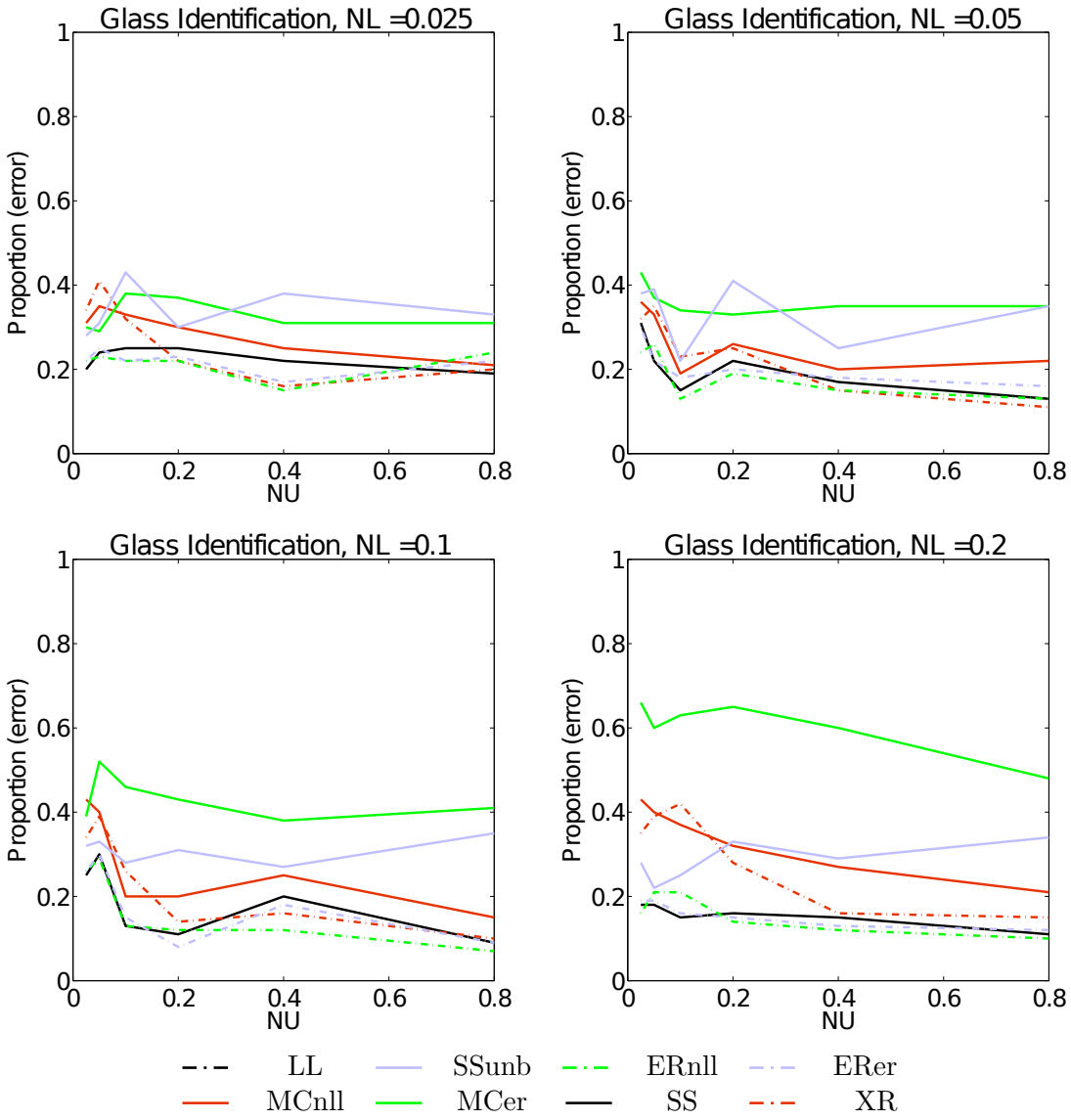


Figure 39: Chance of error rate improvement VS labelled data alone - Glass data set

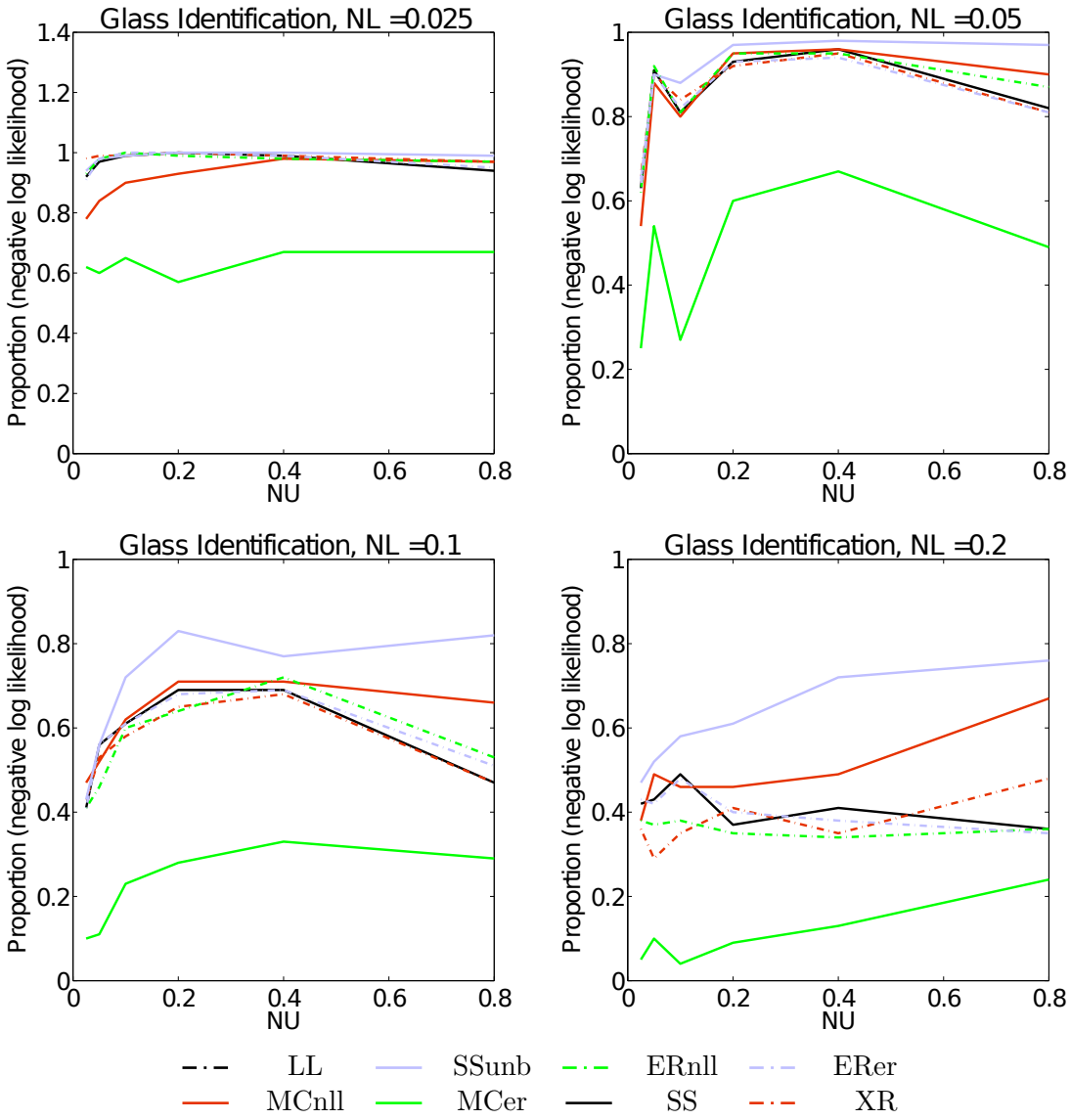


Figure 40: Chance of log likelihood improvement VS labelled data alone - Glass data set

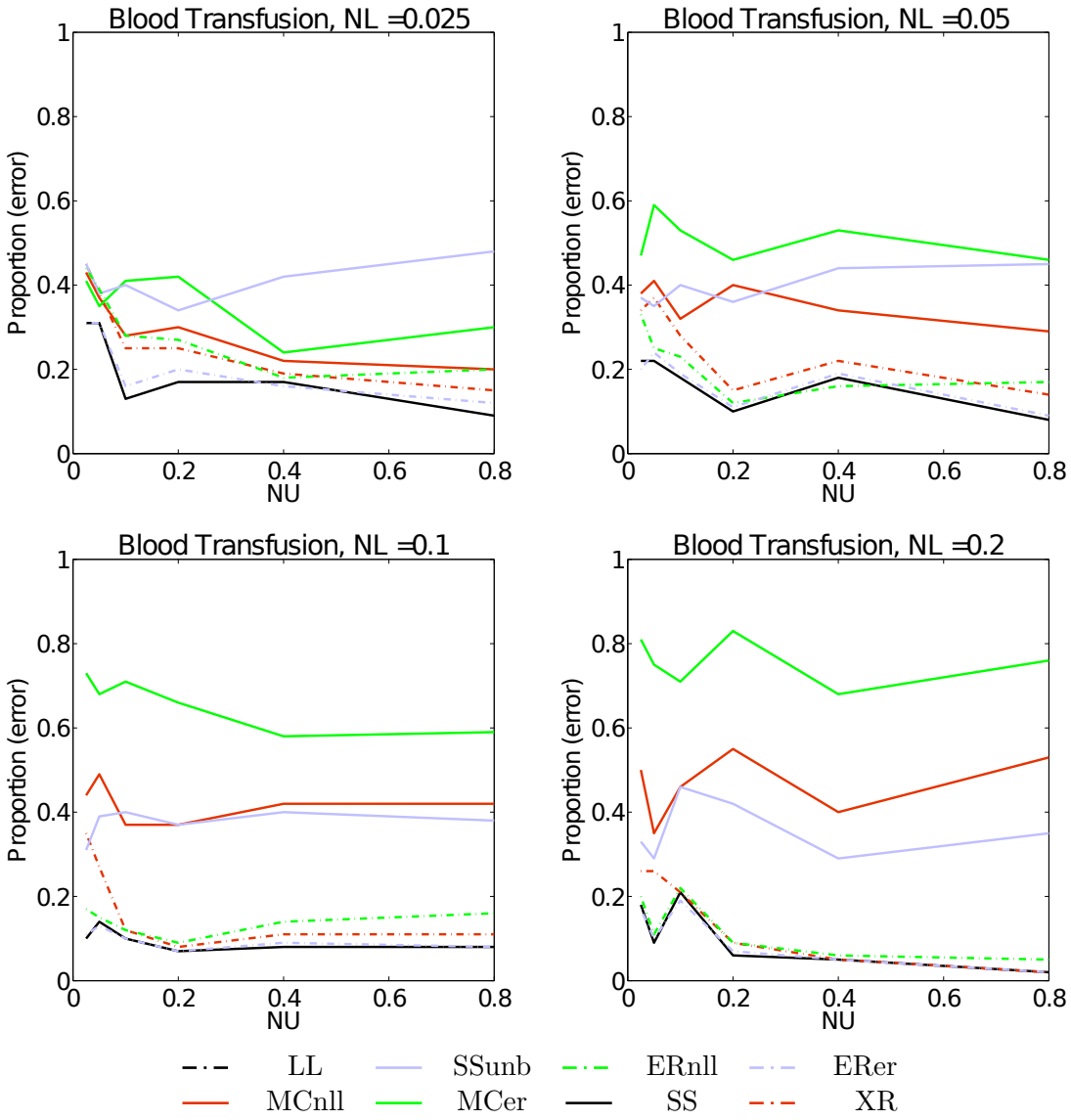


Figure 41: Chance of error rate improvement VS labelled data alone - Blood data set

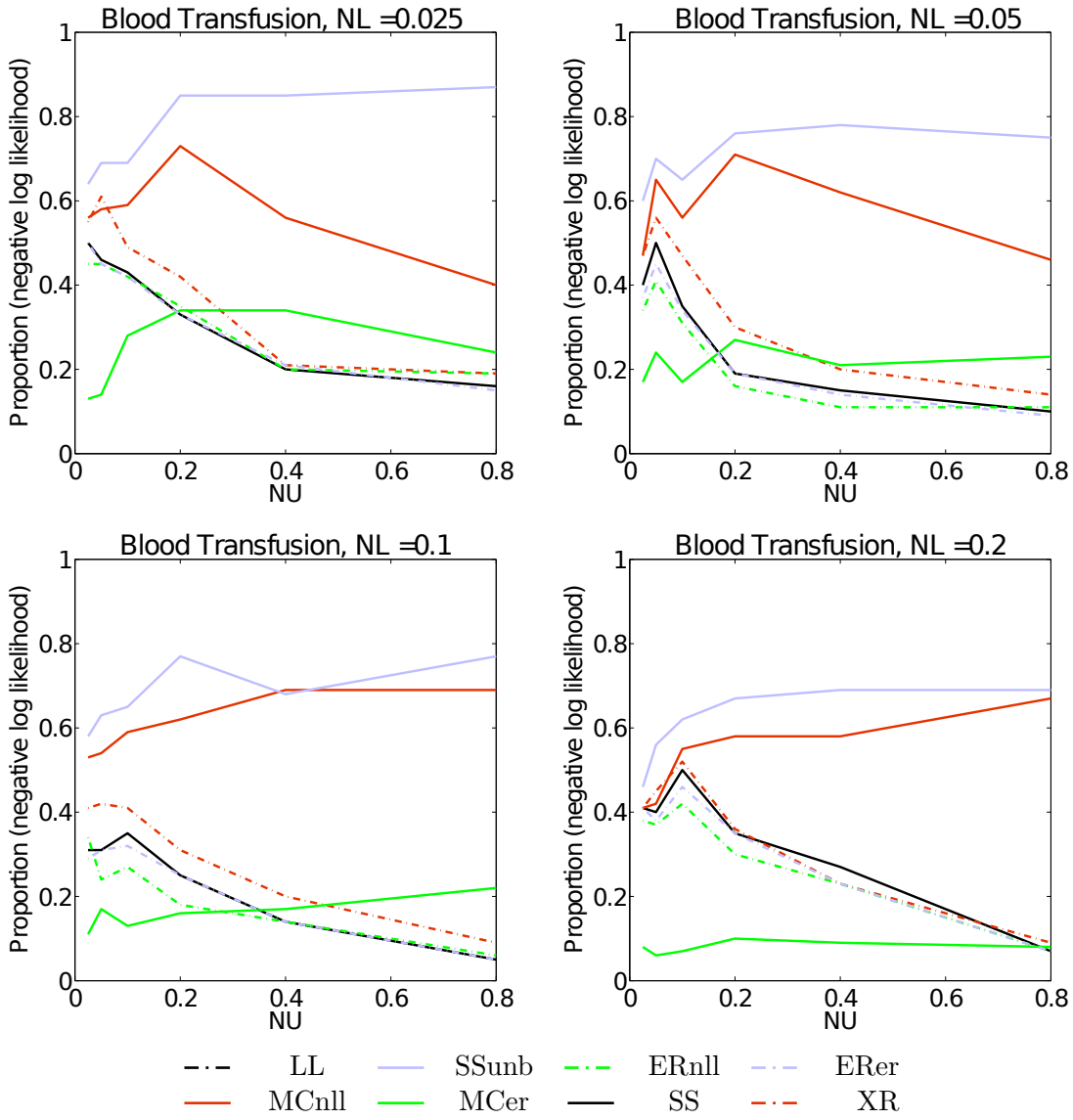


Figure 42: Chance of log likelihood improvement VS labelled data alone - Blood data set

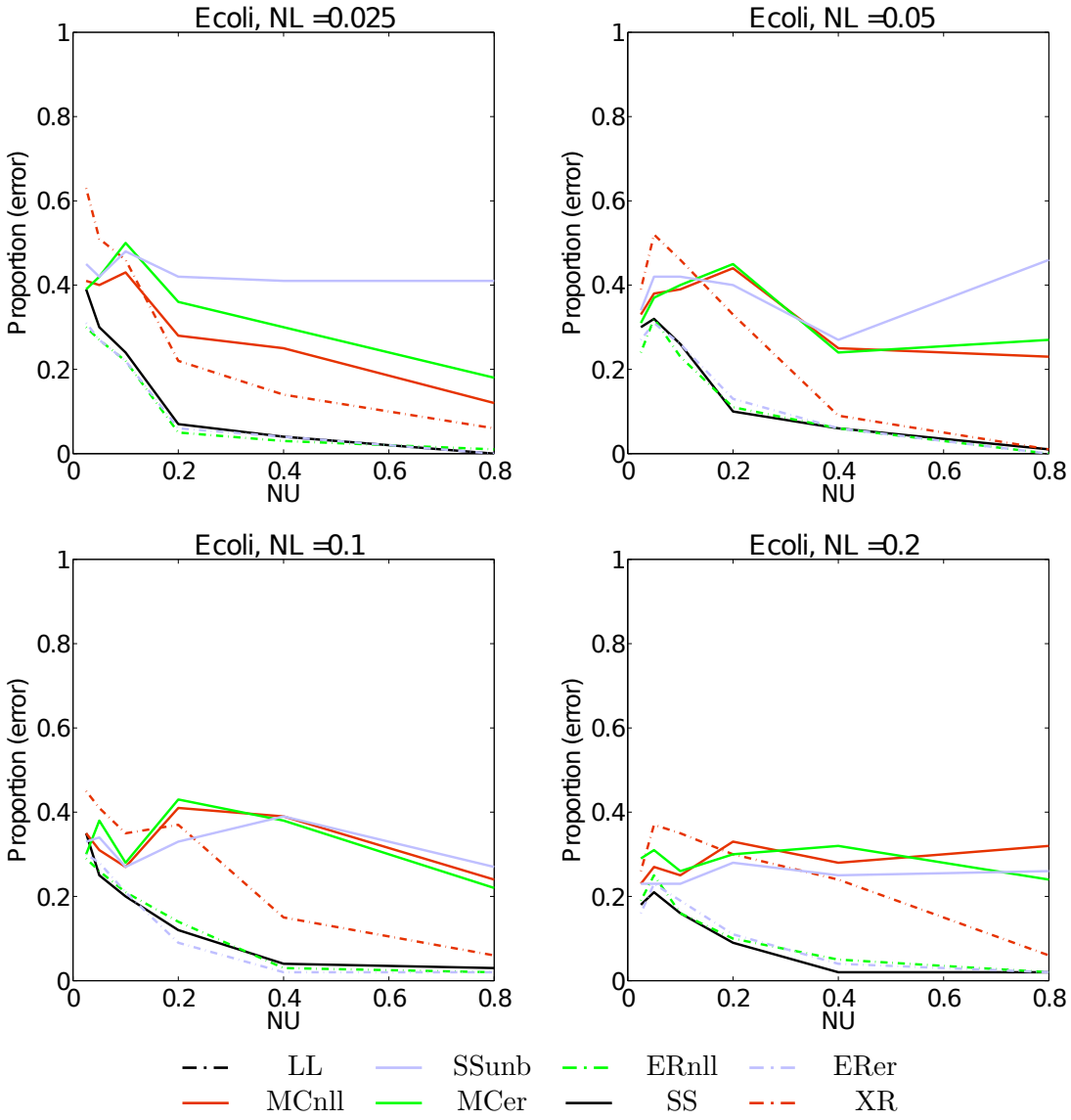


Figure 43: Chance of error rate improvement VS labelled data alone - Ecoli data set

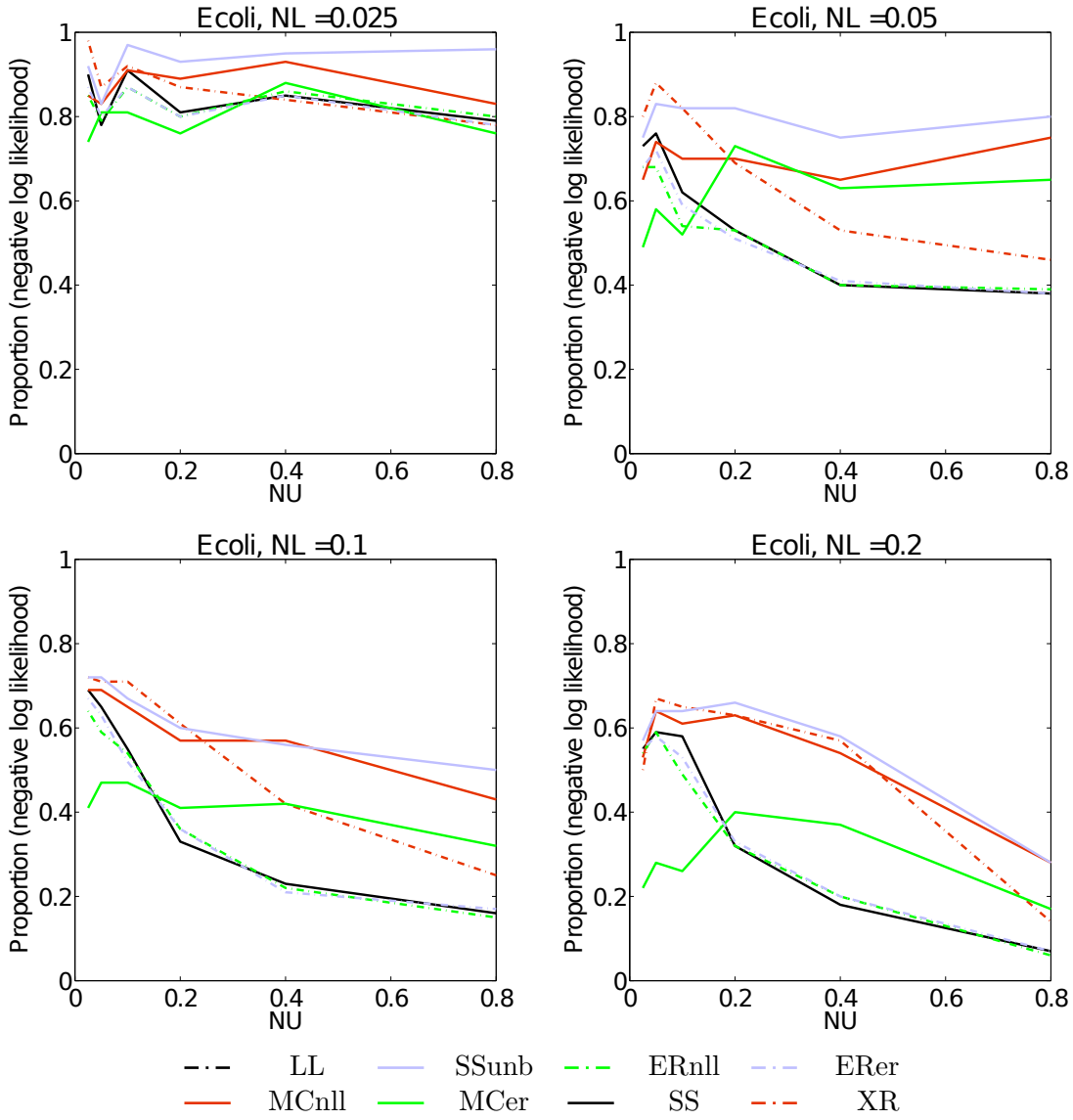


Figure 44: Chance of log likelihood improvement VS labelled data alone - Ecoli data set

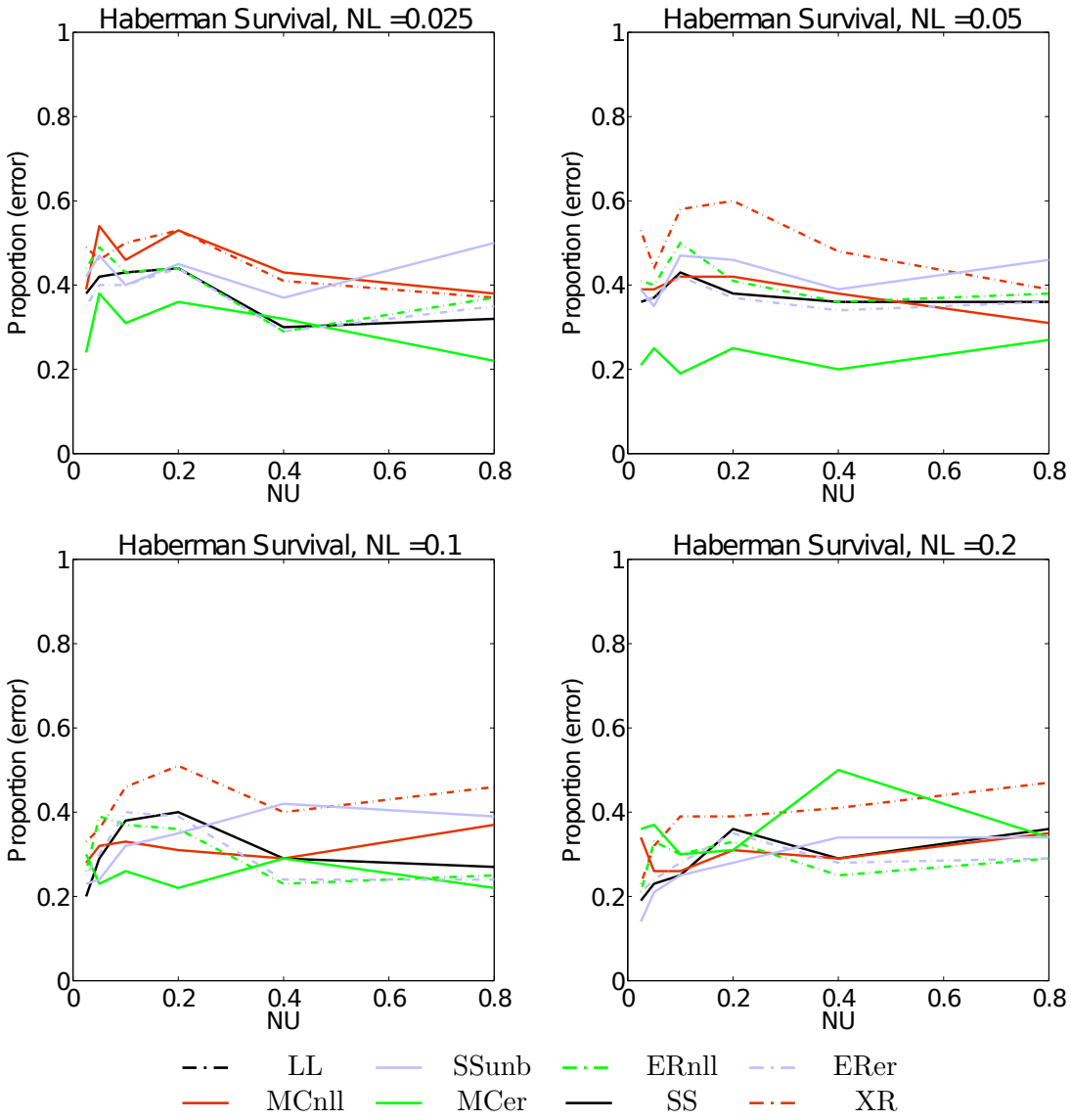


Figure 45: Chance of error rate improvement VS labelled data alone - Haberman data set

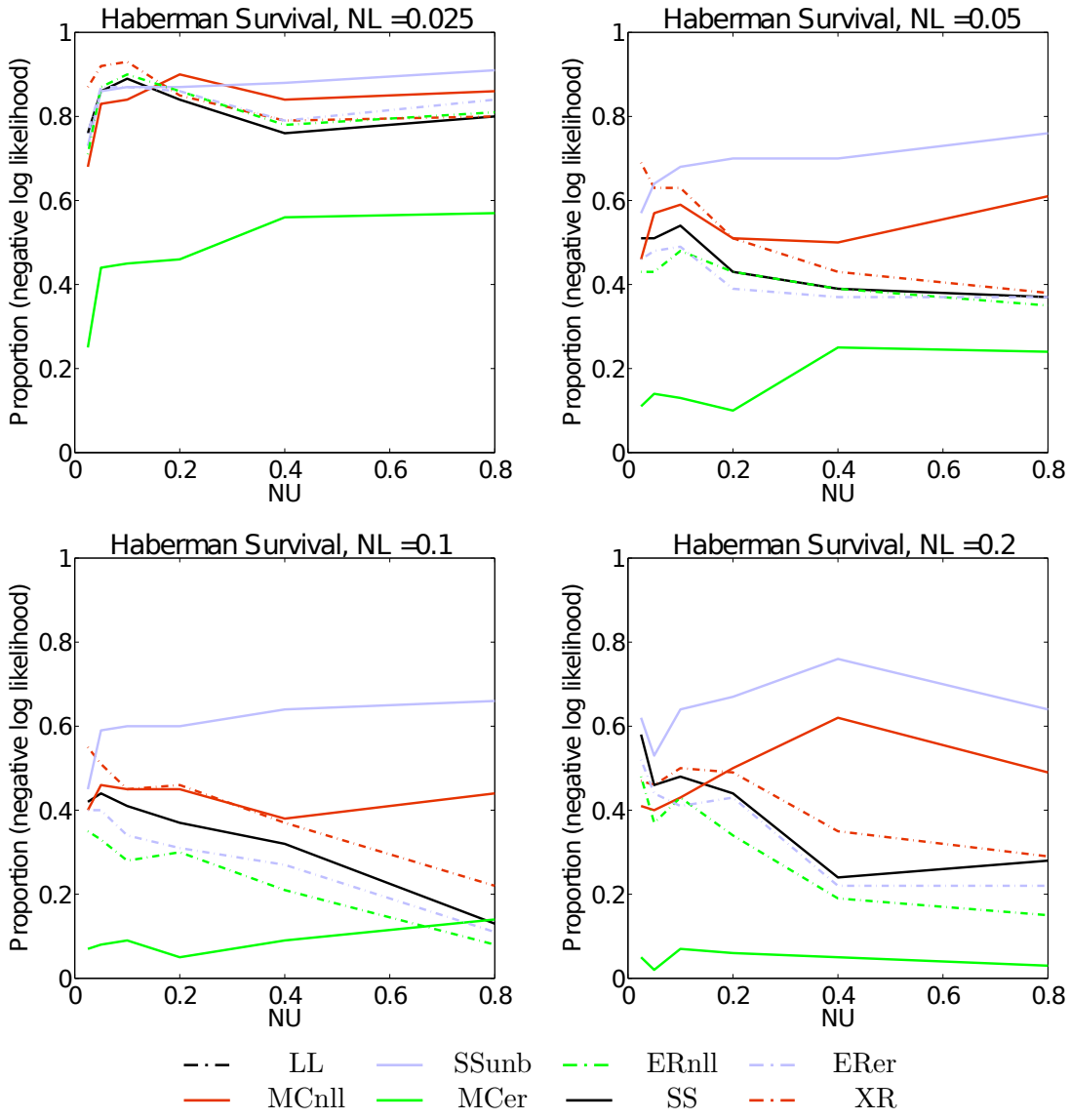


Figure 46: Chance of log likelihood improvement VS labelled data alone - Haberman data set

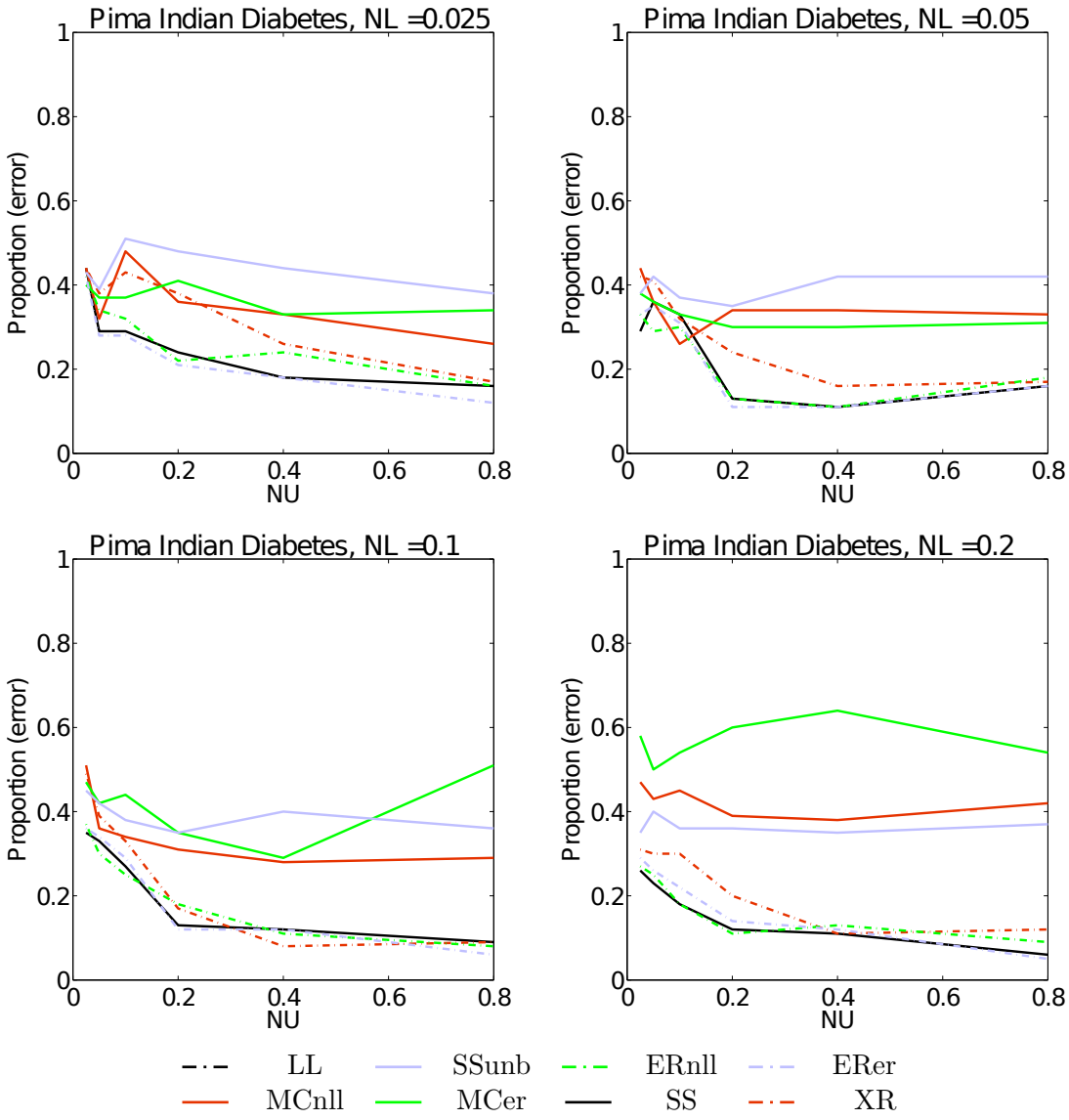


Figure 47: Chance of error rate improvement VS labelled data alone - Pima Indian data set

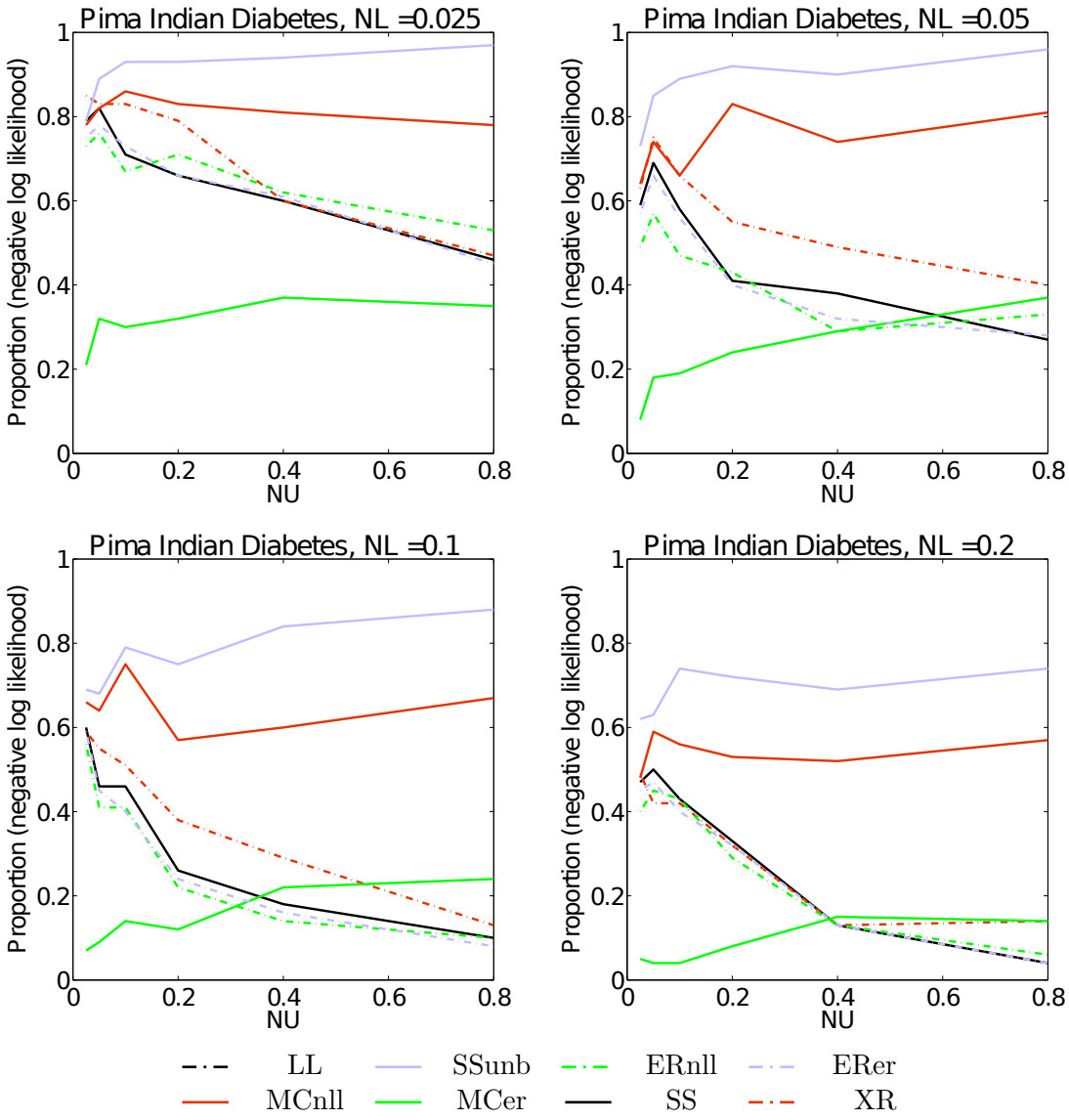


Figure 48: Chance of log likelihood improvement VS labelled data alone - Pima Indian data set

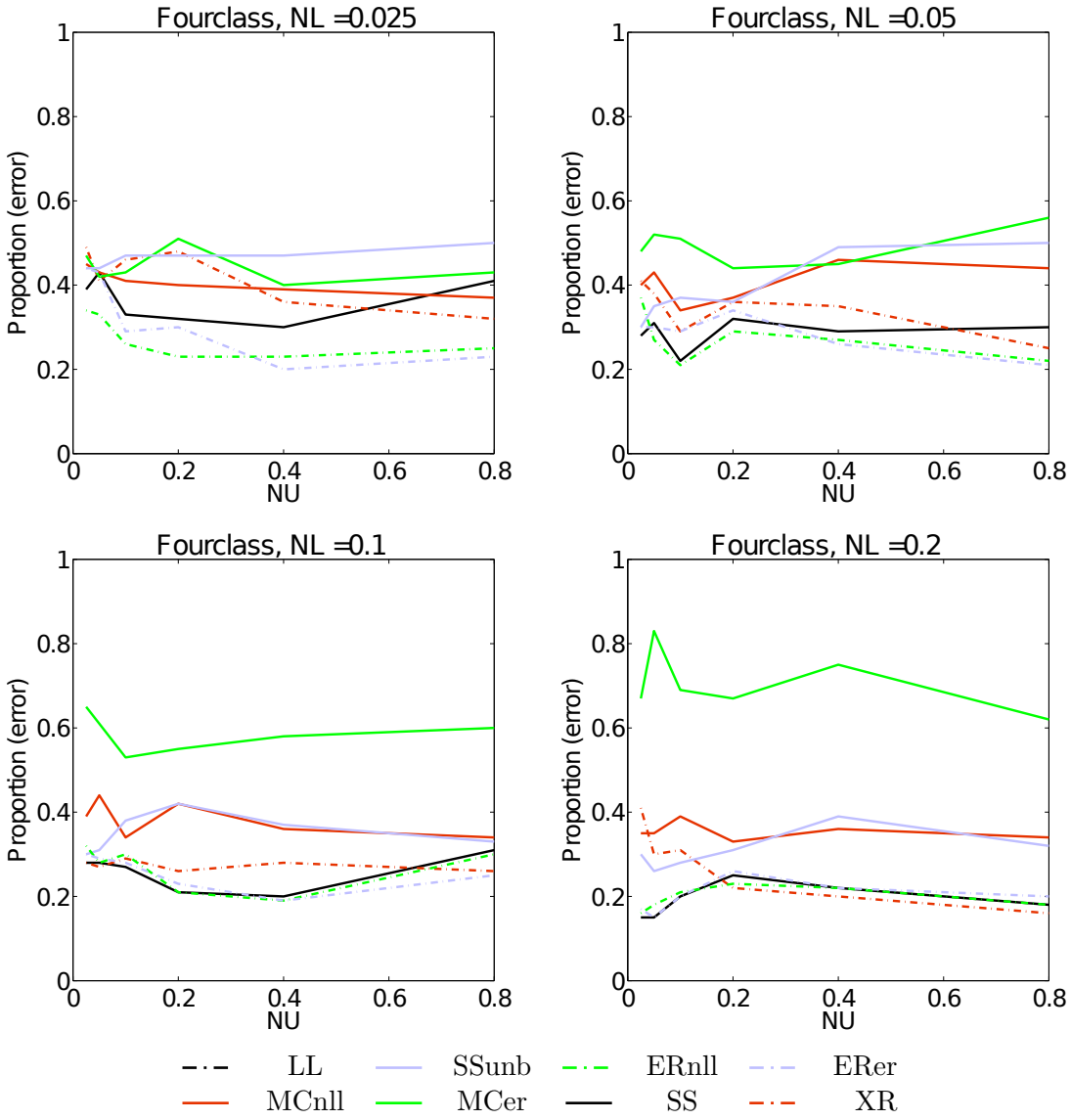


Figure 49: Chance of error rate improvement VS labelled data alone - Fourclass data set

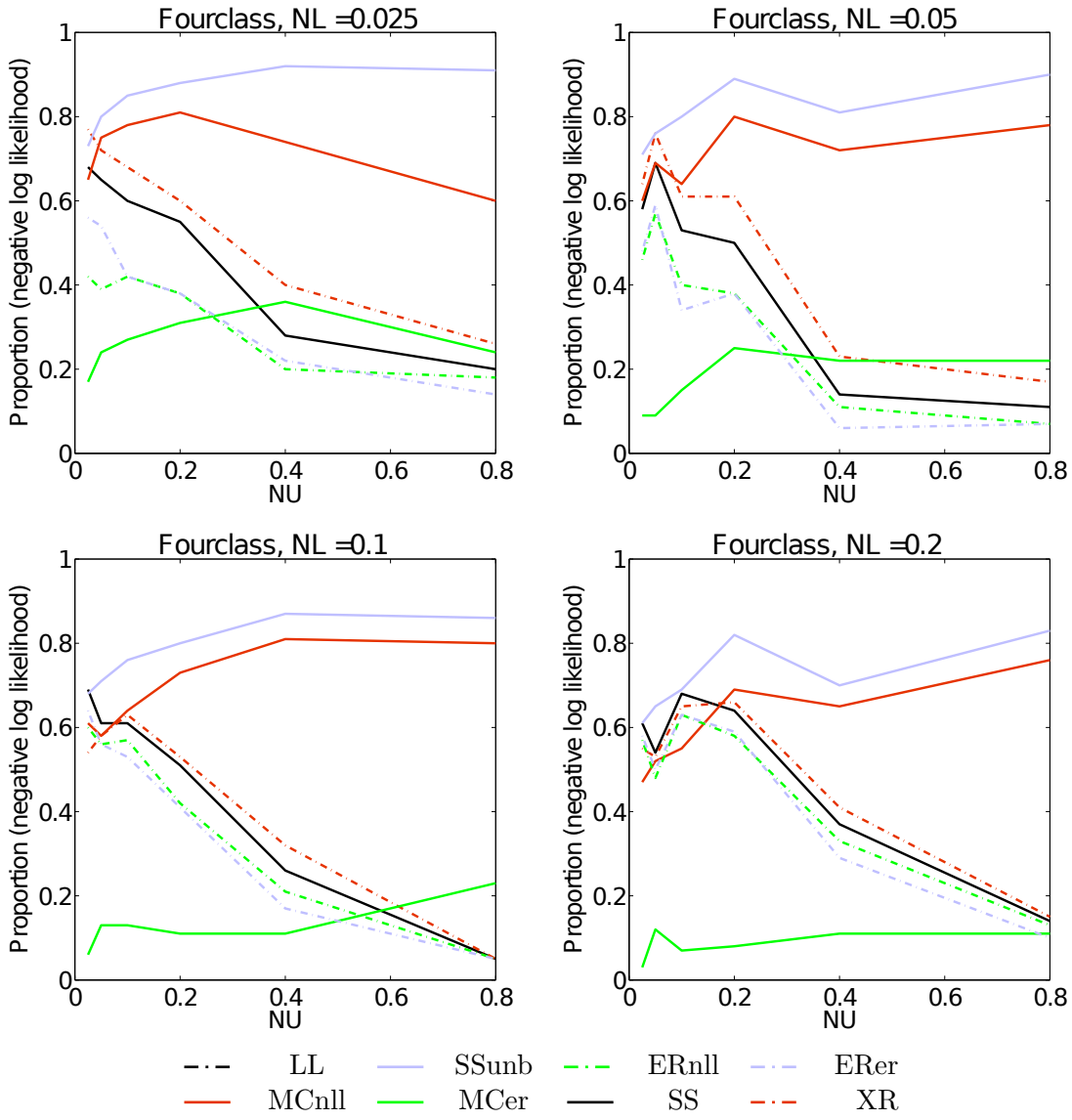


Figure 50: Chance of log likelihood improvement VS labelled data alone - Fourclass data set

References

M. Balcan and A. Blum. An augmented PAC model for semi-supervised learning. In Oliver Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Learning*, pages 383–404. MIT Press, 2005.

C. Beecks, A. Ivanescu, S. Kirchhoff, and T. Seidl. Modeling image similarity by gaussian mixture models and the signature quadratic form distance. In *Computer Vision (ICCV), 2011 IEEE International Conference On*, pages 1754–1761, 2011.

C. Bishop. *Pattern Recognition And Machine Learning*. Springer, 2006.

A. Blum and N. Balcan. A discriminative model for semi-supervised learning. In *Journal Of The ACM (JACM)*, volume 57, pages 19:1–19:46, 2010.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings Of The Workshop On Computational Learning Theory*, pages 92–100, 1998.

V. Castelli and T. Cover. On the exponential value of labeled samples. *Pattern Recognition Letters*, 16(1):105 – 111, 1995.

V. Castelli and T. Cover. The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *Information Theory, IEEE Transactions on*, 42(6):2102 – 2117, 1996.

F. Cozman and I. Cohen. Unlabeled data can degrade classification performance of generative classifiers. In *Fifteenth International Florida Artificial Intelligence Society Conference*, pages 327–331, 2002.

F. Cozman and I. Cohen. Risks of semi-supervised learning: How unlabelled data can degrade performance of generative classifiers. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Learning*, pages 57–72. MIT press, 2006.

F. Cozman, I. Cohen, M. Cirelo, and E. Politécnica. Semi-supervised learning of mixture models. In *Proceedings Of The 20th International Conference On Machine Learning (ICML)*, pages 99–106, 2003.

J. Dillon, K. Balasubramanian, and G. Lebanon. Asymptotic analysis of generative semi-supervised learning. In *Proceedings Of The 27th International Conference On Machine Learning (ICML)*, 2010.

G. Druck, C. Pal, A. McCallum, and X. Zhu. Semi-supervised classification with hybrid generative/discriminative methods. In *KDD '07: Proceedings Of The 13th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, pages 280–289, 2007.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

Y. Grandvalet and Y. Bengio. Entropy Regularization. In *Semi-Supervised Learning*, pages 151–168. MIT Press, 2006.

T. Ho and E. Kleinberg. Building projectable classifiers of arbitrary complexity. In *International Conference On Pattern Recognition (ICPR)*, volume 2, pages 880–885, 1996.

C. Hsu, C. Chang, and C. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

G. Hughes. On the mean accuracy of statistical pattern recognizers. *Information Theory, IEEE Transactions On*, 14(1):55 – 63, 1968.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proceedings Of The 1998 Conference On Advances In Neural Information Processing Systems II*, pages 487–493, 1999.

E. Jaynes. *Probability Theory*. Cambridge University Press, 2003.

H. Kang, S. Yoo, and D. Han. Senti-lexicon and improved naïve bayes algorithms for sentiment analysis of restaurant reviews. *Expert Systems With Applications*, 39(5):6000 – 6010, 2012.

B. Krishnapuram, D. Williams, Y. Xue, A. Hartemink, L. Carin, and M. Figueiredo. On semi-supervised classification. In *Advances In Neural Information Processing Systems (NIPS)*, pages 721–728, 2005.

J. Lagarias, J. Reeds, M. Wright, and P. Wright. Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM Journal Of Optimization*, 9(1):112–147, 1998.

J. Lasserre, C. Bishop, and T. Minka. Principled hybrids of generative and discriminative models. In *Computer Vision And Pattern Recognition (CVPR)*, volume 1, pages 87 – 94, 2006.

J. Lücke and J. Eggert. Expectation truncation and the benefits of preselection in training generative models. In *Journal Of Machine Learning Research (JMLR)*, pages 2855–2900, October 2010.

D. MacKay. *Information Theory, Inference, And Learning Algorithms*. Cambridge University Press, 2003.

G. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning via Expectation Regularization. In *Proceedings Of The 24th International Conference On Machine Learning (ICML)*, pages 593–600, 2007.

A. McCallum, C. Pal, G. Druck, and X. Wang. Multi-conditional learning: Generative/discriminative training for clustering and classification. In *National Conference On Artificial Intelligence*, pages 433–439, 2006.

A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and Naive Bayes. *Advances In Neural Information Processing Systems (NIPS)*, 2:841–848, 2002.

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134, 2000.

J. Nocedal and S. Wright. *Numerical Optimization, Springer Series In Operations Research*. Springer-Verlag, 1999.

J. Ratsaby and S. Venkatesh. Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Conference On Learning Theory (COLT)*, pages 412–417, 1995.

I. Rauschert and R. Collins. A generative model for simultaneous estimation of human body shape and pixel-level segmentation. In *Proceedings Of The 12th European Conference On Computer Vision - Volume Part V*, European Conference On Computer Vision (ECCV), pages 704–717. Springer-Verlag, 2012.

S. Rosset, J. Zhu, H. Zou, and T. Hastie. A method for inferring label sampling mechanisms in semi-supervised learning. In *Advances In Neural Information Processing Systems (NIPS)*, volume 17, pages 1161 – 1168, 2005.

B. Shahshahani and D. Landgrebe. The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *Geoscience And Remote Sensing, IEEE Transactions on*, 32(5):1087 –1095, 1994.

A. Subramanya and J. Bilmes. Soft-supervised learning for text classification. In *Proceedings Of The Conference On Empirical Methods In Natural Language Processing*, pages 1090–1099, 2008.

A. Subramanya and J. Bilmes. Entropic graph regularization in non-parametric semi-supervised classification. In *Advances In Neural Information Processing Systems (NIPS)*, December 2009.

U. Syed and B. Taskar. Semi-supervised learning with adversarially missing label information. In *Advances In Neural Information Processing Systems (NIPS)*, pages 2244–2252, 2010.

M. Szummer and T. Jaakkola. Information Regularization with partially labeled data. In *Advances In Neural Information Processing Systems (NIPS)*, 2002.

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

J. Wang, X. Shen, and W. Pan. On transductive support vector machines. In *Prediction And Discovery*. American Mathematical Society, 2007.

T. Yang and C. Priebe. The effect of model misspecification on semi-supervised classification. *Pattern Analysis And Machine Intelligence (PAMI)*, 33:2093–2103, 2011.

I. Yeh, K. Yang, and T. Ting. Knowledge discovery on RFM model using Bernoulli sequence. *Expert Systems With Applications*, 36(3, Part 2):5866 – 5871, 2009.

T. Zhang. The value of unlabeled data for classification problems. In *International Conference On Machine Learning (ICML)*, pages 1191–1198, 2000.

X. Zhu. Semi-supervised learning literature survey. Technical report, Department of Computer Sciences, University of Wisconsin, Madison, 2005.

F. Zhuang, P. Luo, Z. Shen, Q. He, Y. Xiong, Z. Shi, and H. Xiong. Mining distinction and commonality across multiple domains using generative model for text classification. *Knowledge And Data Engineering, IEEE Transactions On*, 24(11):2025–2039, 2012.

Journal of Machine Learning Research 15 (2014) 445-488 Submitted 3/13; Revised 8/13; Published 2/14

Node-Based Learning of Multiple Gaussian Graphical Models

Karthik Mohan KARNA@UW.EDU
Palma London PALONDON@UW.EDU
Maryam Fazel MFAZEL@UW.EDU
Department of Electrical Engineering
University of Washington
Seattle WA, 98195

Daniela Witten DWITTEN@UW.EDU
Department of Biostatistics
University of Washington
Seattle WA, 98195

Su-In Lee SUINLEE@CS.WASHINGTON.EDU
Departments of Computer Science and Engineering, Genome Sciences
University of Washington
Seattle WA, 98195

Editor: Saharon Rosset

Abstract

We consider the problem of estimating high-dimensional Gaussian graphical models corresponding to a single set of variables under several distinct conditions. This problem is motivated by the task of recovering transcriptional regulatory networks on the basis of gene expression data containing heterogeneous samples, such as different disease states, multiple species, or different developmental stages. We assume that most aspects of the conditional dependence networks are shared, but that there are some structured differences between them. Rather than assuming that similarities and differences between networks are driven by individual edges, we take a *node-based* approach, which in many cases provides a more intuitive interpretation of the network differences. We consider estimation under two distinct assumptions: (1) differences between the K networks are due to individual nodes that are *perturbed* across conditions, or (2) similarities among the K networks are due to the presence of *common hub nodes* that are shared across all K networks. Using a *row-column overlap norm* penalty function, we formulate two convex optimization problems that correspond to these two assumptions. We solve these problems using an alternating direction method of multipliers algorithm, and we derive a set of necessary and sufficient conditions that allows us to decompose the problem into independent subproblems so that our algorithm can be scaled to high-dimensional settings. Our proposal is illustrated on synthetic data, a webpage data set, and a brain cancer gene expression data set.

Keywords: graphical model, structured sparsity, alternating direction method of multipliers, gene regulatory network, lasso, multivariate normal

1. Introduction

Graphical models encode the conditional dependence relationships among a set of p variables (Lauritzen, 1996). They are a tool of growing importance in a number of fields, including finance, biology, and computer vision. A graphical model is often referred to as a conditional dependence *network*, or simply as a *network*. Motivated by network terminology, we can refer to the p variables in a graphical model as *nodes*. If a pair of variables are conditionally dependent, then there is an *edge* between the corresponding pair of nodes; otherwise, no edge is present.

Suppose that we have n observations that are independently drawn from a multivariate normal distribution with covariance matrix Σ. Then the corresponding *Gaussian graphical model* (GGM) that describes the conditional dependence relationships among the variables is encoded by the sparsity pattern of the inverse covariance matrix, Σ^{-1} (see, e.g., Mardia et al., 1979; Lauritzen, 1996). That is, the jth and j'th variables are conditionally independent if and only if $(\Sigma^{-1})_{jj'} = 0$. Unfortunately, when $p > n$, obtaining an accurate estimate of Σ^{-1} is challenging. In such a scenario, we can use prior information—such as the knowledge that many of the pairs of variables are conditionally independent—in order to more accurately estimate Σ^{-1} (see, e.g., Yuan and Lin, 2007a; Friedman et al., 2007; Banerjee et al., 2008).

In this paper, we consider the task of estimating K GGMs on a single set of p variables under the assumption that the GGMs are similar, with certain structured differences. As a motivating example, suppose that we have access to gene expression measurements for n_1 lung cancer samples and n_2 normal lung samples, and that we would like to estimate the gene regulatory networks underlying the normal and cancer lung tissue. We can model each of these regulatory networks using a GGM. We have two obvious options.

1. We can estimate a single network on the basis of all $n_1 + n_2$ tissue samples. But this approach overlooks fundamental differences between the true lung cancer and normal gene regulatory networks.

2. We can estimate separate networks based on the n_1 cancer and n_2 normal samples. However, this approach fails to exploit substantial commonality of the two networks, such as lung-specific pathways.

In order to effectively make use of the available data, we need a principled approach for jointly estimating the two networks in such a way that the two estimates are encouraged to be quite similar to each other, while allowing for certain structured differences. In fact, these differences may be of scientific interest.

Another example of estimating multiple GGMs arises in the analysis of the conditional dependence relationships among p stocks at two distinct points in time. We might be interested in detecting stocks that have differential connectivity with all other stocks across the two time points, as these likely correspond to companies that have undergone significant changes. Yet another example occurs in the field of neuroscience, in which it is of interest to learn how the connectivity of neurons changes over time.

Past work on joint estimation of multiple GGMs has assumed that individual *edges* are shared or differ across conditions (see, e.g., Kolar et al., 2010; Zhang and Wang, 2010; Guo et al., 2011; Danaher et al., 2013); here we refer to such approaches as *edge-based*. In

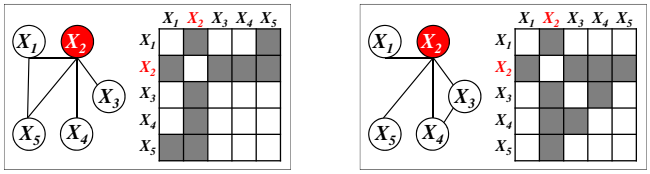
this paper, we instead take a *node-based* approach: we seek to estimate K GGMs under the assumption that similarities and differences between networks are driven by individual *nodes* whose patterns of connectivity to other nodes are shared across networks, or differ between networks. As we will see, node-based learning is more powerful than edge-based learning, since it more fully exploits our prior assumptions about the similarities and differences between networks.

More specifically, in this paper we consider two types of shared network structure.

1. Certain nodes serve as highly-connected *hub* nodes. We assume that the same nodes serve as hubs in each of the K networks. Figure 1 illustrates a toy example of this setting, with $p = 5$ nodes and $K = 2$ networks. In this example, the second variable, X_2, serves as a hub node in each network. In the context of transcriptional regulatory networks, X_2 might represent a gene that encodes a *transcription factor* that regulates a large number of downstream genes in all K contexts. We propose the *common hub (co-hub) node joint graphical lasso* (CNJGL), a convex optimization problem for estimating GGMs in this setting.

2. The networks differ due to particular nodes that are *perturbed* across conditions, and therefore have a completely different connectivity pattern to other nodes in the K networks. Figure 2 displays a toy example, with $p = 5$ nodes and $K = 2$ networks; here we see that all of the network differences are driven by perturbation in the second variable, X_2. In the context of transcriptional regulatory networks, X_2 might represent a gene that is mutated in a particular condition, effectively disrupting its conditional dependence relationships with other genes. We propose the *perturbed-node joint graphical lasso* (PNJGL), a convex optimization problem for estimating GGMs in this context.

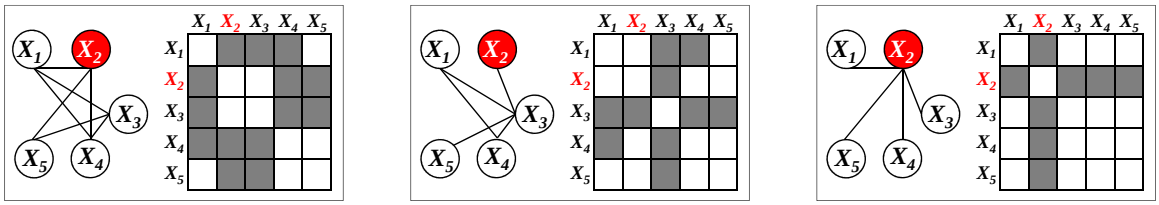
Node-based learning of multiple GGMs is challenging, due to complications resulting from symmetry of the precision matrices. In this paper, we overcome this problem through the use of a new convex regularizer.



(a) (b)

Figure 1: Two networks share a *common hub* (co-hub) node. X_2 serves as a hub node in both networks. *(a):* Network 1 and its adjacency matrix. *(b):* Network 2 and its adjacency matrix.

The rest of this paper is organized as follows. We introduce some relevant background material in Section 2. In Section 3, we present the *row-column overlap norm* (RCON), a regularizer that encourages a matrix to have a support that is the *union of a set of rows and columns*. We apply the RCON penalty to a pair of inverse covariance matrices, or to the difference between a pair of inverse covariance matrices, in order to obtain the CNJGL



(a) (b) (c)

Figure 2: Two networks that differ due to *node perturbation* of X_2. *(a):* Network 1 and its adjacency matrix. *(b):* Network 2 and its adjacency matrix. *(c): Left:* Edges that differ between the two networks. *Right:* Shaded cells indicate edges that differ between Networks 1 and 2.

and PNJGL formulations just described. In Section 4, we propose an *alternating direction method of multipliers* (ADMM) algorithm in order to solve these two convex formulations. In order to scale this algorithm to problems with many variables, in Section 5 we introduce a set of simple conditions on the regularization parameters that indicate that the problem can be broken down into many independent subproblems, leading to substantial algorithm speed-ups. In Section 6, we apply CNJGL and PNJGL to synthetic data, and in Section 7 we apply them to gene expression data and to webpage data. The Discussion is in Section 8. Proofs are in the Appendix.

A preliminary version of some of the ideas in this paper appear in Mohan et al. (2012). There the PNJGL formulation was proposed, along with an ADMM algorithm. Here we expand upon that formulation and present the CNJGL formulation, an ADMM algorithm for solving it, as well as comprehensive results on both real and simulated data. Furthermore, in this paper we discuss theoretical conditions for computational speed-ups, which are critical to application of both PNJGL and CNJGL to data sets with many variables.

2. Background on High-Dimensional GGM Estimation

In this section, we review the literature on learning Gaussian graphical models.

2.1 The Graphical Lasso for Estimating a Single GGM

As was mentioned in Section 1, estimating a single GGM on the basis of n independent and identically distributed observations from a $N_p(\mathbf{0}, \Sigma)$ distribution amounts to learning the sparsity structure of Σ^{-1} (Mardia et al., 1979; Lauritzen, 1996). When $n > p$, one can estimate Σ^{-1} by maximum likelihood. But in high dimensions when p is large relative to n, this is not possible because the empirical covariance matrix is singular. Consequently, a number of authors (among others, Yuan and Lin, 2007a; Friedman et al., 2007; Ravikumar et al., 2008; Banerjee et al., 2008; Scheinberg et al., 2010; Hsieh et al., 2011) have considered maximizing the penalized log likelihood

$$\underset{\Theta \in \mathbb{S}^p_{++}}{\text{maximize}} \left\{ \log\det\Theta - \text{trace}(S\Theta) - \lambda\|\Theta\|_1 \right\}, \tag{1}$$

where S is the empirical covariance matrix, λ is a nonnegative tuning parameter, $\mathbb{S}^p_{++}$ denotes the set of positive definite matrices of size p, and $\|\Theta\|_1 = \sum_{i,j} |\Theta_{ij}|$. The solution

to (1) serves as an estimate of Σ^{-1}, and a zero element in the solution corresponds to a pair of variables that are estimated to be conditionally independent. Due to the ℓ_1 penalty (Tibshirani, 1996) in (1), this estimate will be positive definite for any $\lambda > 0$, and sparse when λ is sufficiently large. We refer to (1) as the *graphical lasso*. Problem (1) is convex, and efficient algorithms for solving it are available (among others, Friedman et al., 2007; Banerjee et al., 2008; Rothman et al., 2008; D'Aspremont et al., 2008; Scheinberg et al., 2010; Witten et al., 2011).

2.2 The Joint Graphical Lasso for Estimating Multiple GGMs

Several formulations have recently been proposed for extending the graphical lasso (1) to the setting in which one has access to a number of observations from K distinct conditions, each with measurements on the same set of p variables. The goal is to estimate a graphical model for each condition under the assumption that the K networks share certain characteristics but are allowed to differ in certain structured ways. Guo et al. (2011) take a non-convex approach to solving this problem. Zhang and Wang (2010) take a convex approach, but use a least squares loss function rather than the negative Gaussian log likelihood. Here we review the convex formulation of Danaher et al. (2013), which forms the starting point for the proposal in this paper.

Suppose that $X_1^k, \ldots, X_{n_k}^k \in \mathbb{R}^p$ are independent and identically distributed from a $N_p(\mathbf{0}, \Sigma^k)$ distribution, for $k = 1, \ldots, K$. Here n_k is the number of observations in the kth condition, or class. Letting S^k denote the empirical covariance matrix for the kth class, we can maximize the penalized log likelihood

$$\underset{\Theta^1 \in \mathbb{S}_{++}^p, \ldots, \Theta^K \in \mathbb{S}_{++}^p}{\text{maximize}} \left\{ L(\Theta^1, \ldots, \Theta^K) - \lambda_1 \sum_{k=1}^K \|\Theta^k\|_1 - \lambda_2 \sum_{i \neq j} P(\Theta_{ij}^1, \ldots, \Theta_{ij}^K) \right\}, \quad (2)$$

where $L(\Theta^1, \ldots, \Theta^K) = \sum_{k=1}^K n_k \left(\log\det \Theta^k - \text{trace}(S^k \Theta^k) \right)$, λ_1 and λ_2 are nonnegative tuning parameters, and $P(\Theta_{ij}^1, \ldots, \Theta_{ij}^K)$ is a convex penalty function applied to each off-diagonal element of $\Theta^1, \ldots, \Theta^K$ in order to encourage similarity among them. Then the $\hat{\Theta}^1, \ldots, \hat{\Theta}^K$ that solve (2) serve as estimates for $(\Sigma^1)^{-1}, \ldots, (\Sigma^K)^{-1}$. Danaher et al. (2013) refer to (2) as the *joint graphical lasso* (JGL). In particular, they consider the use of a *fused lasso penalty* (Tibshirani et al., 2005),

$$P(\Theta_{ij}^1, \ldots, \Theta_{ij}^K) = \sum_{k<k'} |\Theta_{ij}^k - \Theta_{ij}^{k'}|, \quad (3)$$

on the differences between pairs of network edges, as well as a *group lasso penalty* (Yuan and Lin, 2007b),

$$P(\Theta_{ij}^1, \Theta_{ij}^2, \ldots, \Theta_{ij}^K) = \sqrt{\sum_{k=1}^K (\Theta_{ij}^k)^2}, \quad (4)$$

on the edges themselves. Danaher et al. (2013) refer to problem (2) combined with (3) as the *fused graphical lasso* (FGL), and to (2) combined with (4) as the *group graphical lasso* (GGL).

FGL encourages the K network estimates to have identical edge values, whereas GGL encourages the K network estimates to have a shared pattern of sparsity. Both the FGL and GGL optimization problems are convex. An approach related to FGL and GGL is proposed in Hara and Washio (2013).

Because FGL and GGL borrow strength across all available observations in estimating each network, they can lead to much more accurate inference than simply learning each of the K networks separately.

But both FGL and GGL take an *edge-based* approach: they assume that differences between and similarities among the networks arise from individual edges. In this paper, we propose a *node-based* formulation that allows for more powerful estimation of multiple GGMs, under the assumption that network similarities and differences arise from *nodes* whose connectivity patterns to other nodes are shared or disrupted across conditions.

3. Node-Based Joint Graphical Lasso

In this section, we first discuss the failure of a naive approach for node-based learning of multiple GGMs. We then present a norm that will play a critical role in our formulations for this task. Finally, we discuss two approaches for node-based learning of multiple GGMs.

3.1 Why is Node-Based Learning Challenging?

At first glance, node-based learning of multiple GGMs seems straightforward. For instance, consider the task of estimating $K = 2$ networks under the assumption that the connectivity patterns of individual nodes differ across the networks. It seems that we could simply modify (2) combined with (3) as follows,

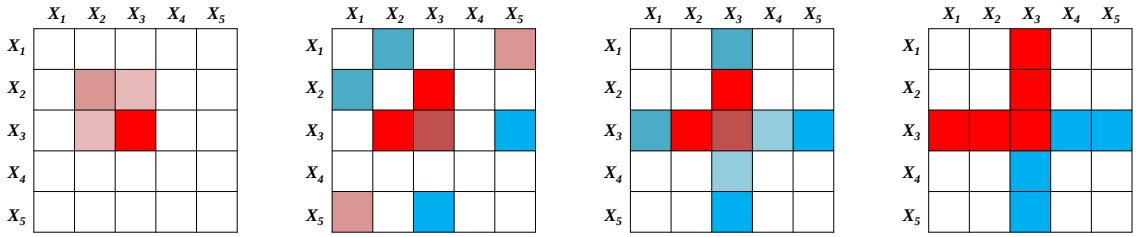
$$\underset{\Theta^1 \in \mathbb{S}^p_{++}, \Theta^2 \in \mathbb{S}^p_{++}}{\text{maximize}} \left\{ L(\Theta^1, \Theta^2) - \lambda_1 \|\Theta^1\|_1 - \lambda_1 \|\Theta^2\|_1 - \lambda_2 \sum_{j=1}^{p} \|\Theta^1_j - \Theta^2_j\|_2 \right\}, \tag{5}$$

where Θ^k_j is the jth column of the matrix Θ^k. This amounts to applying a *group lasso* (Yuan and Lin, 2007b) penalty to the columns of $\Theta^1 - \Theta^2$. Equation (5) seems to accomplish our goal of encouraging $\Theta^1_j = \Theta^2_j$. We will refer to this as the *naive group lasso* approach.

In (5), we have applied the group lasso using p groups; the jth group is the jth column of $\Theta^1 - \Theta^2$. Due to the symmetry of Θ^1 and Θ^2, there is substantial overlap among the p groups: the (i, j)th element of $\Theta^1 - \Theta^2$ is contained in both the ith and jth groups. In the presence of overlapping groups, the group lasso penalty yields estimates whose *support is the complement of the union of groups* (Jacob et al., 2009; Obozinski et al., 2011). Figure 3(a) displays a simple example of the results obtained if we attempt to estimate $(\Sigma^1)^{-1} - (\Sigma^2)^{-1}$ using (5). The figure reveals that (5) cannot be used to detect node perturbation.

A naive approach to co-hub detection is challenging for a similar reason. Recall that the jth node is a co-hub if the jth columns of both Θ^1 and Θ^2 contain predominantly non-zero elements, and let $\text{diag}(\Theta)$ denote a matrix consisting of the diagonal elements of Θ. It is tempting to formulate the optimization problem

$$\underset{\Theta^1 \in \mathbb{S}^p_{++}, \Theta^2 \in \mathbb{S}^p_{++}}{\text{maximize}} \left\{ L(\Theta^1, \Theta^2) - \lambda_1 \|\Theta^1\|_1 - \lambda_1 \|\Theta^2\|_1 - \lambda_2 \sum_{j=1}^{p} \left\| \begin{bmatrix} \Theta^1 - \text{diag}(\Theta^1) \\ \Theta^2 - \text{diag}(\Theta^2) \end{bmatrix}_j \right\|_2 \right\},$$



(a) Naive group lasso (b) RCON: ℓ_1/ℓ_1 (c) RCON: ℓ_1/ℓ_2 (d) RCON: ℓ_1/ℓ_∞

Figure 3: Toy example of the results from applying various penalties in order to estimate a 5×5 matrix, under a symmetry constraint. Zero elements are shown in white; non-zero elements are shown in shades of red (positive elements) and blue (negative elements). *(a):* The naive group lasso applied to the columns of the matrix yields non-zero elements that are the *intersection*, rather than the *union*, of a set of rows and columns. *(b):* The RCON penalty using an ℓ_1/ℓ_1 norm results in unstructured sparsity in the estimated matrix. *(c):* The RCON penalty using an ℓ_1/ℓ_2 norm results in entire rows and columns of non-zero elements. *(d):* The RCON penalty using an ℓ_1/ℓ_∞ norm results in entire rows and columns of non-zero elements; many take on a single maximal (absolute) value.

where the group lasso penalty encourages the off-diagonal elements of many of the columns to be simultaneously zero in Θ^1 and Θ^2. Unfortunately, once again, the presence of overlapping groups encourages the support of the matrices Θ^1 and Θ^2 to be the intersection of a set of rows and columns, as in Figure 3(a), rather than the union of a set of rows and columns.

3.2 Row-Column Overlap Norm

Detection of perturbed nodes or co-hub nodes requires a penalty function that, when applied to a matrix, yields a support given by the union of a set of rows and columns. We now propose the *row-column overlap norm* (RCON) for this task.

Definition 1 *The row-column overlap norm (RCON) induced by a matrix norm $\|.\|$ is defined as*

$$\Omega(\Theta^1,\Theta^2,\ldots,\Theta^K) = \min_{V^1,V^2,\ldots,V^K} \left\| \begin{bmatrix} V^1 \\ V^2 \\ \vdots \\ V^K \end{bmatrix} \right\| \quad \text{subject to} \quad \Theta^k = V^k + (V^k)^T \text{ for } k = 1,\ldots,K.$$

It is easy to check that Ω is indeed a norm for all matrix norms $\|.\|$. Also, when $\|.\|$ is symmetric in its argument, that is, $\|V\| = \|V^T\|$, then

$$\Omega(\Theta^1, \Theta^2, \ldots, \Theta^K) = \frac{1}{2} \left\| \begin{bmatrix} \Theta^1 \\ \Theta^2 \\ \vdots \\ \Theta^K \end{bmatrix} \right\|.$$

Thus if $\|\cdot\|$ is an ℓ_1/ℓ_1 norm, then $\Omega(\Theta^1, \Theta^2, \ldots, \Theta^K) = \frac{1}{2}\sum_{k=1}^K \sum_{i,j} |\Theta^k_{ij}|$.

We now discuss the motivation behind Definition 1. Any symmetric matrix Θ^k can be (non-uniquely) decomposed as $V^k + (V^k)^T$; note that V^k need not be symmetric. This amounts to interpreting Θ^k as a set of columns (the columns of V^k) *plus* a set of rows (the columns of V^k, transposed). In this paper, we are interested in the particular case of RCON penalties where $\|.\|$ is an ℓ_1/ℓ_q norm, given by $\|V\| = \sum_{j=1}^p \|V_j\|_q$, where $1 \leq q \leq \infty$. With a little abuse of notation, we will let Ω_q denote Ω when $\|.\|$ is given by the ℓ_1/ℓ_q norm. Then Ω_q encourages $\Theta^1, \Theta^2, \ldots, \Theta^K$ to decompose into V^k and $(V^k)^T$ such that the summed ℓ_q norms of all of the columns (concatenated over $V^1, \ldots, V^K$) is small. This encourages structures of interest on the columns *and* rows of $\Theta^1, \Theta^2, \ldots, \Theta^K$.

To illustrate this point, in Figure 3 we display schematic results obtained from estimating a 5×5 matrix subject to the RCON penalty Ω_q, for $q = 1$, 2, and ∞. We see from Figure 3(b) that when $q = 1$, the RCON penalty yields a matrix estimate with unstructured sparsity; recall that Ω_1 amounts to an ℓ_1 penalty applied to the matrix entries. When $q = 2$ or $q = \infty$, we see from Figures 3(c)-(d) that the RCON penalty yields a sparse matrix estimate for which the non-zero elements are a set of rows *plus* a set of columns—that is, the union of a set of rows and columns.

We note that Ω_2 can be derived from the *overlap norm* (Obozinski et al., 2011; Jacob et al., 2009) applied to groups given by rows and columns of $\Theta^1, \ldots, \Theta^K$. Details are described in Appendix E. Additional properties of RCON are discussed in Appendix A.

3.3 Node-Based Approaches for Learning GGMs

We discuss two approaches for node-based learning of GGMs. The first promotes networks whose differences are attributable to perturbed nodes. The second encourages the networks to share co-hub nodes.

3.3.1 Perturbed-node Joint Graphical Lasso

Consider the task of jointly estimating K precision matrices by solving

$$\underset{\Theta^1, \Theta^2, \ldots, \Theta^K \in \mathbb{S}^p_{++}}{\text{maximize}} \left\{ L(\Theta^1, \Theta^2, \ldots, \Theta^K) - \lambda_1 \sum_{k=1}^K \|\Theta^k\|_1 - \lambda_2 \sum_{k<k'} \Omega_q(\Theta^k - \Theta^{k'}) \right\}. \quad (6)$$

We refer to the convex optimization problem (6) as the *perturbed-node joint graphical lasso* (PNJGL). Let $\hat{\Theta}^1, \hat{\Theta}^2, \ldots, \hat{\Theta}^K$ denote the solution to (6); these serve as estimates for $(\Sigma^1)^{-1}, \ldots, (\Sigma^K)^{-1}$. In (6), λ_1 and λ_2 are nonnegative tuning parameters, and $q \geq 1$. When $\lambda_2 = 0$, (6) amounts simply to applying the graphical lasso optimization problem (1)

to each condition separately in order to separately estimate K networks. When $\lambda_2 > 0$, we are encouraging similarity among the K network estimates. When $q = 1$, we have the following observation.

Remark 2 *The FGL formulation (Equations 2 and 3) is a special case of PNJGL (6) with $q = 1$.*

In other words, when $q = 1$, (6) amounts to the *edge-based* approach of Danaher et al. (2013) that encourages many entries of $\hat{\Theta}^k - \hat{\Theta}^{k'}$ to equal zero.

However, when $q = 2$ or $q = \infty$, then (6) amounts to a *node-based approach*: the support of $\hat{\Theta}^k - \hat{\Theta}^{k'}$ is encouraged to be a union of a few rows and the corresponding columns. These can be interpreted as a set of nodes that are perturbed across the conditions. An example of the sparsity structure detected by PNJGL with $q = 2$ or $q = \infty$ is shown in Figure 2.

3.3.2 Co-hub Node Joint Graphical Lasso

We now consider jointly estimating K precision matrices by solving the convex optimization problem

$$\underset{\Theta^1,\Theta^2,\ldots,\Theta^K \in \mathbb{S}^p_{++}}{\text{maximize}} \left\{ L(\Theta^1, \Theta^2, \ldots, \Theta^K) - \lambda_1 \sum_{k=1}^{K} \|\Theta^k\|_1 - \lambda_2 \Omega_q(\Theta^1 - \text{diag}(\Theta^1), \ldots, \Theta^K - \text{diag}(\Theta^K)) \right\}. \quad (7)$$

We refer to (7) as the *co-hub node joint graphical lasso* (CNJGL) formulation. In (7), λ_1 and λ_2 are nonnegative tuning parameters, and $q \geq 1$. When $\lambda_2 = 0$ then this amounts to a graphical lasso optimization problem applied to each network separately; however, when $\lambda_2 > 0$, a shared structure is encouraged among the K networks. In particular, (7) encourages network estimates that have a common set of hub nodes—that is, it encourages the supports of $\Theta^1, \Theta^2, \ldots, \Theta^K$ to be the same, and the union of a set of rows and columns.

CNJGL can be interpreted as a node-based extension of the GGL proposal (given in Equations 2 and 4, and originally proposed by Danaher et al., 2013). While GGL encourages the K networks to share a common edge support, CNJGL instead encourages the networks to share a common node support.

We now remark on an additional connection between CNJGL and the graphical lasso.

Remark 3 *If $q = 1$, then CNJGL amounts to a modified graphical lasso on each network separately, with a penalty of λ_1 applied to the diagonal elements, and a penalty of $\lambda_1 + \lambda_2/2$ applied to the off-diagonal elements.*

4. Algorithms

The PNJGL and CNJGL optimization problems (6, 7) are convex, and so can be directly solved in the modeling environment `cvx` (Grant and Boyd, 2010), which calls conic interior-point solvers such as `SeDuMi` or `SDPT3`. However, when applied to solve semi-definite programs, second-order methods such as the interior-point algorithm do not scale well with the problem size.

We next examine the use of existing first-order methods to solve (6) and (7). Several first-order algorithms have been proposed for minimizing a least squares objective with

a group lasso penalty (as in Yuan and Lin, 2007b) in the presence of overlapping groups (Argyriou et al., 2011; Chen et al., 2011; Mosci et al., 2010). Unfortunately, those algorithms cannot be applied to the PNJGL and CNJGL formulations, which involve the RCON penalty rather than simply a standard group lasso with overlapping groups. The RCON penalty is a variant of the overlap norm proposed in Obozinski et al. (2011), and indeed those authors propose an algorithm for minimizing a least squares objective subject to the overlap norm. However, in the context of CNJGL and PNJGL, the objective of interest is a Gaussian log likelihood, and the algorithm of Obozinski et al. (2011) cannot be easily applied.

Another possible approach for solving (6) and (7) involves the use of a standard first-order method, such as a projected subgradient approach. Unfortunately, such an approach is not straightforward, since computing the subgradients of the RCON penalty involves solving a non-trivial optimization problem (to be discussed in detail in Appendix A). Similarly, a proximal gradient approach for solving (6) and (7) is challenging because the proximal operator of the combination of the overlap norm and the ℓ_1 norm has no closed form.

To overcome the challenges outlined above, we propose to solve the PNJGL and CNJGL problems using an *alternating direction method of multipliers* algorithm (ADMM; see, e.g., Boyd et al., 2010).

4.1 The ADMM Approach

Here we briefly outline the standard ADMM approach for a general optimization problem,

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & g(X) + h(X) \\ \text{subject to} \quad & X \in \mathcal{X}. \end{aligned} \tag{8}$$

ADMM is attractive in cases where the proximal operator of $g(X) + h(X)$ cannot be easily computed, but where the proximal operator of $g(X)$ and the proximal operator of $h(X)$ are easily obtained. The approach is as follows (Boyd et al., 2010; Eckstein and Bertsekas, 1992; Gabay and Mercier, 1976):

1. Rewrite the optimization problem (8) as

$$\begin{aligned} \underset{X,Y}{\text{minimize}} \quad & g(X) + h(Y) \\ \text{subject to} \quad & X \in \mathcal{X},\ X = Y, \end{aligned} \tag{9}$$

   where here we have decoupled g and h by introducing a new optimization variable, Y.

2. Form the augmented Lagrangian to (9) by first forming the Lagrangian,

$$L(X, Y, \Lambda) = g(X) + h(Y) + \langle \Lambda, X - Y \rangle,$$

   and then augmenting it by a quadratic function of $X - Y$,

$$L_\rho(X, Y, \Lambda) = L(X, Y, \Lambda) + \frac{\rho}{2} \|X - Y\|_F^2,$$

   where ρ is a positive constant.

3. Iterate until convergence:

   (a) Update each primal variable in turn by minimizing the augmented Lagrangian with respect to that variable, while keeping all other variables fixed. The updates in the kth iteration are as follows:

$$\begin{aligned} X^{k+1} &\leftarrow \arg\min_{X \in \mathcal{X}} L_\rho(X, Y^k, \Lambda^k), \\ Y^{k+1} &\leftarrow \arg\min_{Y} L_\rho(X^{k+1}, Y, \Lambda^k). \end{aligned}$$

   (b) Update the dual variable using a dual-ascent update,

$$\Lambda^{k+1} \leftarrow \Lambda^k + \rho(X^{k+1} - Y^{k+1}).$$

The standard ADMM presented here involves minimization over two primal variables, X and Y. For our problems, we will use a similar algorithm but with *more than two primal variables*. More details about the algorithm and its convergence are discussed in Section 4.2.4.

4.2 ADMM Algorithms for PNJGL and CNJGL

Here we outline the ADMM algorithms for the PNJGL and CNJGL optimization problems; we refer the reader to Appendix F for detailed derivations of the update rules.

4.2.1 ADMM Algorithm for PNJGL

Here we consider solving PNJGL with $K = 2$; the extension for $K > 2$ is slightly more complicated. To begin, we note that (6) can be rewritten as

$$\begin{array}{ll} \underset{\Theta^1, \Theta^2 \in \mathbb{S}^p_{++}, V \in \mathbb{R}^{p \times p}}{\text{maximize}} & \left\{ L(\Theta^1, \Theta^2) - \lambda_1 \|\Theta^1\|_1 - \lambda_1 \|\Theta^2\|_1 - \lambda_2 \sum_{j=1}^{p} \|V_j\|_q \right\} \\ \text{subject to} & \Theta^1 - \Theta^2 = V + V^T. \end{array} \tag{10}$$

We now reformulate (10) by introducing new variables, so as to decouple some of the terms in the objective function that are difficult to optimize jointly:

$$\begin{array}{ll} \underset{\Theta^1 \in S^p_{++}, \Theta^2 \in S^p_{++}, Z^1, Z^2, V, W}{\text{minimize}} & \left\{ -L(\Theta^1, \Theta^2) + \lambda_1 \|Z^1\|_1 + \lambda_1 \|Z^2\|_1 + \lambda_2 \sum_{j=1}^{p} \|V_j\|_q \right\} \\ \text{subject to} & \Theta^1 - \Theta^2 = V + W, V = W^T, \Theta^1 = Z^1, \Theta^2 = Z^2. \end{array} \tag{11}$$

The augmented Lagrangian to (11) is given by

$$\begin{aligned} & - L(\Theta^1, \Theta^2) + \lambda_1 \|Z^1\|_1 + \lambda_1 \|Z^2\|_1 + \lambda_2 \sum_{j=1}^{p} \|V_j\|_q + \langle F, \Theta^1 - \Theta^2 - (V + W) \rangle \\ & + \langle G, V - W^T \rangle + \langle Q^1, \Theta^1 - Z^1 \rangle + \langle Q^2, \Theta^2 - Z^2 \rangle + \tfrac{\rho}{2} \|\Theta^1 - \Theta^2 - (V + W)\|_F^2 \\ & + \tfrac{\rho}{2} \|V - W^T\|_F^2 + \tfrac{\rho}{2} \|\Theta^1 - Z^1\|_F^2 + \tfrac{\rho}{2} \|\Theta^2 - Z^2\|_F^2. \end{aligned} \tag{12}$$

In (12) there are six primal variables and four dual variables. Based on this augmented Lagrangian, the complete ADMM algorithm for (6) is given in Algorithm 1, in which the operator Expand is given by

$$\text{Expand}(A,\rho,n_k) = \underset{\Theta\in\mathbb{S}^p_{++}}{\text{argmin}} \left\{-n_k \log\det(\Theta) + \rho\|\Theta - A\|_F^2\right\} = \frac{1}{2}U\left(D + \sqrt{D^2 + \frac{2n_k}{\rho}I}\right)U^T,$$

where UDU^T is the eigenvalue decomposition of a symmetric matrix A, and as mentioned earlier, n_k is the number of observations in the kth class. The operator $\mathcal{T}_q$ is given by

$$\mathcal{T}_q(A,\lambda) = \underset{X}{\text{argmin}} \left\{\frac{1}{2}\|X - A\|_F^2 + \lambda\sum_{j=1}^{p}\|X_j\|_q\right\},$$

and is also known as the proximal operator corresponding to the ℓ_1/ℓ_q norm. For $q = 1, 2, \infty$, $\mathcal{T}_q$ takes a simple form (see, e.g., Section 5 of Duchi and Singer, 2009).

Algorithm 1: ADMM algorithm for the PNJGL optimization problem (6)

input: $\rho > 0, \mu > 1, t_{\max} > 0$;
Initialize: Primal variables to the identity matrix and dual variables to the zero matrix;
for $t = 1{:}t_{\max}$ **do**
  $\rho \leftarrow \mu\rho$;
  while *Not converged* **do**
    $\Theta^1 \leftarrow \text{Expand}\left(\frac{1}{2}(\Theta^2 + V + W + Z^1) - \frac{1}{2\rho}(Q^1 + n_1S^1 + F), \rho, n_1\right)$;
    $\Theta^2 \leftarrow \text{Expand}\left(\frac{1}{2}(\Theta^1 - (V + W) + Z^2) - \frac{1}{2\rho}(Q^2 + n_2S^2 - F), \rho, n_2\right)$;
    $Z^i \leftarrow \mathcal{T}_1\left(\Theta^i + \frac{Q^i}{\rho}, \frac{\lambda_1}{\rho}\right)$ for $i = 1, 2$;
    $V \leftarrow \mathcal{T}_q\left(\frac{1}{2}(W^T - W + (\Theta^1 - \Theta^2)) + \frac{1}{2\rho}(F - G), \frac{\lambda_2}{2\rho}\right)$;
    $W \leftarrow \frac{1}{2}(V^T - V + (\Theta^1 - \Theta^2)) + \frac{1}{2\rho}(F + G^T)$;
    $F \leftarrow F + \rho(\Theta^1 - \Theta^2 - (V + W))$;
    $G \leftarrow G + \rho(V - W^T)$;
    $Q^i \leftarrow Q^i + \rho(\Theta^i - Z^i)$ for $i = 1, 2$

4.2.2 ADMM Algorithm for CNJGL

The CNJGL formulation in (7) is equivalent to

$$\begin{aligned} \underset{\Theta^i\in\mathbb{S}^p_{++}, V^i\in\mathbb{R}^{p\times p}, i=1\ldots K}{\text{minimize}} \quad & -L(\Theta^1,\Theta^2,\ldots,\Theta^K) + \lambda_1\sum_{i=1}^{K}\|\Theta^i\|_1 + \lambda_2\sum_{j=1}^{p}\left\|\begin{bmatrix} V^1 \\ V^2 \\ \vdots \\ V^K \end{bmatrix}_j\right\|_q \\ \text{subject to} \quad & \Theta^i - \text{diag}(\Theta^i) = V^i + (V^i)^T \text{ for } i = 1,\ldots,K. \end{aligned} \tag{13}$$

One can easily see that the problem (13) is equivalent to the problem

$$\begin{array}{ll} \underset{\Theta^i \in \mathbb{S}^p_{++}, \tilde{V}^i \in \mathbb{R}^{p\times p}, i=1\ldots K}{\text{minimize}} & -L(\Theta^1, \Theta^2, \ldots, \Theta^K) + \lambda_1 \sum_{i=1}^K \|\Theta^i\|_1 + \lambda_2 \sum_{j=1}^p \left\| \begin{bmatrix} \tilde{V}^1 - \text{diag}(\tilde{V}^1) \\ \tilde{V}^2 - \text{diag}(\tilde{V}^2) \\ \vdots \\ \tilde{V}^K - \text{diag}(\tilde{V}^K) \end{bmatrix}_j \right\|_q \\ \text{subject to} & \Theta^i = \tilde{V}^i + (\tilde{V}^i)^T \ \text{ for } i = 1, 2, \ldots, K, \end{array} \tag{14}$$

in the sense that the optimal solution $\{V^i\}$ to (13) and the optimal solution $\{\tilde{V}^i\}$ to (14) have the following relationship: $V^i = \tilde{V}^i - \text{diag}(\tilde{V}^i)$ for $i = 1, 2, \ldots, K$. We now present an ADMM algorithm for solving (14). We reformulate (14) by introducing additional variables in order to decouple some terms of the objective that are difficult to optimize jointly:

$$\begin{array}{ll} \underset{\Theta^i \in \mathbb{S}^p_{++}, Z^i, \tilde{V}^i, W^i \in \mathbb{R}^{p\times p}}{\text{minimize}} & -L(\Theta^1, \Theta^2, \ldots, \Theta^K) + \lambda_1 \sum_{i=1}^K \|Z^i\|_1 + \lambda_2 \sum_{j=1}^p \left\| \begin{bmatrix} \tilde{V}^1 - \text{diag}(\tilde{V}^1) \\ \tilde{V}^2 - \text{diag}(\tilde{V}^2) \\ \vdots \\ \tilde{V}^K - \text{diag}(\tilde{V}^K) \end{bmatrix}_j \right\|_q \\ \text{subject to} & \Theta^i = \tilde{V}^i + W^i, \tilde{V}^i = (W^i)^T, \Theta^i = Z^i \text{ for } i = 1, 2, \ldots, K. \end{array} \tag{15}$$

The augmented Lagrangian to (15) is given by

$$\begin{aligned} &\sum_{i=1}^K n_i(-\log\det(\Theta^i) + \text{trace}(S^i\Theta^i)) + \lambda_1 \sum_{i=1}^K \|Z^i\|_1 + \lambda_2 \sum_{j=1}^p \left\| \begin{bmatrix} \tilde{V}^1 - \text{diag}(\tilde{V}^1) \\ \tilde{V}^2 - \text{diag}(\tilde{V}^2) \\ \vdots \\ \tilde{V}^K - \text{diag}(\tilde{V}^K) \end{bmatrix}_j \right\|_q \quad + \\ &\sum_{i=1}^K \left\{ \langle F^i, \Theta^i - (\tilde{V}^i + W^i)\rangle + \langle G^i, \tilde{V}^i - (W^i)^T\rangle + \langle Q^i, \Theta^i - Z^i\rangle \right\} \quad + \\ &\frac{\rho}{2} \sum_{i=1}^K \left\{ \|\Theta^i - (\tilde{V}^i + W^i)\|_F^2 + \|\tilde{V}^i - (W^i)^T\|_F^2 + \|\Theta^i - Z^i\|_F^2 \right\}. \end{aligned} \tag{16}$$

The corresponding ADMM algorithm is given in Algorithm 2.

Algorithm 2: ADMM algorithm for the CNJGL optimization problem (7)

input: $\rho > 0, \mu > 1, t_{\max} > 0$;
Initialize: Primal variables to the identity matrix and dual variables to the zero matrix;
for $t = 1{:}t_{\max}$ **do**
  $\rho \leftarrow \mu\rho$;
  while *Not converged* **do**
    $\Theta^i \leftarrow \text{Expand}\left(\frac{1}{2}(\tilde{V}^i + W^i + Z^i) - \frac{1}{2\rho}(Q^i + n_i S^i + F^i), \rho, n_i\right)$ for $i = 1, \ldots, K$;
    $Z^i \leftarrow \mathcal{T}_1\left(\Theta^i + \frac{Q^i}{\rho}, \frac{\lambda_1}{\rho}\right)$ for $i = 1, \ldots, K$;
    Let $C^i = \frac{1}{2}((W^i)^T - W^i + \Theta^i) + \frac{1}{2\rho}(F^i - G^i)$ for $i = 1, \ldots, K$;
    $$\begin{bmatrix} \tilde{V}^1 \\ \tilde{V}^2 \\ \vdots \\ \tilde{V}^K \end{bmatrix} \leftarrow \mathcal{T}_q\left(\begin{bmatrix} C^1 - \text{diag}(C^1) \\ C^2 - \text{diag}(C^2) \\ \vdots \\ C^K - \text{diag}(C^K) \end{bmatrix}, \frac{\lambda_2}{2\rho}\right) + \begin{bmatrix} \text{diag}(C^1) \\ \text{diag}(C^2) \\ \vdots \\ \text{diag}(C^K) \end{bmatrix};$$
    $W^i \leftarrow \frac{1}{2}((\tilde{V}^i)^T - \tilde{V}^i + \Theta^i) + \frac{1}{2\rho}(F^i + (G^i)^T)$ for $i = 1, \ldots, K$;
    $F^i \leftarrow F^i + \rho(\Theta^i - (\tilde{V}^i + W^i))$ for $i = 1, \ldots, K$;
    $G^i \leftarrow G^i + \rho(\tilde{V}^i - (W^i)^T)$ for $i = 1, \ldots, K$;
    $Q^i \leftarrow Q^i + \rho(\Theta^i - Z^i)$ for $i = 1, \ldots, K$

4.2.3 Numerical Issues and Run-Time of the ADMM Algorithms

We set $\mu = 5$, $\rho = 0.5$ and $t_{\max} = 1000$ in the PNJGL and CNJGL algorithms. In our implementation of these algorithms, the stopping criterion for the inner loop (corresponding to a fixed ρ) is

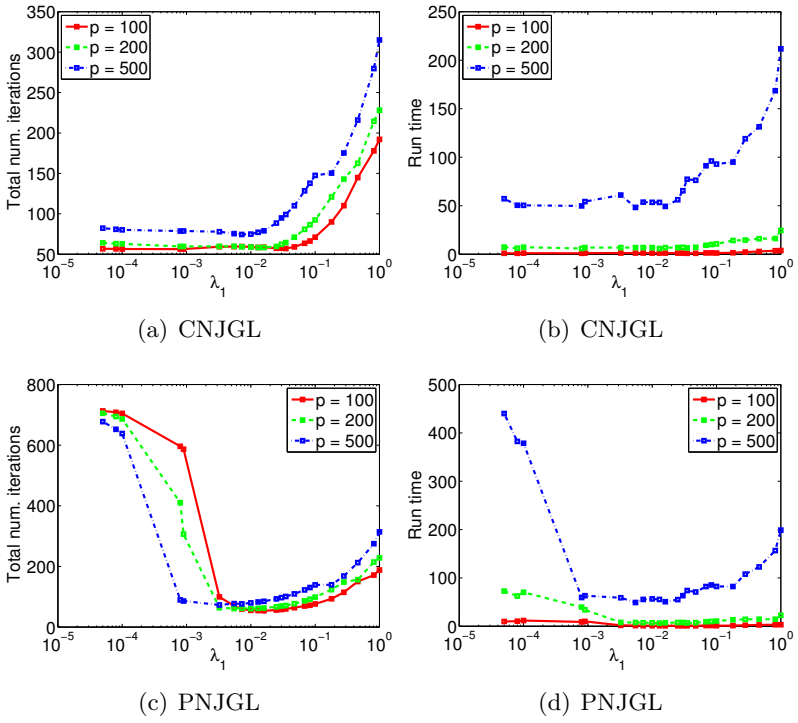
$$\max_{i \in \{1,2,\ldots,K\}} \left\{ \frac{\|(\Theta^i)^{(k+1)} - (\Theta^i)^{(k)}\|_F}{\|(\Theta^i)^{(k)}\|_F} \right\} \leq \epsilon,$$

where $(\Theta^i)^{(k)}$ denotes the estimate of Θ^i in the kth iteration of the ADMM algorithm, and ϵ is a tolerance that is chosen in our experiments to equal 10^{-4}.

The per-iteration complexity of the ADMM algorithms for CNJGL and PNJGL (with $K = 2$) is $O(p^3)$; this is the complexity of computing the SVD. On the other hand, the complexity of a general interior point method is $O(p^6)$. In a small example with $p = 30$, run on an Intel Xeon X3430 2.4Ghz CPU, the interior point method (using `cvx`, which calls `Sedumi`) takes 7 minutes to run, while the ADMM algorithm for PNJGL, coded in `Matlab`, takes only 0.58 seconds. When $p = 50$, the times are 3.5 hours and 2.1 seconds, respectively. Let $\hat{\Theta}^1, \hat{\Theta}^2$ and $\bar{\Theta}^1, \bar{\Theta}^2$ denote the solutions obtained by ADMM and `cvx`, respectively. We observe that on average, the error $\max_{i \in \{1,2\}} \left\{ \|\hat{\Theta}^i - \bar{\Theta}^i\|_F / \|\bar{\Theta}^i\|_F \right\}$ is on the order of 10^{-4}. Thus, the algorithm has good empirical accuracy in recovering the optimal solution.

We now present a more extensive runtime study for the ADMM algorithms for PNJGL and CNJGL. We ran experiments with $p = 100, 200, 500$ and with $n_1 = n_2 = p/2$. We generated synthetic data as described in Section 6. Results are displayed in Figures 4(a)-(d), where the panels depict the run-time and number of iterations required for the algorithm to terminate, as a function of λ_1, and with λ_2 fixed. The number of iterations required for the algorithm to terminate is computed as the total number of inner loop iterations performed in Algorithms 1 and 2. From Figures 4(b) and (d), we observe that as p increases from 100 to 500, the run-times increase substantially, but never exceed several minutes.

Figure 4(a) indicates that for CNJGL, the total number of iterations required for algorithm termination is small when λ_1 is small. In contrast, for PNJGL, Figure 4(c) indicates that the total number of iterations is large when λ_1 is small. This phenomenon results from the use of the identity matrix to initialize the network estimates in the ADMM algorithms: when λ_1 is small, the identity is a poor initialization for PNJGL, but a good initialization for CNJGL (since for CNJGL, λ_2 induces sparsity even when $\lambda_1 = 0$).



(a) CNJGL (b) CNJGL

(c) PNJGL (d) PNJGL

Figure 4: *(a):* The total number of iterations for the CNJGL algorithm, as a function of λ_1. *(b):* Run-time (in seconds) of the CNJGL algorithm, as a function of λ_1. *(c)-(d):* As in (a)-(b), but for the PNJGL algorithm. All results are averaged over 20 random generations of synthetic data.

4.2.4 Convergence of the ADMM Algorithm

Problem (9) involves two (groups of) primal variables, X and Y; in this setting, convergence of ADMM has been established (see, e.g., Boyd et al., 2010; Mota et al., 2011). However, the PNJGL and CNJGL optimization problems involve more than two groups of primal variables, and convergence of ADMM in this setting is an ongoing area of research. Indeed, as mentioned in Eckstein (2012), the standard analysis for ADMM with two groups does not extend in a straightforward way to ADMM with more than two groups of variables. Han and Yuan (2012) and Hong and Luo (2012) show convergence of ADMM with more than two groups of variables under assumptions that do not hold for CNJGL and PNJGL. Under very minimal assumptions, He et al. (2012) proved that a modified ADMM algorithm (with Gauss-Seidel updates) converges to the optimal solution for problems with any number of groups. More general conditions for convergence of the ADMM algorithm with more than two groups is left as a topic for future work. We also leave for future work a reformulation of the CNJGL and PNJGL problems as consensus problems, for which an ADMM algorithm involving two groups of primal variables can be obtained, and for which convergence would be guaranteed. Finally, note that despite the lack of convergence theory, ADMM with more than two groups has been used in practice and often observed to converge faster than other variants. As an example see Tao and Yuan (2011), where their ASALM algorithm (which is the same as ADMM with more than two groups) is reported to be significantly faster than a variant with theoretical convergence.

5. Algorithm-Independent Computational Speed-Ups

The ADMM algorithms presented in the previous section work well on problems of moderate size. In order to solve the PNJGL or CNJGL optimization problems when the number of variables is large, a faster approach is needed. We now describe conditions under which any algorithm for solving the PNJGL or CNJGL problems can be sped up substantially, for an appropriate range of tuning parameter values. Our approach mirrors previous results for the graphical lasso (Witten et al., 2011; Mazumder and Hastie, 2012), and FGL and GGL (Danaher et al., 2013). The idea is simple: if the solutions to the PNJGL or CNJGL optimization problem are block-diagonal (up to some permutation of the variables) with shared support, then we can obtain the global solution to the PNJGL or CNJGL optimization problem by solving the PNJGL or CNJGL problem separately on the variables within each block. This can lead to massive speed-ups. For instance, if the solutions are block-diagonal with L blocks of equal size, then the complexity of our ADMM algorithm reduces from $O(p^3)$ per iteration, to $O((p/L)^3)$ per iteration in each of L independent subproblems. Of course, this hinges upon knowing that the PNJGL or CNJGL solutions are block-diagonal, and knowing the partition of the variables into blocks.

In Sections 5.1-5.3 we derive necessary and sufficient conditions for the solutions to the PNJGL and CNJGL problems to be block-diagonal. Our conditions depend only on the sample covariance matrices $S^1, \ldots, S^k$ and regularization parameters λ_1, λ_2. These conditions can be applied in at most $O(p^2)$ operations. In Section 5.4, we demonstrate the speed-ups that can result from applying these sufficient conditions.

Related results for the graphical lasso (Witten et al., 2011; Mazumder and Hastie, 2012) and FGL and GGL (Danaher et al., 2013) involve a single condition that is both necessary

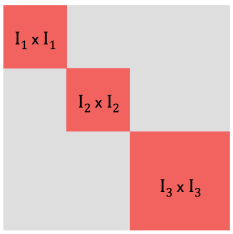


Figure 5: A $p \times p$ matrix is displayed, for which I_1, I_2, I_3 denote a partition of the index set $\{1, 2, \ldots, p\}$. $T = \bigcup_{i=1}^{L} \{I_i \times I_i\}$ is shown in red, and T^c is shown in gray.

and sufficient for the solution to be block diagonal. In contrast, in the results derived below, there is a gap between the necessary and sufficient conditions. Though only the sufficient conditions are required in order to obtain the computational speed-ups discussed in Section 5.4, knowing the necessary conditions allows us to get a handle on the tightness (and, consequently, the practical utility) of the sufficient conditions, for a particular value of the tuning parameters.

We now introduce some notation that will be used throughout this section. Let $(I_1, I_2, \ldots, I_L)$ be a partition of the index set $\{1, 2, \ldots, p\}$, and let $T = \bigcup_{i=1}^{L} \{I_i \times I_i\}$. Define the *support* of a matrix Θ, denoted by $\text{supp}(\Theta)$, as the set of indices of the non-zero entries in Θ. We say Θ is supported on T if $\text{supp}(\Theta) \subseteq T$. Note that any matrix supported on T is block-diagonal subject to some permutation of its rows and columns. Let $|T|$ denote the cardinality of the set T, and let T^c denote the complement of T. The scheme is displayed in Figure 5. In what follows we use an ℓ_1/ℓ_q norm in the RCON penalty, with $q \geq 1$, and let $\frac{1}{s} + \frac{1}{q} = 1$.

5.1 Conditions for PNJGL Formulation to Have Block-Diagonal Solutions

In this section, we give necessary conditions and sufficient conditions on the regularization parameters λ_1, λ_2 in the PNJGL problem (6) so that the resulting precision matrix estimates $\hat{\Theta}^1, \ldots, \hat{\Theta}^K$ have a shared block-diagonal structure (up to a permutation of the variables).

We first present a necessary condition for $\hat{\Theta}^1$ and $\hat{\Theta}^2$ that minimize (6) with $K = 2$ to be block-diagonal.

Theorem 4 *Suppose that the matrices $\hat{\Theta}^1$ and $\hat{\Theta}^2$ that minimize (6) with $K = 2$ have support T. Then, if $q \geq 1$, it must hold that*

$$n_k |S^k_{ij}| \leq \lambda_1 + \lambda_2/2 \qquad \forall (i,j) \in T^c, \quad \text{for } k = 1, 2, \text{ and} \tag{17}$$

$$|n_1 S^1_{ij} + n_2 S^2_{ij}| \leq 2\lambda_1 \qquad \forall (i,j) \in T^c. \tag{18}$$

Furthermore, if $q > 1$, then it must additionally hold that

$$\frac{n_k}{|T^c|} \sum_{(i,j) \in T^c} |S^k_{ij}| \leq \lambda_1 + \frac{\lambda_2}{2} \left(\frac{p}{|T^c|} \right)^{1/s}, \quad \text{for } k = 1, 2. \tag{19}$$

Remark 5 *If $|T^c| = O(p^r)$ with $r > 1$, then as $p \to \infty$, (19) simplifies to $\frac{n_k}{|T^c|} \sum_{(i,j) \in T^c} |S^k_{ij}| \leq \lambda_1$.*

We now present a sufficient condition for $\hat{\Theta}^1, \ldots, \hat{\Theta}^K$ that minimize (6) to be block-diagonal.

Theorem 6 *For $q \geq 1$, a sufficient condition for the matrices $\hat{\Theta}^1, \ldots, \hat{\Theta}^K$ that minimize (6) to each have support T is that*

$$n_k |S_{ij}^k| \leq \lambda_1 \quad \forall (i,j) \in T^c, \quad \text{for } k = 1, \ldots, K.$$

Furthermore, if $q = 1$ and $K = 2$, then the necessary conditions (17) and (18) are also sufficient.

When $q = 1$ and $K = 2$, then the necessary and sufficient conditions in Theorems 4 and 6 are identical, as was previously reported in Danaher et al. (2013). In contrast, there is a gap between the necessary and sufficient conditions in Theorems 4 and 6 when $q > 1$ and $\lambda_2 > 0$. When $\lambda_2 = 0$, the necessary and sufficient conditions in Theorems 4 and 6 reduce to the results laid out in Witten et al. (2011) for the graphical lasso.

5.2 Conditions for CNJGL Formulation to Have Block-Diagonal Solutions

In this section, we give necessary and sufficient conditions on the regularization parameters λ_1, λ_2 in the CNJGL optimization problem (7) so that the resulting precision matrix estimates $\hat{\Theta}^1, \ldots, \hat{\Theta}^K$ have a shared block-diagonal structure (up to a permutation of the variables).

Theorem 7 *Suppose that the matrices $\hat{\Theta}^1, \hat{\Theta}^2, \ldots, \hat{\Theta}^K$ that minimize (7) have support T. Then, if $q \geq 1$, it must hold that*

$$n_k |S_{ij}^k| \leq \lambda_1 + \lambda_2/2 \quad \forall (i,j) \in T^c, \quad \text{for } k = 1, \ldots, K.$$

Furthermore, if $q > 1$, then it must additionally hold that

$$\frac{n_k}{|T^c|} \sum_{(i,j) \in T^c} |S_{ij}^k| \leq \lambda_1 + \frac{\lambda_2}{2} \left(\frac{p}{|T^c|} \right)^{1/s}, \quad \text{for } k = 1, \ldots, K. \tag{20}$$

Remark 8 *If $|T^c| = O(p^r)$ with $r > 1$, then as $p \to \infty$, (20) simplifies to $\frac{n_k}{|T^c|} \sum_{(i,j) \in T^c} |S_{ij}^k| \leq \lambda_1$.*

We now present a sufficient condition for $\hat{\Theta}^1, \hat{\Theta}^2, \ldots, \hat{\Theta}^K$ that minimize (7) to be block-diagonal.

Theorem 9 *A sufficient condition for $\hat{\Theta}^1, \hat{\Theta}^2, \ldots, \hat{\Theta}^K$ that minimize (7) to have support T is that*

$$n_k |S_{ij}^k| \leq \lambda_1 \quad \forall (i,j) \in T^c, \quad \text{for } k = 1, \ldots, K.$$

As was the case for the PNJGL formulation, there is a gap between the necessary and sufficient conditions for the estimated precision matrices from the CNJGL formulation to have a common block-diagonal support.

5.3 General Sufficient Conditions

In this section, we give sufficient conditions for the solution to a general class of optimization problems that include FGL, PNJGL, and CNJGL as special cases to be block-diagonal. Consider the optimization problem

$$\underset{\Theta^1,\ldots,\Theta^K\in\mathbb{S}^p_{++}}{\text{minimize}} \left\{ \sum_{k=1}^K n_k(-\log\det(\Theta^k)+\langle\Theta^k,S^k\rangle) + \sum_{k=1}^K \lambda_1\|\Theta^k\|_1 + \lambda_2 h(\Theta^1,\ldots,\Theta^K) \right\}. \tag{21}$$

Once again, let T be the support of a $p\times p$ block-diagonal matrix. Let Θ_T denote the restriction of any $p\times p$ matrix Θ to T; that is, $(\Theta_T)_{ij}=\begin{cases}\Theta_{ij} & \text{if } (i,j)\in T\\ 0 & \text{else}\end{cases}$. Assume that the function h satisfies

$$h(\Theta^1,\ldots,\Theta^K) > h(\Theta^1_U,\ldots,\Theta^K_U)$$

for any matrices $\Theta^1,\ldots,\Theta^K$ whose support strictly contains U.

Theorem 10 *A sufficient condition for the matrices $\hat\Theta^1,\ldots,\hat\Theta^K$ that solve (21) to have support T is that*

$$n_k|S^k_{ij}|\le\lambda_1 \quad \forall(i,j)\in T^c, \quad \text{for } k=1,\ldots,K.$$

Note that this sufficient condition applies to a broad class of regularizers h; indeed, the sufficient conditions for PNJGL and CNJGL given in Theorems 6 and 9 are special cases of Theorem 10. In contrast, the necessary conditions for PNJGL and CNJGL in Theorems 4 and 7 exploit the specific structure of the RCON penalty.

5.4 Evaluation of Speed-Ups on Synthetic Data

Theorems 6 and 9 provide sufficient conditions for the precision matrix estimates from PNJGL or CNJGL to be block-diagonal with a given support. How can these be used in order to obtain computational speed-ups? We construct a $p\times p$ matrix A with elements

$$A_{ij}=\begin{cases}1 & \text{if } i=j\\ 1 & \text{if } n_k|S^k_{ij}|>\lambda_1 \text{ for any } k=1,\ldots,K\\ 0 & \text{else}\end{cases}.$$

We can then check, in $O(p^2)$ operations, whether A is (subject to some permutation of the rows and columns) block-diagonal, and can also determine the partition of the rows and columns corresponding to the blocks (see, e.g., Tarjan, 1972). Then, by Theorems 6 and 9, we can conclude that the PNJGL or CNJGL estimates are block-diagonal, with the same partition of the variables into blocks. Inspection of the PNJGL and CNJGL optimization problems reveals that we can then solve the problems on the variables within each block separately, in order to obtain the global solution to the original PNJGL or CNJGL optimization problems.

We now investigate the speed-ups that result from applying this approach. We consider the problem of estimating two networks of size $p=500$. We create two inverse covariance matrices that are block diagonal with two equally-sized blocks, and sparse within each

block. We then generate $n_1 = 250$ observations from a multivariate normal distribution with the first covariance matrix, and $n_2 = 250$ observations from a multivariate normal distribution with the second covariance matrix. These observations are used to generate sample covariance matrices S^1 and S^2. We then performed CNJGL and PNJGL with $\lambda_2 = 1$ and a range of λ_1 values, with and without the computational speed-ups just described.

Figure 6 displays the performance of the CNJGL and PNJGL formulations, averaged over 20 data sets generated in this way. In each panel, the x-axis shows the number of blocks into which the optimization problems were decomposed using the sufficient conditions; note that this is a surrogate for the value of λ_1 in the CNJGL or PNJGL optimization problems. Figure 6(a) displays the ratio of the run-time taken by the ADMM algorithm when exploiting the sufficient conditions to the run-time when not using the sufficient conditions. Figure 6(b) displays the true-positive ratio—that is, the ratio of the number of true positive edges in the precision matrix estimates to the total number of edges in the precision matrix estimates. Figure 6(c) displays the total number of true positives for the CNJGL and PNJGL estimates. Figure 6 indicates that the sufficient conditions detailed in this section lead to substantial computational improvements.

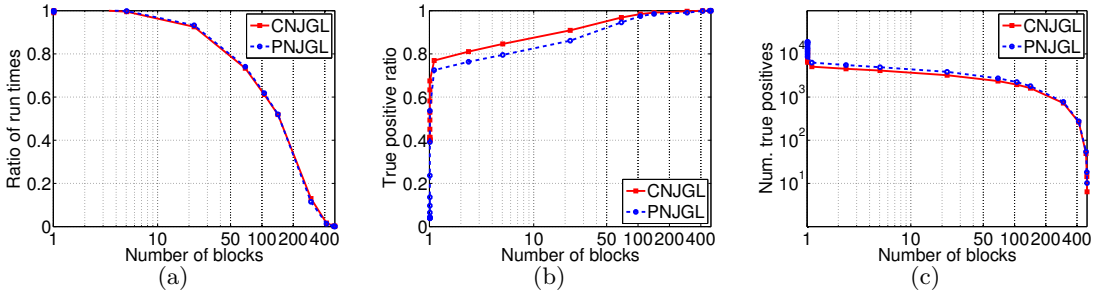


Figure 6: Speed-ups for CNJGL and PNJGL on a simulation set-up with $p = 500$ and $n_1 = n_2 = 250$. The true inverse covariance matrices are block-diagonal with two equally-sized sparse blocks. The x-axis in each panel displays the number of blocks into which the CNJGL or PNJGL problems are decomposed using the sufficient conditions; this is a surrogate for λ_1. The y-axes display *(a)*: the ratio of run-times with and without the sufficient conditions; *(b)*: the true positive ratio of the edges estimated; and *(c)*: the total number of true positive edges estimated.

6. Simulation Study

In this section, we present the results of a simulation study demonstrating the empirical performance of PNJGL and CNJGL.

6.1 Data Generation

In the simulation study, we generated two synthetic networks (either Erdos-Renyi, scale-free, or community), each of which contains a common set of p nodes. Four of the p nodes

were then modified in order to create two perturbed nodes and two co-hub nodes. Details are provided in Sections 6.1.1-6.1.3.

6.1.1 Data Generation for Erdos-Renyi Network

We generated the data as follows, for $p = 100$, and $n \in \{25, 50, 100, 200\}$:

Step 1: To generate an Erdos-Renyi network, we created a $p \times p$ symmetric matrix A with elements

$$A_{ij} \sim_{\text{i.i.d.}} \begin{cases} 0 & \text{with probability } 0.98, \\ \text{Unif}([-0.6, -0.3] \cup [0.3, 0.6]) & \text{otherwise.} \end{cases}$$

Step 2: We duplicated A into two matrices, A^1 and A^2. We selected two nodes at random, and for each node, we set the elements of the corresponding row and column of either A^1 or A^2 (chosen at random) to be i.i.d. draws from a Unif$([-0.6, -0.3] \cup [0.3, 0.6])$ distribution. This results in two perturbed nodes.

Step 3: We randomly selected two nodes to serve as co-hub nodes, and set each element of the corresponding rows and columns in each network to be i.i.d. draws from a Unif$([-0.6, -0.3] \cup [0.3, 0.6])$ distribution. In other words, these co-hub nodes are *identical* across the two networks.

Step 4: In order to make the matrices positive definite, we let $c = \min(\lambda_{\min}(A^1), \lambda_{\min}(A^2))$, where $\lambda_{\min}(\cdot)$ indicates the smallest eigenvalue of the matrix. We then set $(\Sigma^1)^{-1}$ equal to $A^1 + (0.1 + |c|)I$ and set $(\Sigma^2)^{-1}$ equal to $A^2 + (0.1 + |c|)I$, where I is the $p \times p$ identity matrix.

Step 5: We generated n independent observations each from a $N(0, \Sigma^1)$ and a $N(0, \Sigma^2)$ distribution, and used them to compute the sample covariance matrices S^1 and S^2.

6.1.2 Data Generation for Scale-free Network

The data generation proceeded as in Section 6.1.1, except that Step 1 was modified:

Step 1: We used the `SFNG` functions in `Matlab` (George, 2007) with parameters `mlinks=2` and `seed=1` to generate a scale-free network with p nodes. We then created a $p \times p$ symmetric matrix A that has non-zero elements only for the edges in the scale-free network. These non-zero elements were generated i.i.d. from a Unif$([-0.6, -0.3] \cup [0.3, 0.6])$ distribution.

Steps 2-5 proceeded as in Section 6.1.1.

6.1.3 Data Generation for Community Network

We generated data as in Section 6.1.1, except for one modification: at the end of Step 3, we set the [1:40, 61:100] and [61:100, 1:40] submatrices of A^1 and A^2 equal to zero.

Then A^1 and A^2 have non-zero entries concentrated in the top and bottom 60×60 principal submatrices. These two submatrices correspond to two communities. Twenty nodes overlap between the two communities.

6.2 Results

We now define several metrics used to measure algorithm performance. We wish to quantify each algorithm's (1) recovery of the support of the true inverse covariance matrices, (2) successful detection of co-hub and perturbed nodes, and (3) error in estimation of $\Theta^1 = (\Sigma^1)^{-1}$ and $\Theta^2 = (\Sigma_2)^{-1}$. Details are given in Table 1. These metrics are discussed further in Appendix G.

We compared the performance of PNJGL to its edge-based counterpart FGL, as well as to graphical lasso (GL). We compared the performance of CNJGL to GGL and GL. We expect CNJGL to be able to detect co-hub nodes (and, to a lesser extent, perturbed nodes), and we expect PNJGL to be able to detect perturbed nodes. (The co-hub nodes will not be detected by PNJGL, since they are identical across the networks.)

The simulation results for the set-up of Section 6.1.1 are displayed in Figures 7 and 8. Each row corresponds to a sample size while each column corresponds to a performance metric. In Figure 7, PNJGL, FGL, and GL are compared, and in Figure 8, CNJGL, GGL, and GL are compared. Within each plot, each colored line corresponds to the results obtained using a fixed value of λ_2 (for either PNJGL, FGL, CNJGL, or GGL), as λ_1 is varied. Recall that GL corresponds to any of these four approaches with $\lambda_2 = 0$. Note that the number of positive edges (defined in Table 1) decreases approximately monotonically with the regularization parameter λ_1, and so on the x-axis we plot the number of positive edges, rather than λ_1, for ease of interpretation.

In Figure 7, we observe that PNJGL outperforms FGL and GL for a suitable range of the regularization parameter λ_2, in the sense that for a fixed number of edges estimated, PNJGL identifies more true positives, correctly identifies a greater ratio of perturbed nodes, and yields a lower Frobenius error in the estimates of Θ^1 and Θ^2. In particular, PNJGL performs best relative to FGL and GL when the number of samples is the smallest, that is, in the high-dimensional data setting. Unlike FGL, PNJGL fully exploits the fact that differences between Θ^1 and Θ^2 are due to node perturbation. Not surprisingly, GL performs worst among the three algorithms, since it does not borrow strength across the conditions in estimating Θ^1 and Θ^2.

In Figure 8, we note that CNJGL outperforms GGL and GL for a suitable range of the regularization parameter λ_2. In particular, CNJGL outperforms GGL and GL by a larger margin when the number of samples is the smallest. Once again, GL performs the worst since it does not borrow strength across the two networks; CNJGL performs the best since it fully exploits the presence of hub nodes in the data.

We note one interesting feature of Figure 8: the colored lines corresponding to CNJGL with very large values of λ_2 do not extend beyond around 400 positive edges. This is because for CNJGL, a large value of λ_2 induces sparsity in the network estimates, even if λ_1 is small or zero. Consequently, it is not possible to obtain a dense estimate of Θ^1 and Θ^2 if CNJGL is performed with a large value of λ_2. In contrast, in the case of PNJGL, sparsity is induced only by λ_1, and not at all by λ_2. We note that a similar situation occurs for the edge-based

(1)	Positive edges: $\sum_{i<j}\Big(1\{|\hat{\Theta}^1_{ij}|>t_0\}+1\{|\hat{\Theta}^2_{ij}|>t_0\}\Big)$ True positive edges: $\sum_{i<j}\Big(1\{|\Theta^1_{ij}|>t_0 \text{ and } |\hat{\Theta}^1_{ij}|>t_0\}+1\{|\Theta^2_{ij}|>t_0 \text{ and } |\hat{\Theta}^2_{ij}|>t_0\}\Big)$
(2)	Positive perturbed columns (PPC): PNJGL: $\sum_{i=1}^p 1\Big\{\|\hat{V}_{-i,i}\|_2>t_s\Big\}$; FGL/GL: $\sum_{i=1}^p 1\{\|(\hat{\Theta}^1-\hat{\Theta}^2)_{-i,i}\|_2>t_s\}$ True positive perturbed columns (TPPC): PNJGL: $\sum_{i\in I_p} 1\{\|\hat{V}_{-i,i}\|_2>t_s\}$; FGL/GL: $\sum_{i\in I_p} 1\{\|(\hat{\Theta}^1-\hat{\Theta}^2)_{-i,i}\|_2>t_s\}$, where I_P is the set of perturbed column indices. Positive co-hub columns (PCC): CNJGL: $\sum_{i=1}^p 1\Big\{\|\hat{V}^1_{-i,i}\|_2>t_s \text{and} \|\hat{V}^2_{-i,i}\|_2>t_s\Big\}$; GGL/GL: $\sum_{i=1}^p 1\Big\{\|\hat{\Theta}^1_{-i,i}\|_2>t_s \text{and} \|\hat{\Theta}^2_{-i,i}\|_2>t_s\Big\}$ True positive co-hub columns (TPCC): CNJGL: $\sum_{i\in I_c} 1\Big\{\|\hat{V}^1_{-i,i}\|_2>t_s \text{and} \|\hat{V}^2_{-i,i}\|_2>t_s\Big\}$; GGL/GL: $\sum_{i\in I_c} 1\Big\{\|\hat{\Theta}^1_{-i,i}\|_2>t_s \text{and} \|\hat{\Theta}^2_{-i,i}\|_2>t_s\Big\}$, where I_C is the set of co-hub column indices.
(3)	Error: $\sqrt{\sum_{i<j}(\Theta^1_{ij}-\hat{\Theta}^1_{ij})^2}+\sqrt{\sum_{i<j}(\Theta^2_{ij}-\hat{\Theta}^2_{ij})^2}$

Table 1: Metrics used to quantify algorithm performance. Here Θ^1 and Θ^2 denote the true inverse covariance matrices, and $\hat{\Theta}^1$ and $\hat{\Theta}^2$ denote the two estimated inverse covariance matrices. Here $1\{A\}$ is an indicator variable that equals one if the event A holds, and equals zero otherwise. (1) Metrics based on recovery of the support of Θ^1 and Θ^2. Here $t_0 = 10^{-6}$. (2) Metrics based on identification of perturbed nodes and co-hub nodes. The metrics PPC and TPPC quantify node perturbation, and are applied to PNJGL, FGL, and GL. The metrics PCC and TPCC relate to co-hub detection, and are applied to CNJGL, GGL, and GL. We let $t_s = \mu + 5.5\sigma$, where μ is the mean and σ is the standard deviation of $\{\|\hat{V}_{-i,i}\|_2\}_{i=1}^p$ (PPC or TPPC for PNJGL), $\{\|(\hat{\Theta}^1-\hat{\Theta}^2)_{-i,i}\|_2\}_{i=1}^p$ (PPC or TPPC for FGL/GL), $\{\|\hat{V}^1_{-i,i}\|_2\}_{i=1}^p$ and $\{\|\hat{V}^2_{-i,i}\|_2\}_{i=1}^p$ (PPC or TPPC for CNJGL), or $\{\|\hat{\Theta}^1_{-i,i}\|_2\}_{i=1}^p$ and $\{\|\hat{\Theta}^2_{-i,i}\|_2\}_{i=1}^p$ (PPC or TPPC for GGL/GL). However, results are very insensitive to the value of t_s, as is shown in Appendix G. (3) Frobenius error of estimation of Θ^1 and Θ^2.

counterparts of CNJGL and PNJGL: when GGL is performed with a large value of λ_2 then the network estimates are necessarily sparse, regardless of the value of λ_1. But the same is not true for FGL.

The simulation results for the set-ups of Sections 6.1.2 and 6.1.3 are displayed in Figures 9 and 10, respectively, for the case $n = 50$. The results show that once again, PNJGL and CNJGL substantially outperform the edge-based approaches on the three metrics defined earlier.

7. Real Data Analysis

In this section, we present the results of PNJGL and CNJGL applied to two real data sets: gene expression data set and university webpage data set.

7.1 Gene Expression Data

In this experiment, we aim to reconstruct the gene regulatory networks of two subtypes of glioblastoma multiforme (GBM), as well as to identify genes that can improve our understanding of the disease. Cancer is caused by somatic (cancer-specific) mutations in the genes involved in various cellular processes including cell cycle, cell growth, and DNA repair; such mutations can lead to uncontrolled cell growth. We will show that PNJGL and CNJGL can be used to identify genes that play central roles in the development and progression of cancer. PNJGL tries to identify genes whose interactions with other genes vary significantly between the subtypes. Such genes are likely to have deleterious somatic mutations. CNJGL tries to identify genes that have interactions with many other genes in all subtypes. Such genes are likely to play an important role in controlling other genes' expression, and are typically called *regulators*.

We applied the proposed methods to a publicly available gene expression data set that measures mRNA expression levels of 11,861 genes in 220 tissue samples from patients with GBM (Verhaak et al., 2010). The raw gene expression data were generated using the Affymetrix GeneChips technology. We downloaded the raw data in .CEL format from the The Caner Genome Atlas (TCGA) website. The raw data were normalized by using the Affymetrix `MAS5` algorithm, which has been shown to perform well in many studies (Lim et al., 2007). The data were then log2 transformed and batch-effected corrected using the software `ComBat` (Johnson and Li, 2006). Each patient has one of four subtypes of GBM—Proneural, Neural, Classical, or Mesenchymal. We selected two subtypes, Proneural (53 tissue samples) and Mesenchymal (56 tissue samples), that have the largest sample sizes. All analyses were restricted to the corresponding set of 109 tissue samples.

To evaluate PNJGL's ability to identify genes with somatic mutations, we focused on the following 10 genes that have been suggested to be frequently mutated across the four GBM subtypes (Verhaak et al., 2010): TP53, PTEN, NF1, EGFR, IDH1, PIK3R1, RB1, ERBB2, PIK3CA, PDGFRA. We then considered inferring the regulatory network of a set of genes that is known to be involved in a single biological process, based on the Reactome database (Matthews et al., 2008). In particular, we focused our analysis on the "TCR signaling" gene set, which contains the largest number of mutated genes. This gene set contains 34 genes, of which three (PTEN, PIK3R1, and PIK3CA) are in the list of 10 genes suggested to be mutated in GBM. We applied PNJGL with $q = 2$ to the resulting 53×34 and 56×34 gene

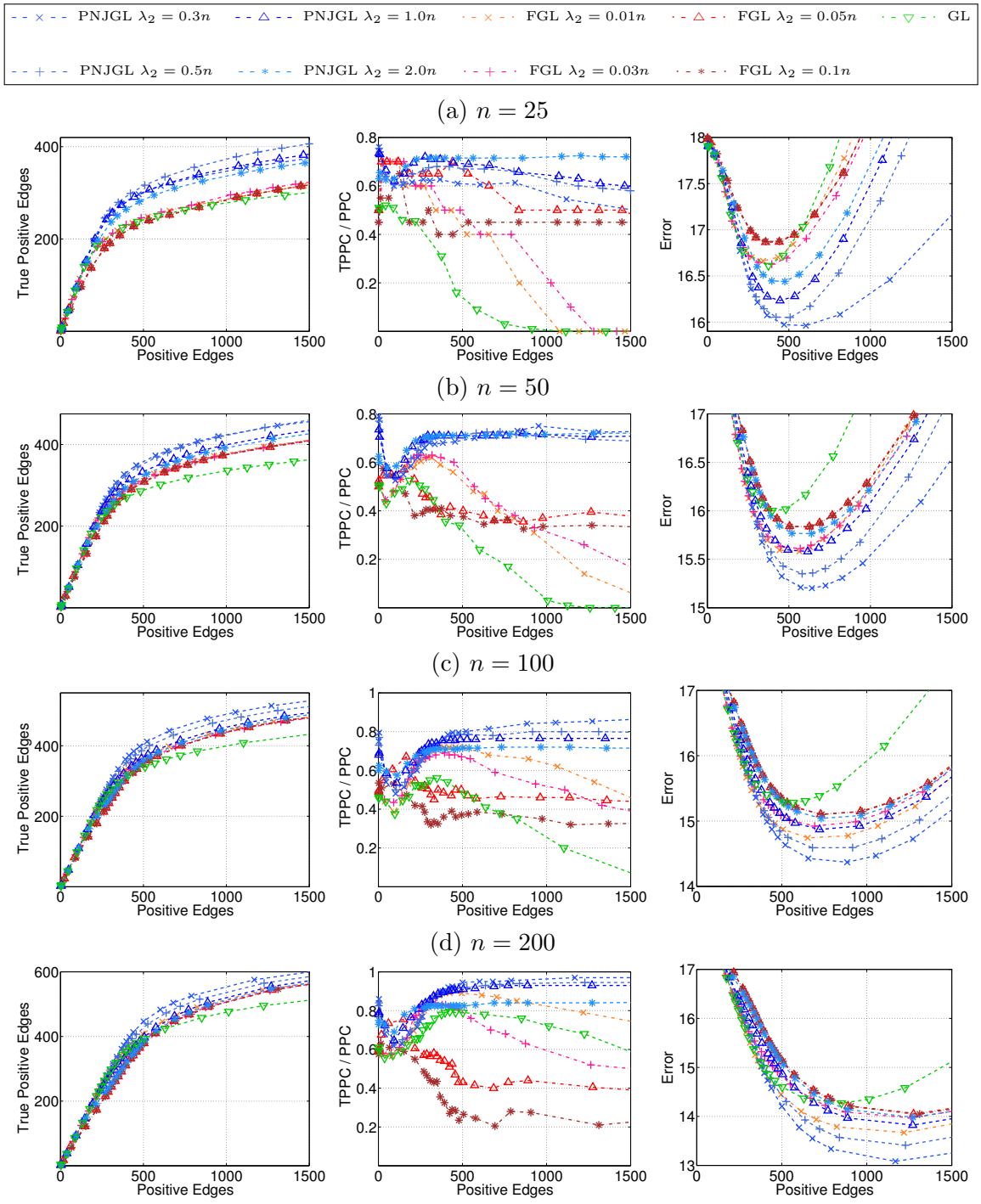


Figure 7: Simulation results on Erdos-Renyi network (Section 6.1.1) for PNJGL with $q = 2$, FGL, and GL, for *(a):* $n = 25$, *(b):* $n = 50$, *(c):* $n = 100$, *(d):* $n = 200$, when $p = 100$. Each colored line corresponds to a fixed value of λ_2, as λ_1 is varied. Axes are described in detail in Table 1. Results are averaged over 100 random generations of the data.

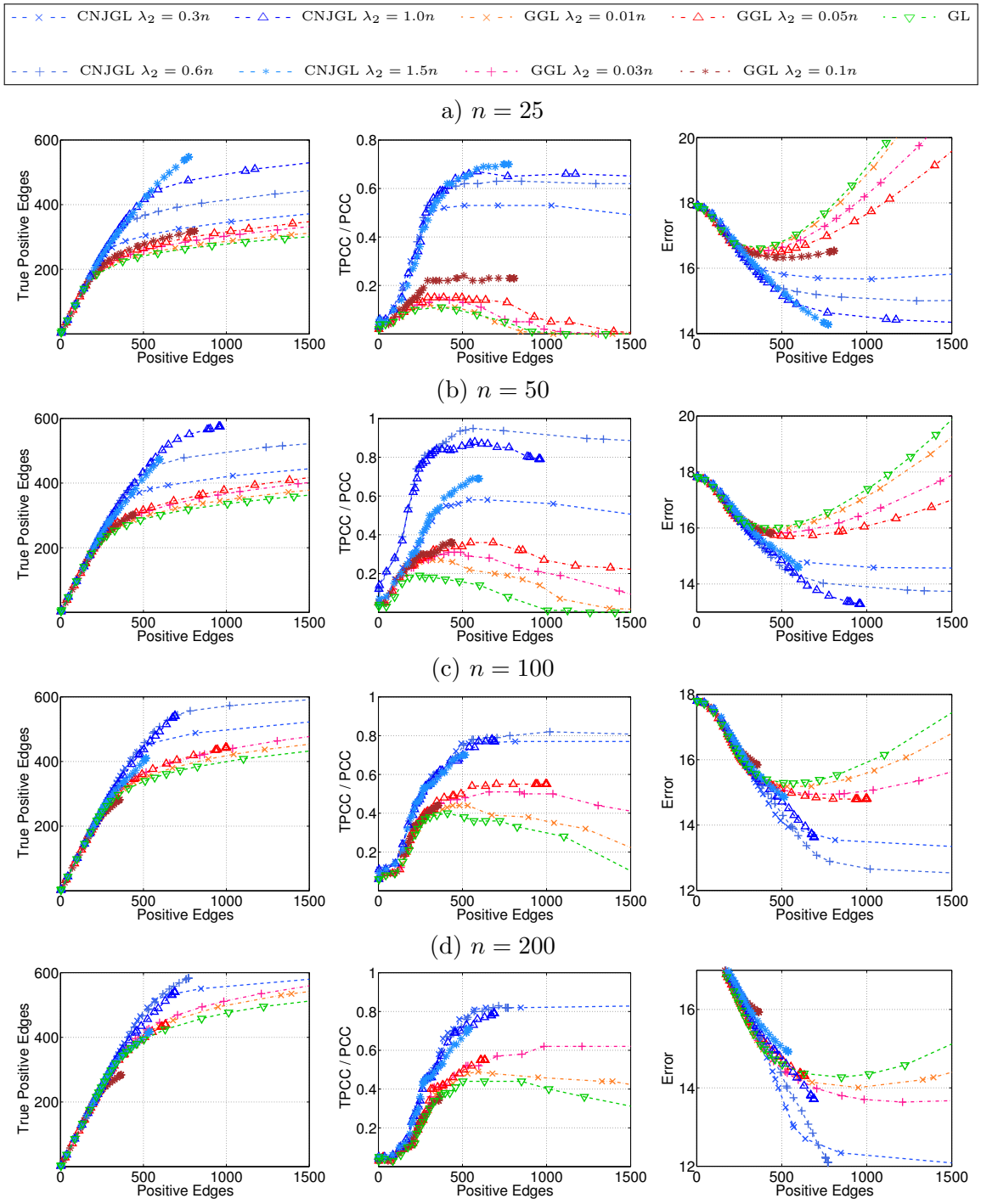
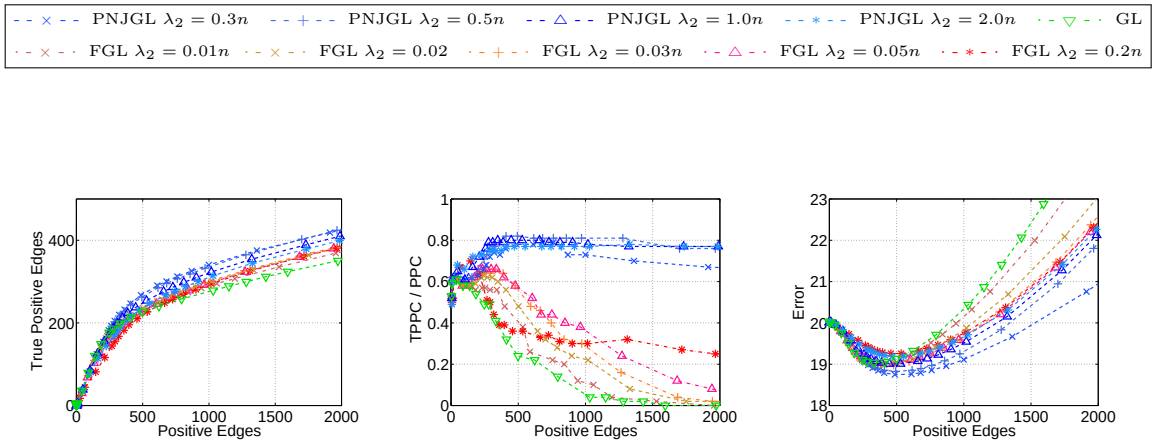


Figure 8: Simulation results on Erdos-Renyi network (Section 6.1.1) for CNJGL with $q = 2$, GGL, and GL, for *(a):* $n = 25$, *(b):* $n = 50$, *(c):* $n = 100$, *(d):* $n = 200$, when $p = 100$. Each colored line corresponds to a fixed value of λ_2, as λ_1 is varied. Axes are described in detail in Table 1. Results are averaged over 100 random generations of the data.

(a) PNJGL/FGL/GL:



(b) CNJGL/GGL/GL:

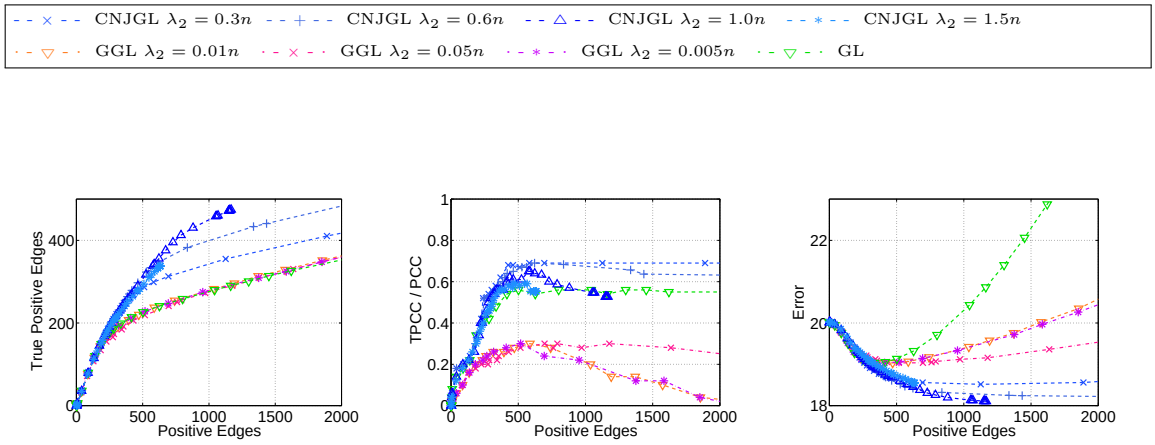
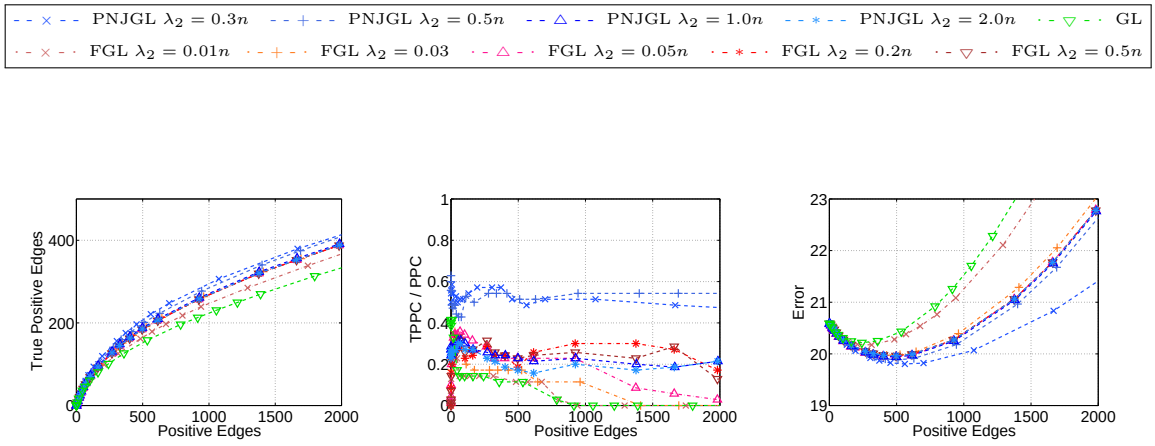


Figure 9: Simulation results on scale-free network (Section 6.1.2) for *(a):* PNJGL with $q = 2$, FGL, and GL, and *(b):* CNJGL with $q = 2$, GGL, and GL, with $p = 100$ and $n = 50$. Each colored line corresponds to a fixed value of λ_2, as λ_1 is varied. Axes are described in detail in Table 1. Results are averaged over 50 random generations of the data.

(a) PNJGL/FGL/GL:



(b) CNJGL/GGL/GL:

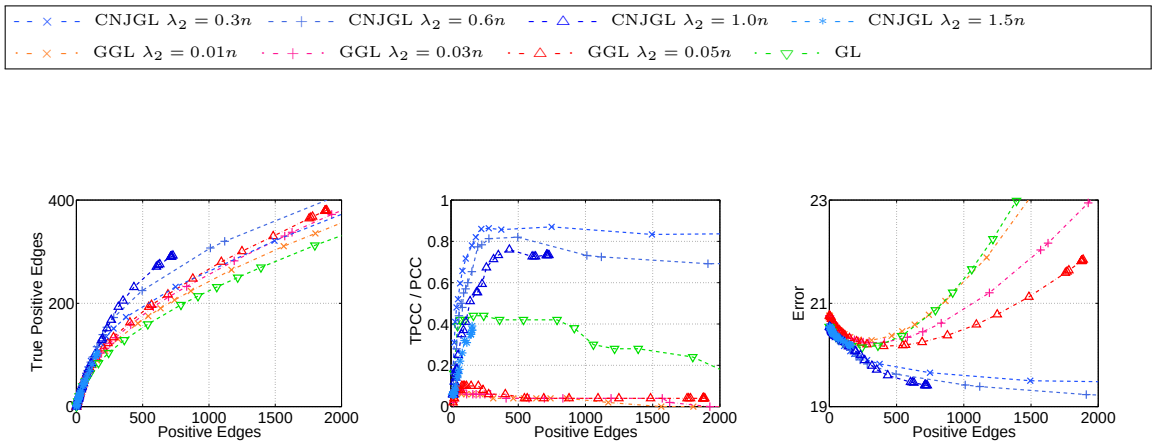
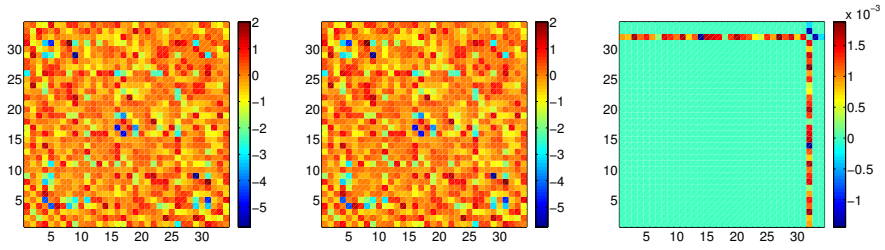


Figure 10: Simulation results on community network (Section 6.1.3) for *(a):* PNJGL with $q = 2$, FGL, and GL, and *(b):* CNJGL with $q = 2$, GGL, and GL, with $p = 100$ and $n = 50$. Each colored line corresponds to a fixed value of λ_2, as λ_1 is varied. Axes are described in detail in Table 1. Results are averaged over 50 random generations of the data.

expression data sets, after standardizing each gene to have variance one. As can be seen in Figure 11, the pattern of network differences indicates that one of the three highly-mutated genes is in fact perturbed across the two GBM subtypes. The perturbed gene is PTEN, a tumor suppressor gene, and it is known that mutations in this gene are associated with the development and progression of many cancers (see, e.g., Chalhoub and Baker, 2009).

To evaluate the performance of CNJGL in identifying genes known to be regulators, we used a manually curated list of genes that have been identified as regulators in a previous study (Gentles et al., 2009); this list includes genes annotated as transcription factors, chromatin modifiers, or translation initiation genes. We then selected a gene set from Reactome, called "G2/M checkpoints," which is relevant to cancer and contains a large number of regulators. This gene set contains 38 genes of which 15 are regulators. We applied CNJGL to the resulting 53×38 and 56×38 gene expression data sets, to see if the 15 regulators tend to be identified as co-hub genes. Figure 12 indicates that all four co-hub genes (CDC6, MCM6, CCNB1 and CCNB2) detected by CNJGL are known to be regulators.



(a) (b) (c)

Figure 11: GBM data analysis for PNJGL with $q = 2$. The sample covariance matrices S^1 and S^2 were generated from samples with two cancer subtypes, with sizes $n_1 = 53$ and $n_2 = 56$. Only the 34 genes contained in the Reactome "TCR Signaling" pathway were included in this analysis. Of these genes, three are frequently mutated in GBM: PTEN, PIK3R1, and PIK3CA. These three genes correspond to the last three columns in the matrices displayed (columns 32 through 34). PNJGL was performed with $\lambda_1 = 0$ and $\lambda_2 = 2$. We display *(a):* the estimated matrix $\hat{\Theta}^1$; *(b):* the estimated matrix $\hat{\Theta}^2$; and *(c):* the difference matrix $\hat{\Theta}^1 - \hat{\Theta}^2$. The gene PTEN is identified as perturbed.

7.2 University Webpage Data

We applied PNJGL and CNJGL to the university webpages data set from the "World Wide Knowledge Base" project at Carnegie Mellon University. This data set was pre-processed by Cardoso-Cachopo (2009). The data set describes the number of appearances of various terms, or words, on webpages from the computer science departments of Cornell, Texas, Washington and Wisconsin. We consider the 544 student webpages, and the 374 faculty webpages. We standardize the student webpage data so that each term has mean zero and standard deviation one, and we also standardize the faculty webpage data so that each term

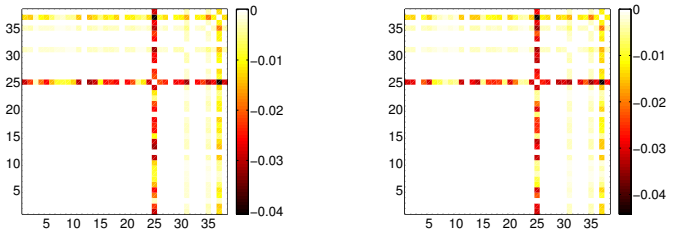


Figure 12: GBM data analysis for CNJGL with $q = 2$. The sample covariance matrices S^1 and S^2 were generated from samples with two cancer subtypes, with sizes $n_1 = 53$ and $n_2 = 56$. Only the 38 genes contained in the Reactome "G2/M checkpoints" pathway were included in this analysis. Of these genes, 15 have been previously identified as regulators. These 15 genes correspond to the last 15 columns in the matrices (columns 24 through 38). CNJGL was performed with $\lambda_1 = 13$ and $\lambda_2 = 410$. We display *(a):* the estimated matrix $\hat{\Theta}^1$; *(b):* the estimated matrix $\hat{\Theta}^2$. Four of the regulator genes are identified by CNJGL. These genes are CDC6, MCM6, CCNB1, and CCNB2.

has mean zero and standard deviation one. Our goal is to identify terms that are perturbed or co-hub between the student and faculty webpage networks. We restrict our analysis to the 100 terms with the largest entropy.

We performed 5-fold cross-validation of the log-likelihood, computed as

$$\log\det\Theta^1 - \text{trace}(S^1\Theta^1) + \log\det\Theta^2 - \text{trace}(S^2\Theta^2),$$

for PNJGL, FGL, CNJGL, GGL, and GL, using a range of tuning parameters. The results for PNJGL, FGL and GL are found in Figure 13(a). PNJGL and FGL achieve comparable log-likelihood values. However, for a fixed number of non-zero edges, PNJGL outperforms FGL, suggesting that PNJGL can achieve a comparable model fit for a more interpretable model. Figure 13(b) displays the results for CNJGL, GGL and GL. It appears that PNJGL and FGL provide the best fit to the data.

Given that PNJGL fits the data well, we highlight a particular solution, found in Figure 14. PNJGL is performed with $\lambda_1 = 27$, $\lambda_2 = 381$; these values were chosen because they result in a high log-likelihood in Figure 13(a), and yield an interpretable pair of network estimates. Several perturbed nodes are identified: *advisor, high, construct, email, applic, fax, and receiv*. The student and faculty webpage precision matrices, $\hat{\Theta}^S$ and $\hat{\Theta}^F$, are overlaid in Figure 14.

For example, the perturbed node *receiv* is connected to the terms *advis, inform,* and *student* among the student webpages. In contrast, among faculty webpages, the phrase *receiv* is connected to *associate* and *faculty*.

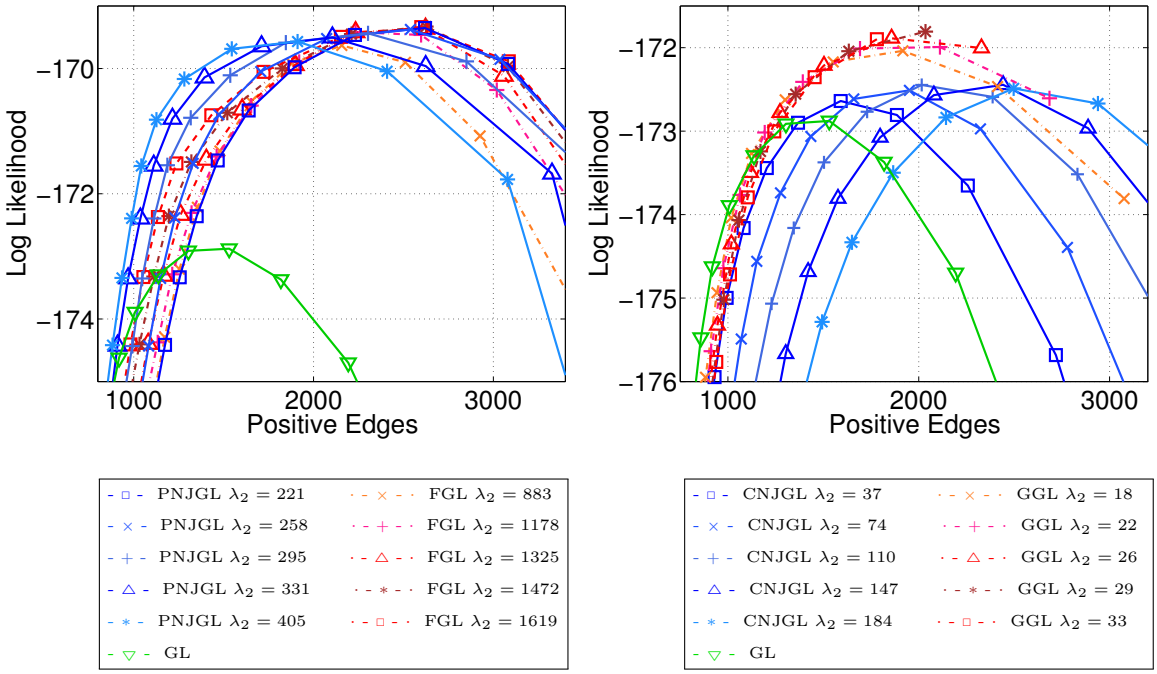


Figure 13: On the webpage data, five-fold cross-validation was performed for *(a):* PNJGL, FGL, and GL; and *(b):* CNJGL, GGL, and GL. Each colored line corresponds to a fixed value of λ_2, as λ_1 is varied. Positive edges are defined in Table 1. The cross-validated log likelihood is displayed.

8. Discussion

We have proposed node-based learning of multiple Gaussian graphical models through the use of two convex formulations, perturbed-node joint graphical lasso and cohub node joint graphical lasso. These techniques are well-motivated by many real-world applications, such as learning transcriptional regulatory networks in multiple contexts from gene expression data. Both of these formulations rely on the use of the row-column overlap norm penalty, which when applied to a matrix encourages a support that can be expressed as the union of a few rows and columns. We solve the convex optimization problems that correspond to PNJGL and CNJGL using the ADMM algorithm, which is more efficient and scalable than standard interior point methods and also first-order methods such as projected subgradient. We also provide necessary and sufficient conditions on the regularization parameters in CNJGL and PNJGL so that the optimal solutions to these formulations are block diagonal, up to a permutation of the rows and columns. When the sufficient conditions are met, any algorithm that is applicable to these two formulations can be sped up by breaking down the optimization problem into smaller subproblems. Our proposed approaches lead to better performance than two alternative approaches: learning Gaussian graphical models

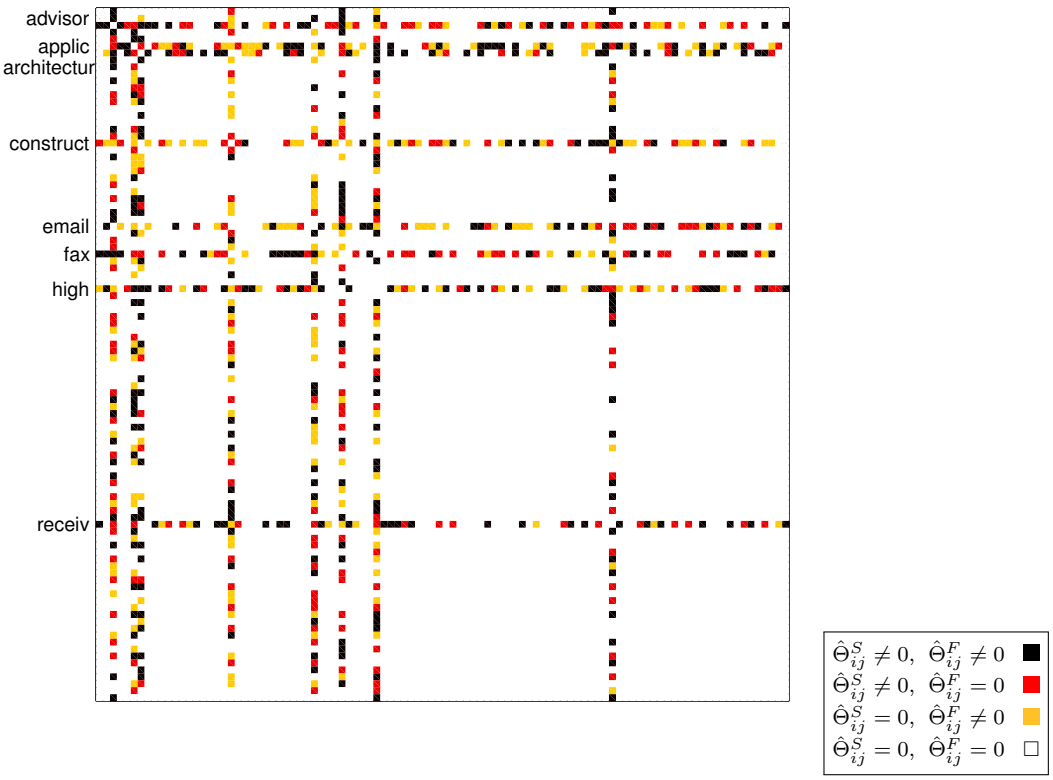


Figure 14: Student and faculty webpage precision matrices, $\hat{\Theta}^S$ and $\hat{\Theta}^F$, for PNJGL performed with $\lambda_1 = 27$, $\lambda_2 = 381$. Eight perturbed nodes are labeled. The color of each square in the figure indicates whether the corresponding edge is present in both networks, absent in both networks, or present in only the student or only the faculty network.

under the assumption of edge perturbation or shared edges, or simply learning each model separately.

We next discuss possible directions for future work.

- We have focused on promoting a row-column structure in either the difference of the networks or in the networks themselves. However, the RCON penalty can be generalized to other forms of structured sparsity. For instance, we might believe that particular sets of genes in the same pathway tend to be simultaneously activated or perturbed across multiple distinct conditions; a modification of the RCON penalty can be used in this setting.

- Convergence of the ADMM algorithm in the presence of more than two sets of variable updates has only been addressed partially in the literature. However, the PNJGL and CNJGL formulations can be rewritten along the lines of an approach given in Ma et al. (2013), so that only two sets of primal variables are involved, so that convergence is

guaranteed. We leave for future study an investigation of whether this alternative approach leads to better performance in practice.

- Transcriptional regulatory networks involve tens of thousands of genes. Hence it is imperative that our algorithms scale up to large problem sizes. In future work, speed-ups of our ADMM algorithm as well as adaptations of other fast algorithms such as the accelerated proximal gradient method or second-order methods can be considered.

- In Section 5, we presented a set of conditions that allow us to break up the CNJGL and PNJGL optimization problems into many independent subproblems. However, there is a gap between the necessary and sufficient conditions that we presented. Making this gap tighter could potentially lead to greater computational improvements.

- Tuning parameter selection in high-dimensional unsupervised settings remains an open problem. An existing approach such as stability selection (Meinshausen and Buhlmann, 2010) could be applied in order to select the tuning parameters λ_1 and λ_2 for CNJGL and PNJGL.

- The CNJGL and PNJGL formulations are aimed at jointly learning several high-dimensional Gaussian graphical models. These approaches could be modified in order to learn other types of probabilistic graphical models (see, e.g., Ravikumar et al., 2010; Yang et al., 2012).

- It is well-known that adaptive weights can improve the performance of penalized estimation approaches in other contexts (e.g., the adaptive lasso of Zou, 2006 improves over the lasso of Tibshirani, 1996). In a similar manner, the use of adaptive weights may provide improvement over the PNJGL and CNJGL proposals in this paper. Other options include *reweighted ℓ_1 norm* approaches that adjust the weights iteratively: one example is the algorithm proposed in Lobo et al. (2007) and further studied in Candes et al. (2007). This algorithm uses a weight for each variable that is proportional to the inverse of its value in the previous iteration, yielding improvements over the use of an ℓ_1 norm. This method can be seen as locally minimizing the sum of the logarithms of the entries, solved by iterative linearization. In general, any of these approaches can be explored for the problems in this paper.

`Matlab` code implementing CNJGL and PNJGL is available at `http://faculty.washington.edu/mfazel/`, `http://www.biostat.washington.edu/~dwitten/software.html`, and `http://suinlee.cs.washington.edu/software`.

Acknowledgments

The authors acknowledge funding from the following sources: NIH DP5OD009145 and NSF CAREER DMS-1252624 to DW, NSF CAREER ECCS-0847077 to MF, and Univ. Washington Royalty Research Fund to DW, MF, and SL.

Appendix A. Dual Characterization of RCON

Lemma 11 *The dual representation of Ω is given by*

$$\begin{aligned} \Omega(\Theta^1,\ldots,\Theta^K) \;=\; & \max_{\Lambda^1,\ldots,\Lambda^K\in\mathbb{R}^{p\times p}} \; \sum_{i=1}^K \langle \Lambda^i, \Theta^i\rangle \\ & \text{subject to} \quad \left\| \left[\begin{array}{c} \Lambda^1+(\Lambda^1)^T \\ \vdots \\ \Lambda^K+(\Lambda^K)^T \end{array} \right]_j \right\|_* \leq 1 \text{ for } j=1,2,\ldots,p, \end{aligned} \tag{22}$$

where $\|\cdot\|$ denotes any norm, and $\|\cdot\|_*$ its corresponding dual norm.

Proof Recall that Ω is given by

$$\begin{aligned} \Omega(\Theta^1,\ldots,\Theta^K) = \min_{V^1,\ldots,V^K\in\mathbb{R}^{p\times p}} & \left\| \left[\begin{array}{c} V^1 \\ \vdots \\ V^K \end{array} \right] \right\| \\ \text{subject to} \quad & \Theta^i = V^i + (V^i)^T,\ i=1,2,\ldots,K. \end{aligned} \tag{23}$$

Let $Z = \left[\begin{array}{c} Z^1 \\ \vdots \\ Z^K \end{array} \right]$ where $Z^k \in \mathbb{R}^{p\times p}$. Then (23) is equivalent to

$$\Omega(\Theta^1,\ldots,\Theta^K) = \min_{V^i:\,\Theta^i=V^i+(V^i)^T,\,i=1,\ldots,K} \; \max_{Z:\|Z\|_*\leq 1} \; \sum_{i=1}^K \langle Z^i, V^i\rangle, \tag{24}$$

where $\|.\|_*$ is the dual norm to $\|.\|$. Since in (24) the cost function is bilinear in the two sets of variables and the constraints are compact convex sets, by the minimax theorem, we can swap max and min to get

$$\Omega(\Theta^1,\ldots,\Theta^K) = \max_{Z:\|Z\|_*\leq 1} \; \min_{V^i:\,\Theta^i=V^i+(V^i)^T,\,i=1,\ldots,K} \sum_{i=1}^K \langle Z^i, V^i\rangle \ . \tag{25}$$

Now, note that the dual to the inner minimization problem with respect to $V^1,\ldots,V^K$ in (25) is given by

$$\begin{aligned} \underset{\Lambda^1,\ldots,\Lambda^K}{\text{maximize}} \quad & \sum_{i=1}^K \langle \Lambda^i, \Theta^i\rangle \\ \text{subject to} \quad & Z^i = \Lambda^i + (\Lambda^i)^T,\ i=1,2,\ldots,K. \end{aligned} \tag{26}$$

Plugging (26) into (25), the lemma follows. ■

By definition, the subdifferential of Ω is given by the set of all K-tuples $(\Lambda^1, \ldots, \Lambda^K)$ that are optimal solutions to problem (22). Note that if $(\Lambda^1, \ldots, \Lambda^K)$ is an optimal solution to (22), then any $(\Lambda^1 + Y^1, \ldots, \Lambda^K + Y^K)$ with skew-symmetric matrices $Y^1, \ldots, Y^K$ is also an optimal solution.

Appendix B. Proof of Theorem 4

The optimality conditions for the PNJGL optimization problem (6) with $K = 2$ are given by

$$\begin{aligned} -n_1(\Theta^1)^{-1} + n_1 S^1 + \lambda_1 \Gamma^1 + \lambda_2 \Lambda &= 0, \quad (27) \\ -n_2(\Theta^2)^{-1} + n_2 S^2 + \lambda_1 \Gamma^2 - \lambda_2 \Lambda &= 0, \quad (28) \end{aligned}$$

where Γ^1 and Γ^2 are subgradients of $\|\Theta^1\|_1$ and $\|\Theta^2\|_1$, and $(\Lambda, -\Lambda)$ is a subgradient of $\Omega_q(\Theta^1 - \Theta^2)$. (Note that $\Omega_q(\Theta^1 - \Theta^2)$ is a composition of Ω_q with the linear function $\Theta^1 - \Theta^2$, and apply the chain rule.) Also note that the right-hand side of the above equations is a zero matrix of size $p \times p$.

Now suppose that Θ^1 and Θ^2 that solve (6) are supported on T. Then since $(\Theta^1)^{-1}, (\Theta^2)^{-1}$ are supported on T, we have that

$$\begin{aligned} n_1 S^1_{T^c} + \lambda_1 \Gamma^1_{T^c} + \lambda_2 \Lambda_{T^c} &= 0, \\ n_2 S^2_{T^c} + \lambda_1 \Gamma^2_{T^c} - \lambda_2 \Lambda_{T^c} &= 0. \quad (29) \end{aligned}$$

Summing the two equations in (29) yields

$$(n_1 S^1_{T^c} + n_2 S^2_{T^c}) + \lambda_1 (\Gamma^1_{T^c} + \Gamma^2_{T^c}) = 0. \quad (30)$$

It thus follows from (30) that

$$\|n_1 S^1_{T^c} + n_2 S^2_{T^c}\|_\infty \leq \lambda_1 \|\Gamma^1_{T^c} + \Gamma^2_{T^c}\|_\infty \leq 2\lambda_1, \quad (31)$$

where here $\|\cdot\|_\infty$ indicates the maximal absolute element of a matrix, and where the second inequality in (31) follows from the fact that the subgradient of the ℓ_1 norm is bounded in absolute value by one.

We now assume, without loss of generality, that the Λ that solves (27) and (28) is symmetric. (In fact, one can easily show that there exist symmetric subgradients Γ^1, Γ^2, and Λ that satisfy (27) and (28).) Moreover, recall from Lemma 11 that $\|(\Lambda + \Lambda^T)_j\|_s \leq 1$. Therefore, $\|\Lambda_j\|_s \leq \frac{1}{2}$. Using Holder's inequality and noting that $\|y\|_1 = \langle y, \operatorname{sgn}(y) \rangle$ for a vector y, we obtain

$$\begin{aligned} \|\Lambda_{T^c}\|_1 &= \langle \Lambda_{T^c}, \operatorname{sgn}(\Lambda_{T^c}) \rangle \\ &\leq \|\operatorname{sgn}(\Lambda_{T^c})\|_q \|\Lambda_{T^c}\|_s \\ &\leq |T^c|^{\frac{1}{q}} \|\Lambda_{T^c}\|_s \\ &\leq |T^c|^{\frac{1}{q}} \|\Lambda\|_s \\ &\leq \tfrac{1}{2} |T^c|^{\frac{1}{q}} p^{\frac{1}{s}}, \end{aligned} \quad (32)$$

where the last inequality follows from the fact that $\|\Lambda\|_s^s = \sum_{j=1}^p \|\Lambda_j\|_s^s \leq p(\frac{1}{2})^s$, and where in (32), $\|A\|_q$ and $\|A\|_s$ indicate the ℓ_q and ℓ_s norms of vec(A) respectively.

From (29), we have for each $k \in \{1,2\}$ that

$$\begin{aligned} n_k\|S^k_{T^c}\|_1 &\leq \|\lambda_1\Gamma^k_{T^c} + \lambda_2\Lambda_{T^c}\|_1 \\ &\leq \lambda_1\|\Gamma^k_{T^c}\|_1 + \lambda_2\|\Lambda_{T^c}\|_1 \\ &\leq \lambda_1|T^c| + \lambda_2\frac{|T^c|^{\frac{1}{q}}p^{\frac{1}{s}}}{2}, \end{aligned}$$

where the last inequality follows from the fact that the elements of Γ^k are bounded in absolute value by one, and (32). The theorem now follows by noting from (29) that for each $k \in \{1,2\}$,

$$n_k\|S^k_{T^c}\|_\infty \leq \lambda_1\|\Gamma^k_{T^c}\|_\infty + \lambda_2\|\Lambda_{T^c}\|_\infty \leq \lambda_1 + \frac{\lambda_2}{2}.$$

Appendix C. Proof of Theorem 7

Proof The optimality conditions for the CNJGL problem (7) are given by

$$-n_k(\Theta^k)^{-1} + n_kS^k + \lambda_1\Gamma^k + \lambda_2\Lambda^k = 0, \quad k = 1,\ldots,K, \tag{33}$$

where Γ^k is a subgradient of $\|\Theta^k\|_1$. Also, the K-tuple $(\Lambda^1,\ldots,\Lambda^K)$ is a subgradient of $\Omega_q(\Theta^1 - \text{diag}(\Theta^1),\ldots,\Theta^K - \text{diag}(\Theta^K))$, and the right-hand side is a $p \times p$ matrix of zeros. We can assume, without loss of generality, that the subgradients Γ^k and Λ^k that satisfy (33) are symmetric, since Lemma 11 indicates that if $(\Lambda^1,\ldots,\Lambda^K)$ is a subgradient of $\Omega_q(\Theta^1 - \text{diag}(\Theta^1),\ldots,\Theta^k - \text{diag}(\Theta^k))$, then $((\Lambda^1 + (\Lambda^1)^T)/2,\ldots,(\Lambda^K + (\Lambda^K)^T)/2)$ is a subgradient as well.

Now suppose that $\Theta^1,\ldots,\Theta^K$ that solve (7) are supported on T. Since $(\Theta^k)^{-1}$ is supported on T for all k, we have

$$n_kS^k_{T^c} + \lambda_1\Gamma^k_{T^c} + \lambda_2\Lambda^k_{T^c} = 0. \tag{34}$$

We use the triangle inequality for the ℓ_1 norm (applied elementwise to the matrix) to get

$$n_k\|S^k_{T^c}\|_1 \leq \lambda_1\|\Gamma^k_{T^c}\|_1 + \lambda_2\|\Lambda^k_{T^c}\|_1. \tag{35}$$

We have $\|\Gamma^k\|_\infty \leq 1$ since Γ^k is a subgradient of the ℓ_1 norm, which gives $\|\Gamma^k_{T^c}\|_1 \leq |T^c|$.

Also Λ^k is a part of a subgradient to Ω_q, so by Lemma 11, $\|(\Lambda^k + (\Lambda^k)^T)_j\|_s \leq 1$ for $j \in \{1,2,\ldots,p\}$. Since Λ^k is symmetric, we have that $\|\Lambda^k_j\|_s \leq \frac{1}{2}$. Using the same reasoning as in (32) of Appendix B, we obtain

$$\|\Lambda^k_{T^c}\|_1 \leq \frac{1}{2}|T^c|^{\frac{1}{q}}p^{\frac{1}{s}}. \tag{36}$$

Combining (35) and (36) yields

$$n_k\|S^k_{T^c}\|_1 \leq \lambda_1|T^c| + \tfrac{\lambda_2}{2}|T^c|^{\frac{1}{q}}p^{\frac{1}{s}}.$$

The theorem follows by noting from (34) that

$$n_k\|S^k_{T^c}\|_\infty \leq \lambda_1\|\Gamma^k_{T^c}\|_\infty + \lambda_2\|\Lambda^k_{T^c}\|_\infty \leq \lambda_1 + \tfrac{\lambda_2}{2}.$$

■

Appendix D. Proof of Theorem 10

Assume that the sufficient condition holds. In order to prove the theorem, we must show that

$$\begin{aligned}
&\sum_{k=1}^{K} n_k(-\log\det(\Theta^k) + \langle \Theta^k, S^k\rangle)) + \lambda_1 \sum_{k=1}^{k} \|\Theta^k\|_1 + \lambda_2 h(\Theta^1, \ldots, \Theta^K) \\
> &\sum_{k=1}^{K} n_k(-\log\det(\Theta^k_T) + \langle \Theta^k_T, S^k\rangle)) + \lambda_1 \sum_{k=1}^{k} \|\Theta^k_T\|_1 + \lambda_2 h(\Theta^1_T, \ldots, \Theta^K_T).
\end{aligned}$$

By assumption,

$$h(\Theta^1, \ldots, \Theta^K) > h(\Theta^1_T, \ldots, \Theta^K_T). \tag{37}$$

We will now show that

$$n_k\langle \Theta^k, S^k\rangle + \lambda_1\|\Theta^k\|_1 \geq n_k\langle \Theta^k_T, S^k\rangle + \lambda_1\|\Theta^k_T\|_1, \tag{38}$$

or equivalently, that

$$-n_k\langle \Theta^k_{T^c}, S^k\rangle \leq \lambda_1\|\Theta^k_{T^c}\|_1. \tag{39}$$

Note that $\langle \Theta^k_{T^c}, S^k\rangle = \langle \Theta^k_{T^c}, S^k_{T^c}\rangle$. By the sufficient condition, $n_k\|S^k_{T^c}\|_\infty \leq \lambda_1$. So

$$\begin{aligned}
-n_k\langle \Theta^k_{T^c}, S^k\rangle &= -n_k\langle \Theta^k_{T^c}, S^k_{T^c}\rangle \\
&\leq \|n_k S^k_{T^c}\|_\infty \|\Theta^k_{T^c}\|_1 \\
&\leq \lambda_1\|\Theta^k_{T^c}\|_1.
\end{aligned}$$

So (39) holds, and hence (38) holds.

Finally, we apply Fischer's inequality, which states that $\det(\Theta^k) \leq \det(\Theta^k_T)$, and so

$$-\log\det(\Theta^k) \geq -\log\det(\Theta^k_T). \tag{40}$$

Combining (37), (38), and (40), the theorem holds.

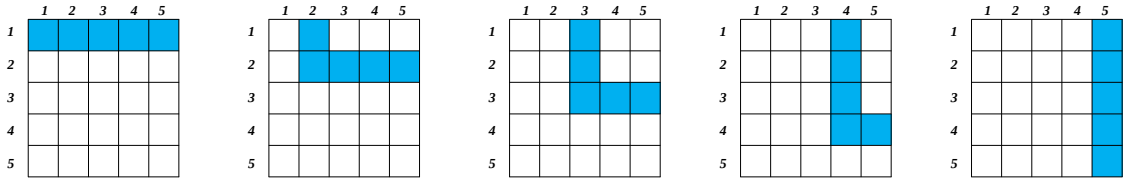
Appendix E. Connection Between RCON and Obozinski et al. (2011)

We now show that the RCON penalty with $q = 2$ can be derived from the overlap norm of Obozinski et al. (2011). For simplicity, here we restrict ourselves to the RCON with $K = 1$. The general case of $K \geq 1$ can be shown via a simple extension of this argument.

Given any symmetric $p \times p$ matrix Θ, let $\Theta_\triangle$ be the $p \times p$ upper-triangular matrix such that $\Theta = \Theta_\triangle + \Theta_\triangle^T$. That is,

$$(\Theta_\triangle)_{kl} = \begin{cases} \Theta_{kl} & \text{if } k < l \\ \Theta_{kk}/2 & \text{if } k = l \\ 0 & \text{if } k > l. \end{cases} \tag{41}$$

Now define p groups, $g_1, \ldots, g_p$, each of which contains p variables, as displayed in Figure 15. Note that these groups overlap: if $k \leq l$, then the (k, l) element of a matrix is contained in both the kth and lth groups.



(a) g_1 (b) g_2 (c) g_3 (d) g_4 (e) g_5

Figure 15: Depiction of groups $g_1, \ldots, g_5$ for a 5×5 matrix. Each group's elements are shown in blue.

The overlap norm corresponding to these groups is given by

$$\begin{aligned} \Omega^O(\Theta) \quad = \quad & \min_{V^1,\ldots,V^p \in \mathbb{R}^{p\times p}} \quad \sum_{j=1}^{p} \|V^j\|_F \\ & \text{subject to} \quad \Theta_{\triangle} = \sum_{j=1}^{p} V^j, \ \text{supp}(V^j) \subseteq g_j, \end{aligned}$$

where the relation between Θ and $\Theta_{\triangle}$ is as in Equation (41). We can rewrite this as

$$\begin{aligned} \Omega^O(\Theta) \quad = \quad & \min_{V^1,\ldots,V^p \in \mathbb{R}^{p\times p}} \quad \sum_{j=1}^{p} \|V^j\|_F \\ & \text{subject to} \quad \Theta = \sum_{j=1}^{p} V^j + (\sum_{j=1}^{p} V^j)^T, \ \text{supp}(V^j) \subseteq g_j. \end{aligned} \tag{42}$$

Now, define a $p \times p$ matrix A such that

$$A_{ij} = \begin{cases} (V^j)_{ij} & \text{if} \quad i < j \\ (V^j)_{ji} & \text{if} \quad i > j \\ (V^j)_{jj} & \text{if} \quad i = j \end{cases}.$$

Note that $A + A^T = \sum_{j=1}^{p} V^j + (\sum_{j=1}^{p} V^j)^T$. Furthermore, $\|V^j\|_F = \|A_j\|_2$, where A_j denotes the jth column of A. So we can rewrite (42) as

$$\begin{aligned} \Omega^O(\Theta) \quad = \quad & \min_{V^1,\ldots,V^p \in \mathbb{R}^{p\times p}} \quad \sum_{j=1}^{p} \|A_j\|_2 \\ & \text{subject to} \quad \Theta = A + A^T. \end{aligned}$$

This is exactly the RCON penalty with $K = 1$ and $q = 2$. Thus, with a bit of work, we have derived the RCON from the overlap norm (Obozinski et al., 2011). Our penalty is useful because it accommodates groups given by the rows and columns of a symmetric matrix in an elegant and convenient way.

Appendix F. Derivation of Updates for ADMM Algorithms

We derive the updates for ADMM algorithm when applied to PNJGL and CNJGL formulations respectively. We first begin with the PNJGL formulation.

F.1 Updates for ADMM Algorithm for PNJGL

Let $\mathcal{L}_\rho(\Theta^1, \Theta^2, Z^1, Z^2, V, W, F, G, Q^1, Q^2)$ denote the augmented Lagrangian (12). In each iteration of the ADMM algorithm, each primal variable is updated while holding the other variables fixed. The dual variables are updated using a simple dual-ascent update rule. Below, we derive the update rules for the primal variables.

F.1.1 Θ^1 Update

Note that

$$\begin{aligned}\Theta^1 &= \underset{\Theta}{\operatorname{argmin}} \quad \mathcal{L}_\rho(\Theta, \Theta^2, Z^1, Z^2, V, W, F, G, Q^1, Q^2) \\ &= \underset{\Theta}{\operatorname{argmin}} \quad n_1(-\log\det\Theta) + \rho\left\|\Theta - \tfrac{1}{2}\left((\Theta^2 + V + W + Z^1) - \tfrac{1}{\rho}(F + Q^1 + n_1 S^1)\right)\right\|_F^2.\end{aligned}$$

Now it follows from the definition of the Expand operator that

$$\Theta^1 \leftarrow \text{Expand}\left(\frac{1}{2}(\Theta^2 + V + W + Z^1) - \frac{1}{2\rho}(Q^1 + n_1 S^1 + F), \rho, n_1\right).$$

The update for Θ^2 can be derived in a similar fashion.

F.1.2 Z^1 Update

$$\begin{aligned}Z^1 &= \underset{Z}{\operatorname{argmin}} \quad \mathcal{L}_\rho(\Theta^1, \Theta^2, Z, Z^2, V, W, F, G, Q^1, Q^2) \\ &= \underset{Z}{\operatorname{argmin}} \quad \tfrac{1}{2}\left\|Z^1 - (\Theta^1 + \tfrac{Q^1}{\rho})\right\|_F^2 + \tfrac{\lambda_1}{\rho}\|Z^1\|_1.\end{aligned}$$

By the definition of the soft-thresholding operator $\mathcal{T}_1$, it follows that

$$Z^1 = \mathcal{T}_1\left(\Theta^1 + \frac{Q^1}{\rho}, \frac{\lambda_1}{\rho}\right).$$

The update for Z^2 is similarly derived.

F.1.3 V Update

$$\begin{aligned}V &= \underset{X}{\operatorname{argmin}} \quad \mathcal{L}_\rho(\Theta^1, \Theta^2, Z^1, Z^2, X, W, F, G, Q^1, Q^2) \\ &= \underset{X}{\operatorname{argmin}} \quad \tfrac{\lambda_2}{2\rho}\sum_{j=1}^{p}\|X_j\|_q + \frac{1}{2}\left\|X - \frac{1}{2}\left((W^T + \Theta^1 - \Theta^2 - W) + \frac{1}{\rho}(F - G)\right)\right\|_F^2.\end{aligned}$$

By the definition of the soft-scaling operator $\mathcal{T}_2$, it follows that

$$V = \mathcal{T}_2\left(\frac{1}{2}(W^T - W + \Theta^1 - \Theta^2) + \frac{1}{2\rho}(F - G), \frac{\lambda_2}{2\rho}\right).$$

The update for W is easy to derive and we therefore skip it.

F.2 Updates for ADMM Algorithm for CNJGL

Let $\mathcal{L}_\rho(\{\Theta^i\},\{Z^i\},\{\tilde{V}^i\},\{W^i\},\{F^i\},\{G^i\},\{Q^i\})$ denote the augmented Lagrangian (16). Below, we derive the update rules for the primal variables $\{\tilde{V}^i\}$. The update rules for the other primal variables are similar to the derivations discussed for PNJGL, and hence we omit their derivations.

The update rules for $\tilde{V}^1,\tilde{V}^2,\ldots,\tilde{V}^K$ are coupled, so we derive them simultaneously. Note that

$$\begin{aligned}
\{\tilde{V}^i\}_{i=1}^K &= \underset{A^1,\ldots,A^K}{\operatorname{argmin}} \; \mathcal{L}_\rho\left(\{\Theta^i\}_{i=1}^K,\{Z^i\}_{i=1}^K,\{A^i\}_{i=1}^K,\{W^i\}_{i=1}^K,\{F^i\}_{i=1}^K,\{G^i\}_{i=1}^K,\{Q^i\}_{i=1}^K\right) \\
&= \underset{A^1,\ldots,A^K}{\operatorname{argmin}} \; \lambda_2\sum_{j=1}^p \left\| \begin{bmatrix} A^1-\operatorname{diag}(A^1) \\ \vdots \\ A^K-\operatorname{diag}(A^K) \end{bmatrix}_j \right\|_q + \\
&\qquad \rho\sum_{i=1}^K \left\| A^i - \frac{1}{2}\left((W^i)^T+\Theta^i-W^i+\frac{1}{\rho}(F^i-G^i)\right)\right\|_F^2 .
\end{aligned}$$

Let $C^i=\frac{1}{2}((W^i)^T+\Theta^i-W^i+\frac{1}{\rho}(F^i-G^i))$. Then the update

$$\begin{bmatrix} \tilde{V}^1 \\ \vdots \\ \tilde{V}^K \end{bmatrix} \leftarrow \mathcal{T}_q\left(\begin{bmatrix} C^1-\operatorname{diag}(C^1) \\ \vdots \\ C^K-\operatorname{diag}(C^K) \end{bmatrix}, \frac{\lambda_2}{2\rho}\right) + \begin{bmatrix} \operatorname{diag}(C^1) \\ \vdots \\ \operatorname{diag}(C^K) \end{bmatrix}$$

follows by inspection.

Appendix G. Additional Simulation Results

Here we present more detailed results for an instance of the simulation study described in Section 6, for the case $n=25$. Figure 16 illustrates how the PPC, TPPC, PCC and TPCC metrics are computed. As described in Table 1, for PNJGL, PPC is given by the number of columns of $\hat{V}$ whose ℓ_2 norms exceed the threshold t_s. Figure 16(a) indicates that the two perturbed nodes in the data are identified as perturbed by PNJGL. Furthermore, given the large gap between the perturbed and non-perturbed columns, PPC is relatively insensitive to the choice of t_s. Similar results apply to the TPPC, PCC and TPCC metrics.

In order to generate Figure 16, PNJGL, FGL, CNJGL, GGL, and GL were performed using tuning parameter values that led to the best identification of perturbed and cohub nodes. However, the results displayed in Figure 16 were quite robust to the choice of tuning parameter.

References

A. Argyriou, C.A. Micchelli, and M. Pontil. Efficient first order methods for linear composite regularizers. `arXiv:1104.1436 [cs.LG]`, 2011.

O. Banerjee, L. E. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *JMLR*, 9:485–516, 2008.

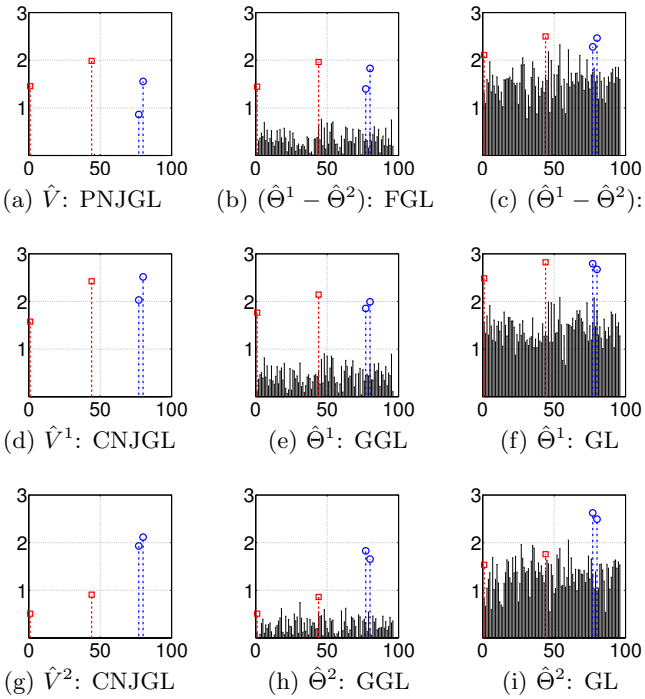


Figure 16: In all plots, the x-axis indexes the columns of the indicated matrix, and the y-axis displays the ℓ_2 norms of the columns of the indicated matrix, with diagonal elements removed. The sample size is $n = 25$. Perturbed nodes are indicated in red (with square markers), and cohub nodes are indicated in blue (with circle markers). *(a)-(c):* Detection of perturbed nodes by PNJGL with $q = 2$, FGL, and GL. *(d)-(i):* Detection of cohub nodes by CNJGL with $q = 2$, GGL, and GL. *(a):* PNJGL with $q = 2$ was performed with $\lambda_1 = 2.5$ and $\lambda_2 = 12.5$. *(b):* FGL was performed with $\lambda_1 = 2.5$ and $\lambda_2 = 0.75$. *(c):* GL was performed with $\lambda = 1.5$. *(d), (g):* CNJGL was performed with $q = 2$ and $\lambda_1 = 0.5$, $\lambda_2 = 37.5$. *(e), (h):* GGL was performed with $\lambda_1 = 0.5$ and $\lambda_2 = 2.5$. *(f), (i):* GL was performed with $\lambda = 0.75$.

S.P. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in ML*, 3(1):1–122, 2010.

E.J. Candes, M. Wakin, and S. Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier Analysis and Applications*, 14:877–905, 2007.

A. Cardoso-Cachopo, 2009. URL `http://web.ist.utl.pt/acardoso/datasets/`.

N. Chalhoub and S.J. Baker. PTEN and the PI3-kinase pathway in cancer. *Annual Review of Pathology*, 4:127–150, 2009.

X. Chen, Q. Lin, S. Kim, J.G. Carbonell, and E.P. Xing. Smoothing proximal gradient method for general structured sparse learning. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2011.

P. Danaher, P. Wang, and D. Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society, Series B*, 2013.

A. D'Aspremont, O. Banerjee, and L. El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 30(1):56–66, 2008.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, pages 2899 – 2934, 2009.

J. Eckstein. Augmented lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. Technical Report RUTCOR Research Report RRR 32-2012, Rutgers University, 2012.

J. Eckstein and D.P. Bertsekas. On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programing: Series A and B*, 55(3):293 – 318, 1992.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9:432–441, 2007.

D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1): 17–40, 1976.

A.J. Gentles, A.A. Alizadeh, S.-I. Lee, J.H. Myklebust, C.M. Shachaf, R. Levy, D. Koller, and S.K. Plevritis. A pluripotency signature predicts histologic transformation and influences survival in follicular lymphoma patients. *Blood*, 114(15):3158–66, 2009.

M. George. B-a scale-free network generation and visualization, 2007. Matlab code.

M. Grant and S. Boyd. `cvx` version 1.21. "http://cvxr.com/cvx", October 2010.

J. Guo, E. Levina, G. Michailidis, and J. Zhu. Joint estimation of multiple graphical models. *Biometrika*, 98(1):1–15, 2011.

D. Han and Z. Yuan. A note on the alternating direction method of multipliers. *Journal of Optimization Theory and Applications*, 155(1):227–238, 2012.

S. Hara and T. Washio. Learning a common substructure of multiple graphical gaussian models. *Neural Networks*, 38:23–38, 2013.

B. He, M. Tao, and X. Yuan. Alternating direction method with gaussian back substitution for separable convex programming. *SIAM Journal of Optimization*, pages 313 – 340, 2012.

M. Hong and Z. Luo. On the linear convergence of the alternating direction method of multipliers. `arXiv:1208:3922 [math.OC]`, 2012.

C.J. Hsieh, M. Sustik, I. Dhillon, and P. Ravikumar. Sparse inverse covariance estimation using quadratic approximation. *Advances in Neural Information Processing Systems*, 2011.

L. Jacob, G. Obozinski, and J.P. Vert. Group lasso with overlap and graph lasso. *Proceedings of the 26th International Conference on Machine Learning*, 2009.

W. Evan Johnson and C. Li. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8(1):118–27, 2006.

M. Kolar, L. Song, A. Ahmed, and E.P. Xing. Estimating time-varying networks. *Annals of Applied Statistics*, 4 (1):94–123, 2010.

S.L. Lauritzen. *Graphical Models*. Oxford Science Publications, 1996.

W. K. Lim, J. Wang, C. Lefebvre, and A.. Clifano. Comparative analysis of microarray normalization procedures: effects on reverse engineering gene networks. *Bioinformatics*, 23(13):282–288, 2007.

M. Lobo, M. Fazel, and S. Boyd. Portfolio optimization with linear and fixed transaction costs. *Annals of Operations Research*, 152(1):376 – 394, 2007.

S. Ma, L. Xue, and H. Zou. Alternating direction methods for latent variable Gaussian graphical model selection. *Neural Computation*, 2013.

K.V. Mardia, J. Kent, and J.M. Bibby. *Multivariate Analysis*. Academic Press, 1979.

L. Matthews, G. Gopinath, M. Gillespie, M. Caudy, D. Croft, B. de Bono, P. Garapati, J. Hemish, H. Hermjakob, B. Jassal, A. Kanapin, S. Lewis, S. Mahajan, B. May, E. Schmidt, I. Vastrik, G. Wu, E. Birney, L. Stein, and P. D'Eustachio. Reactome knowledgebase of biological pathways and processes. *Nucleic Acids Research*, 37:D619–22, 2008.

R. Mazumder and T. Hastie. Exact covariance-thresholding into connected components for large-scale graphical lasso. *Journal of Machine learning Research*, 13:723 – 736, 2012.

M. Meinshausen and P. Buhlmann. Stability selection (with discussion). *Journal of the Royal Statistical Society, Series B*, 72:417–473, 2010.

K. Mohan, M. Chung, S. Han, D. Witten, S. Lee, and M. Fazel. Structured learning of gaussian graphical models. *Advances in Neural Information Processing Systems*, 2012.

S. Mosci, S. Villa, A. Verri, and L. Rosasco. A primal-dual algorithm for group sparse regularization with overlapping groups. *Advances in Neural Information Processing Systems*, pages 2604 – 2612, 2010.

J.F.C Mota, J.M.F Xavier, P.M.Q Aguiar, and M. Puschel. A proof of convergence for the alternating direction method of multipliers applied to polyhedral-constrained functions. `arXiv:1112.2295 [math.OC]`, 2011.

G. Obozinski, L. Jacob, and J.P. Vert. Group lasso with overlaps: the latent group lasso approach. *arXiv preprint arXiv:1110.0413*, 2011.

P. Ravikumar, M.J. Wainwright, G. Raskutti, and B. Yu. Model selection in gaussian graphical models: high-dimensional consistency of l1-regularized MLE. *Advances in Neural Information Processing Systems*, 2008.

P. Ravikumar, M.J. Wainwright, and J.D. Lafferty. High-dimensional Ising model selection using l1-regularized logistic regression. *Annals of Statisitcs*, 38(3):1287–1319, 2010.

A. Rothman, E. Levina, and J. Zhu. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:494–515, 2008.

K. Scheinberg, S. Ma, and D. Goldfarb. Sparse inverse covariance selection via alternating linearization methods. *Advances in Neural Information Processing Systems*, 2010.

M. Tao and X. Yuan. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM J. Optimization*, 21(1):57–81, 2011.

R.E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society, Series B*, 67:91–108, 2005.

R.G.W. Verhaak et al. Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in PDGFRA, IDH1, EGFR, and NF1. *Cancer Cell*, 17(1):98–110, 2010.

D.M. Witten, J.H. Friedman, and N. Simon. New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, 20(4):892–900, 2011.

E. Yang, P. Ravikumar, G.I. Allen, and Z. Liu. Graphical models via generalized linear models. *Advances in Neural Information Processing Systems*, 2012.

M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(10):19–35, 2007a.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2007b.

B. Zhang and Y. Wang. Learning structural changes of Gaussian graphical models in controlled experiments. *Proc. 26th Conference on Uncertainty in Artifical Intelligence*, 2010.

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429, 2006.

Journal of Machine Learning Research 15 (2014) 489-493 Submitted 3/13; Revised 8/13; Published 2/14

The FASTCLIME Package for Linear Programming and Large-Scale Precision Matrix Estimation in R

Haotian Pang HPANG@PRINCETON.EDU
Department of Electrical Engineering
Princeton University
Olden St
Princeton, NJ 08540, USA

Han Liu HANLIU@PRINCETON.EDU
Robert Vanderbei RVDB@PRINCETON.EDU
Department of Operations Research and Financial Engineering
Princeton University
98 Charlton St
Princeton, NJ 08540, USA

Editor: Antti Honkela

Abstract

We develop an R package FASTCLIME for solving a family of regularized linear programming (LP) problems. Our package efficiently implements the parametric simplex algorithm, which provides a scalable and sophisticated tool for solving large-scale linear programs. As an illustrative example, one use of our LP solver is to implement an important sparse precision matrix estimation method called *CLIME* (Constrained L_1 Minimization Estimator). Compared with existing packages for this problem such as CLIME and FLARE, our package has three advantages: (1) it efficiently calculates the full piecewise-linear regularization path; (2) it provides an accurate dual certificate as stopping criterion; (3) it is completely coded in C and is highly portable. This package is designed to be useful to statisticians and machine learning researchers for solving a wide range of problems.

Keywords: high dimensional data, sparse precision matrix, linear programming, parametric simplex method, undirected graphical model

1. Introduction and Parametric Simplex Method

We introduce an R package, FASTCLIME, that efficiently solves a family of regularized LP problems. Our package has two major components. First, we provide an interface function that implements the parametric simplex method (PSM). This algorithm can efficiently solve large-scale LPs. Second, we apply the PSM to implement an important sparse precision matrix estimation method called CLIME (Cai et al., 2011), which is useful in high-dimensional graphical models. In the rest of this section, we describe briefly the main idea of the PSM. We refer readers who are unfamiliar with simplex methods in general to Vanderbei (2008). Consider the LP problem

$$\max c^T\beta \quad \text{subject to: } A\beta \leq b,\ \beta \geq 0,$$

where $A \in \mathbb{R}^{n\times d}$, $c \in \mathbb{R}^d$ and $b \in \mathbb{R}^n$ are given and, "$\geq$" and "$\leq$" are defined componentwise. Simplex methods expect problems in "equality form". Therefore, the first task is to introduce new variables which we tack onto the end of β and make it a longer vector $\hat{\beta}$. We rewrite the constraints with the new variables on the left and the old ones on the right:

$$\max c^T\beta_N \quad \text{subject to: } \beta_B = b - A\beta_N,\ \beta_B \geq 0,\ \beta_N \geq 0.$$

Here, $N = \{1, 2, \ldots, d\}$, $B = \{d+1, d+2, \ldots, d+n\}$, and β_B and β_N denote subvectors of $\hat{\beta}$ associated with the indices in the set. In each iteration, one variable on the left is swapped with one on the right. In general, the variables on the left are called *basic variables* and the variables on the right are called *nonbasic variables*. As the algorithm progresses, the set of nonbasic variables changes and the objective function is re-expressed purely in terms of the current nonbasic variables and therefore the coefficients for the objective function change. In a similar manner, the coefficients in the linear equality constraints also change. We denote these changed quantities by $\hat{A}$, $\hat{b}$, and $\hat{c}$. Associated with each of these updated representations of the equations of the problem is a particular candidate "solution" obtained by setting $\beta_N = 0$ and reading off the corresponding values for basic variables $\beta_B = \hat{b}$. If $\hat{b} \geq 0$, then the candidate solution is feasible (that is, satisfies all constraints). If in addition $\hat{c} \leq 0$, then the solution is optimal.

Each variant of the simplex method is defined by the rule for choosing the pair of variables to swap at each iteration. The PSM's rule is described as follows (Vanderbei, 2008). Before the algorithm starts, it parametrically perturbs b and c:

$$\max(c + \lambda c^*)^T\beta \quad \text{subject to: } A\beta \leq b + \lambda b^*,\ \beta \geq 0. \tag{1}$$

Here $b^* \geq 0$ and $c^* \leq 0$; they are called *perturbation vectors*. With this choice, the perturbed problem is optimal for large λ. The method then uses pivots to systematically reduce λ to smaller values while maintaining optimality as it goes. Once the interval of optimal λ values covers zero, we simply set $\lambda = 0$ and read off the solution to the original problem. Sometimes there is a natural choice of the perturbation vectors b^* and c^* suggested by the underlying problem for which it is known that the initial solution to the perturbed problem is optimal for some value of λ. Otherwise, the solver generates perturbations on its own.

If we are only interested in solving generic LPs, the PSM is comparable to any other variant of the simplex method. However, as we will see in the next section, the parametric simplex method is particularly well-suited to machine learning problems since the relaxation parameter λ in (1) is naturally related to the regularization parameter in sparse learning problems. This connection allows us to solve a full range of learning problems corresponding to all the regularization parameters. If a regularized learning problem can be formulated as (1), then the entire solution path can be obtained by solving **one** LP with the PSM. More precisely, at each iteration of the PSM, the current "solution" is the optimal solution for some interval of λ values. If these solutions are stored, then when λ reaches 0, we have the optimal solution to every λ-perturbed problem for all λ between 0 and the starting value.

We describe the application of the PSM to sparse precision matrix estimation in Section 2. Numerical benchmark and comparisons with other implementations of the precision matrix estimation are provided in Section 3. For details of examples and how to use the package, we refer the user to the companion vignette and package references.

2. Application to Sparse Precision Matrix Estimation

Estimating large covariance and precision matrices is a fundamental problem which has many applications in modern statistics and machine learning. We denote $\Omega = \Sigma^{-1}$, where Σ is the population covariance matrix. Under Gaussian model, the sparse precision matrix Ω encodes the conditional independence relationships among the variables and that is why sparse precision matrices are closely related to undirected graphs. Recently, several sparse precision matrix estimation methods have been proposed, including penalized maximum-likelihood estimation (MLE) (Banerjee et al., 2008; Friedman et al., 2007b,a, 2010), neighborhood selection method (Meinshausen and Bühlmann, 2006) and LP based methods (Cai et al., 2011; Yuan, 2011). In general, solvers based on penalized MLE methods, such as `QUIC` (Hsieh et al., 2011) and `HUGE` (Zhao and Liu, 2012), are faster than the others. However, these MLE methods aim to find an approximate solution quickly whereas the linear programming methods are designed to find solutions that are correct essentially to machine precision. The comparison of classification performance can be found in Cai et al. (2011) and it is shown that CLIME uniformly outperforms the MLE methods. Because of the good theoretical properties shown by CLIME, we would like to develop a fast algorithm for implementing this method which serves as an important building block for more sophisticated learning algorithms.

The CLIME solves the following optimization problem

$$\min \|\Omega\|_1 \quad \text{subject to: } \|\Sigma_n \Omega - I_d\|_{\max} \leq \lambda \text{ and } \Omega \in \mathbb{R}^{d\times d},$$

where I_d is the d-dimensional identity matrix, Σ_n is the sample covariance matrix, and $\lambda > 0$ is a tuning parameter. Here $\|\Omega\|_1 = \sum_{j,k} |\Omega|_{j,k}$ and $\|\cdot\|_{\max}$ is the elementwise sup-norm. This minimization problem can be further decomposed into d smaller problems, which allows us to recover the precision matrix in a column by column fashion. For the i-th subproblem, we get the i-th column of Ω, denoted as $\hat{\beta}$, by solving

$$\min \|\beta\|_1 \quad \text{subject to: } \|\Sigma_n \beta - e_i\|_\infty \leq \lambda \text{ and } \beta \in \mathbb{R}^d, \tag{2}$$

where $\|\beta\|_1 = \sum_{j=1}^d |\beta_j|$ and $e_i \in \mathbb{R}^d$ is the i-th basis vector.

The original clime package manually sets a default path for λ and solves the LP problem for each different value of λ. In this paper, we propose to use the PSM to solve this problem more efficiently. CLIME can be easily formulated in parametric simplex LP form. Let β^+ and β^- be the positive and negative parts of β. Since $\beta = \beta^+ - \beta^-$ and $\|\beta\|_1 = \beta^+ + \beta^-$, Equation (2) becomes:

$$\min \beta_+ + \beta_- \quad \text{subject to: } \begin{pmatrix} \Sigma_n & -\Sigma_n \\ -\Sigma_n & \Sigma_n \end{pmatrix} \begin{pmatrix} \beta^+ \\ \beta^- \end{pmatrix} \leq \begin{pmatrix} \lambda + e_i \\ \lambda - e_i \end{pmatrix}. \tag{3}$$

Comparing (1) and (3), we can give the following identification:

$$A = \begin{pmatrix} \Sigma_n & -\Sigma_n \\ -\Sigma_n & \Sigma_n \end{pmatrix}, \quad b = \begin{pmatrix} e_i \\ -e_i \end{pmatrix}, \quad c = \begin{pmatrix} -1 \\ \vdots \\ -1 \end{pmatrix}, \quad b^* = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \quad c^* = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}.$$

The path of λ defined by the PSM corresponds to the path of λ as described in CLIME. Therefore, CLIME can be solved efficiently by the PSM; furthermore, when the optimal solution is sparse, the parametric simplex is able to find the optimal solution after very few iterations.

3. Performance Benchmark

For our experiments we focused solely on CLIME. We compare the timing performance of our package with the packages flare and clime. Flare uses the Alternating Direction Method of Multipliers (ADMM) algorithm to evaluate CLIME (Li et al., 2012), whereas clime solves a sequence of LP problems for a certain specific set of values of λ. As explained in Section 1, our method calculates the solution for *all* λ, while flare and clime use a discrete set of λ values as specified in the function. We fix the sample size n to be 200 and vary the data dimension d from 50 to 800. We generate our data using `fastclime.generator`, without any particular data structures. Clime and fastclime are based on algorithms that solve problems to machine precision (10^{-5}). Flare, on the other hand, is an ADMM-based algorithm that stops when the change from one iteration to the next drops below the same threshold. As shown in Table 1, fastclime performances significantly faster than clime when d equals 50 and 100. When d becomes large, we are not able to obtain results from clime in one hour. We also notice that, in most cases, fastclime performances consistently better than flare, and it has a smaller deviation compared with flare. The reason fastclime outperforms the other methods is primarily because the PSM only solves one LP problem to get the entire solution path for all λ quickly and without using much memory. The code is implemented on a i5-3320 2.6GHz computer with 8G RAM, and the R version used is 2.15.0.

Method	d=50	d=100	d=200	d=400	d=800
clime	103.52(9.11)	937.37(6.77)	N/A	N/A	N/A
flare	0.632(0.335)	1.886(0.755)	10.770(0.184)	74.106(33.940)	763.632(135.724)
fastclime	0.248(0.0148)	0.928(0.0268)	9.928(3.702)	53.038(1.488)	386.880(58.210)

Table 1: Average Timing Performance of Three Solvers in Seconds

4. Summary and Acknowledgements

We developed a new package named fastclime, for solving linear programming problems with a relaxation parameter and high dimensional sparse precision matrix estimation. We plan to maintain and support this package in the future. Han Liu is supported by NSF Grants III-1116730 and NSF III-1332109, NIH R01MH102339, NIH R01GM083084, and NIH R01HG06841, and FDA HHSF223201000072C. Robert Vanderbei is supported by ONR Grant N000141310093.

References

O. Banerjee, L. E. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation. *Journal of Machine Learning Research*, 9:485–516, 2008.

T. Cai, W. Liu, and X. Luo. A constrained l_1 minimization approach to sparse precision matrix estimation. *J. American Statistical Association*, 106:594–607, 2011.

J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007a.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2007b.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 2010.

C-J. Hsieh, M. A. Sustik, I. S. Dhillon, and P. Ravikumar. Sparse inverse covariance matrix estimation using quadratic approximation. *Advances in Neural Information Processing Systems*, 24, 2011.

X. Li, T. Zhao, X. Yuan, and H. Liu. An R package flare for high dimensional linear regression and precision matrix estimator. *R Package Vigette*, 2012.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

R. Vanderbei. *Linear Programming, Fundations and Extensions*. Springer, 2008.

M. Yuan. High dimensional inverse covariance matrix estimation via linear programming. *Journal of Machine Learning Research*, 11:2261–2286, 2011.

T. Zhao and H. Liu. The huge package for high-dimensional undirected graph estimation in R. *Journal of Machine Learning Research*, 13:1059–1062, 2012.

Journal of Machine Learning Research 15 (2014) 495-499 Submitted 9/13; Revised 12/13; Published 2/14

LIBOL: A Library for Online Learning Algorithms

Steven C.H. Hoi CHHOI@NTU.EDU.SG
Jialei Wang JL.WANG@NTU.EDU.SG
Peilin Zhao ZHAO0106@NTU.EDU.SG
School of Computer Engineering
Nanyang Technological University
Singapore 639798

Editor: Mark Reid

Abstract

LIBOL is an open-source library for large-scale online learning, which consists of a large family of efficient and scalable state-of-the-art online learning algorithms for large-scale online classification tasks. We have offered easy-to-use command-line tools and examples for users and developers, and also have made comprehensive documents available for both beginners and advanced users. LIBOL is not only a machine learning toolbox, but also a comprehensive experimental platform for conducting online learning research.

Keywords: online learning, massive-scale classification, big data analytics

1. Introduction

Online learning represents an important family of efficient and scalable machine learning algorithms for large-scale applications. In general, online learning algorithms are fast, simple, and often make few statistical assumptions, making them applicable to a wide range of applications. Online learning has been actively studied in several communities, including machine learning, statistics, and artificial intelligence. Over the past years, a variety of online learning algorithms have been proposed, but so far there is very few comprehensive library which includes most of the state-of-the-art algorithms for researchers to make easy side-by-side comparisons and for developers to explore their various applications.

In this work, we develop LIBOL as an easy-to-use online learning tool that consists of a large family of classical and recent state-of-the-art online learning algorithms for large-scale online classification tasks. In contrast to many existing software for large-scale data classification, LIBOL enjoys significant advantages for massive-scale data classification in the era of big data nowadays, especially in efficiency, scalability, parallelization, and adaptability. Our goal is to make LIBOL not only a useful machine learning tool for practical users, but also a comprehensive experimental platform for machine learning researchers to conduct online learning research. The LIBOL software is available at `http://libol.stevenhoi.org/`. A more comprehensive and up-to-date documentation for the latest software is available at `http://libol.stevenhoi.org/LIBOL_manual.pdf`.

2. A Family of Online Learning Algorithms for Linear Classification

Online learning operates on a sequence of data examples with time stamps. At each step t, the learner receives an incoming example $\mathbf{x}_t \in \mathcal{X}$ in a d-dimensional vector space, that is, $\mathcal{X} = \mathbb{R}^d$. It first attempts to predict the class label of the incoming instance, $\hat{y}_t = \mathsf{sgn}(f(\mathbf{x}_t; \mathbf{w}_t)) = \mathsf{sgn}(\mathbf{w}_t \cdot \mathbf{x}_t) \in \mathcal{Y}$, and $\mathcal{Y} = \{-1, +1\}$ for binary classification tasks. After making the prediction, the true label $y_t \in \mathcal{Y}$ is revealed, and the learners then computes the loss $\ell(y_t, \hat{y}_t)$ based on some criterion to measure the difference between the learner's prediction and the revealed true label y_t. Based on the result of the loss, the learner finally decides when and how to update the classification model at the end of each learning step. The following algorithmic framework gives an overview of most online learning algorithms[1] for linear classification, where $\Delta(\mathbf{w}_t; (\mathbf{x}_t, y_t))$ denotes the update of the classification models. Different online learning algorithms in general are distinguished in terms of different definitions and designs of the loss function $\ell(\cdot)$ and their various updating functions $\Delta(\cdot)$.

Algorithm 1: LIBOL: Online Learning Framework for Linear Classification.

```
Initialize: w_1 = 0
for t = 1, 2, ..., T do
    The learner receives an incoming instance: x_t ∈ X;
    The learner predicts the class label: ŷ_t = sgn(f(x_t; w_t));
    The true class label is revealed from the environment: y_t ∈ Y;
    The learner calculates the suffered loss: ℓ(w_t; (x_t, y_t));
    if ℓ(w_t; (x_t, y_t)) > 0 then
        The learner updates the classification model:
        w_{t+1} ← w_t + Δ(w_t; (x_t, y_t));
    end
end
```

The goal of our work is to implement a large family of diverse online learning methods in literature to facilitate research and development of online learning techniques to real-world applications. In particular, this software consists of 29 different online learning algorithms and variants for both binary and multiclass classification tasks. In general, they can be grouped into two major categories: (i) first order learning (Rosenblatt, 1958; Crammer et al., 2006), and (ii) second order learning (Dredze et al., 2008; Wang et al., 2012a; Yang et al., 2009). Examples of online learning algorithms in the first order learning category include the following list of classical and popular algorithms:

- Perceptron: the classical online learning algorithm (Rosenblatt, 1958);
- ALMA: A New Approximate Maximal Margin Classification Algorithm (Gentile, 2001);
- ROMMA: the relaxed online maximum margin algorithms (Li and Long, 2002);
- OGD: the Online Gradient Descent (OGD) algorithms (Zinkevich, 2003);
- PA: the Passive Aggressive (PA) online learning algorithms (Crammer et al., 2006).

1. Note that second order online learning algorithms follow a slightly different procedure.

To improve the efficacy of first order learning methods, recent years have witnessed the emerging active studies of second order online learning algorithms. One family of recent second order algorithms (Dredze et al., 2008) assume the weight vector follows a Gaussian distribution $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ with mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$. The model parameters, including both Σ and $\boldsymbol{\mu}$ are updated in the online learning process. Examples of the second order online learning algorithms include the following:

- SOP: the Second Order Perceptron (SOP) algorithm (Cesa-Bianchi et al., 2005);
- CW: the Confidence-Weighted (CW) learning algorithm (Dredze et al., 2008);
- IELLIP: online learning algorithms by improved ellipsoid method (Yang et al., 2009);
- AROW: the Adaptive Regularization of Weight Vectors (Crammer et al., 2009);
- NAROW: New variant of Adaptive Regularization (Orabona and Crammer, 2010);
- NHERD: the Normal Herding method via Gaussian Herding (Crammer and Lee, 2010)
- SCW: the recently proposed Soft Confidence Weighted algorithms (Wang et al., 2012a).

3. The Software Package

The current version (V0.3.0) of LIBOL implements a large family of online learning algorithms and their variants, including 16 algorithms for binary classification, and 13 algorithms for multiclass classification. The package includes the MATLAB library and command-line tools for both online binary and multiclass classification tasks. In addition to MATLAB implementation, we also provide C/C++ implementation for the core functions. The data formats used by this software are compatible with popular machine learning and data mining packages, such as LIBSVM, SVM-light, and WEKA, etc.

3.1 Practical Usage

To illustrate the online learning procedure, we take two data sets from the LIBSVM website, including one small data set "svmguide3" with 1243 instances and one large data set "ijcnn1" with 141,691 instances. In the following example, we use the default "Perceptron" algorithm to demo the usage of LIBOL for a binary classification ('bc') task:

```
$ demo('bc','Perceptron','svmguide3')
```

The results output by the above command are summarized as follows:

Algorithm:	mistake rate	nb of updates	cpu time (seconds)
Perceptron	0.3318 +/- 0.0118	412.45 +/- 14.66	0.0516 +/- 0.0008

To ease researchers to run a full set of experiments for side-by-side comparisons of different algorithms, we offer a very easy-to-use example program as follows:

```
$ run_experiment('bc','svmguide3')
```

The above command will run side-by-side comparison of varied online learning algorithms on the given data set fully automatically, including all the parameter settings and selection. The full set of experimental results will be generated by the library automatically, as shown in Table 1 and Figure 1. This library provides a fairly easy-to-use testbed to facilitate online learning researchers to develop their new algorithms and conduct side-by-side comparisons with the state-of-the-art algorithms with the minimal efforts.

Data Set:	svmguide3 (#samples=1243,#dimensions=36)			ijcnn1 (#samples=141,691,#dimensions=22)		
Algorithm	mistake	# updates	time (s)	mistake	# updates	time (s)
Perceptron	0.332 ± 0.012	412.4 ± 14.7	0.052 ± 0.002	0.106 ± 0.000	15059.9 ± 65.1	5.668 ± 0.064
ROMMA	0.329 ± 0.019	409.3 ± 23.2	0.056 ± 0.001	0.101 ± 0.001	14284.2 ± 89.8	5.823 ± 0.111
aROMMA	0.328 ± 0.018	500.1 ± 29.1	0.055 ± 0.001	0.101 ± 0.001	14776.0 ± 101.8	5.665 ± 0.062
ALMA	0.230 ± 0.006	592.9 ± 6.6	0.062 ± 0.001	0.071 ± 0.000	21474.0 ± 80.4	6.662 ± 0.127
OGD	0.237 ± 0.003	636.5 ± 4.2	0.064 ± 0.002	0.095 ± 0.000	27465.8 ± 31.1	6.369 ± 0.107
PA	0.318 ± 0.013	721.1 ± 18.3	0.060 ± 0.001	0.102 ± 0.001	33847.9 ± 135.2	5.955 ± 0.091
PA1	0.236 ± 0.002	763.4 ± 11.5	0.064 ± 0.001	0.077 ± 0.000	28376.3 ± 84.2	6.352 ± 0.109
PA2	0.253 ± 0.007	1131.5 ± 15.6	0.069 ± 0.001	0.081 ± 0.000	61093.8 ± 199.3	6.876 ± 0.114
SOP	0.297 ± 0.012	369.1 ± 14.8	0.095 ± 0.002	0.102 ± 0.001	14470.7 ± 81.3	10.616 ± 0.096
IELLIP	0.332 ± 0.013	412.7 ± 16.1	0.081 ± 0.002	0.119 ± 0.001	16876.8 ± 72.9	8.079 ± 0.082
CW	0.299 ± 0.011	704.6 ± 19.1	0.118 ± 0.002	0.093 ± 0.001	30648.3 ± 166.2	9.499 ± 0.110
NHERD	0.217 ± 0.007	1150.5 ± 27.7	0.096 ± 0.002	0.084 ± 0.001	86660.7 ± 2692.6	9.735 ± 0.133
AROW	0.222 ± 0.004	1219.5 ± 8.1	0.112 ± 0.002	0.082 ± 0.000	74247.1 ± 846.3	10.164 ± 0.069
NAROW	0.276 ± 0.042	1193.8 ± 23.2	0.118 ± 0.002	0.095 ± 0.008	103843.6 ± 8841.3	12.027 ± 0.467
SCW	0.206 ± 0.004	593.4 ± 13.9	0.085 ± 0.002	0.060 ± 0.002	11077.3 ± 678.3	7.921 ± 0.144
SCW2	0.212 ± 0.009	802.0 ± 73.2	0.092 ± 0.002	0.070 ± 0.001	30833.8 ± 2116.8	8.681 ± 0.150

Table 1: Comparison of a variety of online learning algorithms on two data sets.

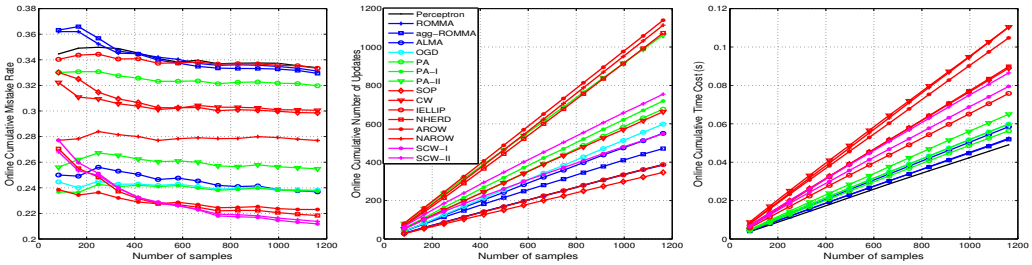


Figure 1: Comparison of a variety of online learning algorithms on data set "svmguide3".

3.2 Documentation and Design

The LIBOL package comes with comprehensive documentation. The README file describes the setup and usage. Users can read the "Quick Start" section to begin shortly. All the functions and related data structures are explained in detail. If the README file does not give the information users want, they can check the online FAQ. In addition to software documentation, theoretical properties of the algorithms and comparisons can be found in Wang et al. (2012a). The authors are also willing to answer any further questions.

The design principle is to keep the package simple, easy to read and extend. All codes follow the MATALB standards with core functions implemented in C/C++. It generally needs no external libraries, except for the support of popular data formats, such as LIBSVM and WEKA data sets for which existing libraries are included. LIBOL is written in a modular way. All the online learning algorithms can be called via the uniform "ol_train()" function by setting proper options. One can easily develop a new algorithm and make side-by-side comparisons with the existing ones in the package. Our goal is to make LIBOL not only a machine learning tool, but also an experimental platform for online learning research.

4. Conclusion

LIBOL is an easy-to-use open-source package for online learning research and development. The current version of LIBOL includes a large number of online learning algorithms for online classification tasks. LIBOL is still being improved by feedback from practical users and new research results (Zhao et al., 2011a,b; Wang et al., 2012b; Hoi et al., 2013a,b). We hope to make LIBOL not only a useful machine learning tool, but also an ideal research platform for conducting online learning research. The ultimate goal is to make easy learning with massive data streams for tackling the grand challenge of big data analytics.

References

Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. A second-order perceptron algorithm. *SIAM J. Comput.*, 34(3):640–668, 2005.

Koby Crammer and Daniel D. Lee. Learning via gaussian herding. In *NIPS*, pages 451–459, 2010.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *NIPS*, pages 345–352, 2009.

Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *ICML*, pages 264–271, 2008.

Claudio Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

Steven C. H. Hoi, Rong Jin, Peilin Zhao, and Tianbao Yang. Online multiple kernel classification. *Machine Learning*, 90(2):289–316, 2013a.

Steven C. H. Hoi, Jialei Wang, Peilin Zhao, Jinfeng Zhuang, and Zhi-Yong Liu. Large scale online kernel classification. In *IJCAI*, 2013b.

Yi Li and Philip M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1-3):361–387, 2002.

Francesco Orabona and Koby Crammer. New adaptive algorithms for online classification. In *NIPS*, pages 1840–1848, 2010.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.*, 7:551–585, 1958.

Jialei Wang, Peilin Zhao, and Steven C. H. Hoi. Exact soft confidence-weighted learning. *ICML*, 2012a.

Jialei Wang, Peilin Zhao, and Steven CH Hoi. Cost-sensitive online classification. In *IEEE 12th International Conference on Data Mining (ICDM)*, pages 1140–1145. IEEE, 2012b.

Liu Yang, Rong Jin, and Jieping Ye. Online learning by ellipsoid method. In *ICML*, page 145, 2009.

Peilin Zhao, Steven C. H. Hoi, and Rong Jin. Double updating online learning. *Journal of Machine Learning Research*, 12:1587–1615, 2011a.

Peilin Zhao, Steven C. H. Hoi, Rong Jin, and Tianbao Yang. Online auc maximization. In *ICML*, pages 233–240, 2011b.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.

Journal of Machine Learning Research 15 (2014) 501-532 Submitted 2/13; Revised 10/13; Published 2/14

Improving Markov Network Structure Learning Using Decision Trees

Daniel Lowd LOWD@CS.UOREGON.EDU
Department of Computer and Information Science
University of Oregon
Eugene, OR 97403, USA

Jesse Davis JESSE.DAVIS@CS.KULEUVEN.BE
Department of Computer Science
Katholieke Universiteit Leuven
3001 Heverlee, Belgium

Editor: Max Chickering

Abstract

Most existing algorithms for learning Markov network structure either are limited to learning interactions among few variables or are very slow, due to the large space of possible structures. In this paper, we propose three new methods for using decision trees to learn Markov network structures. The advantage of using decision trees is that they are very fast to learn and can represent complex interactions among many variables. The first method, DTSL, learns a decision tree to predict each variable and converts each tree into a set of conjunctive features that define the Markov network structure. The second, DT-BLM, builds on DTSL by using it to initialize a search-based Markov network learning algorithm recently proposed by Davis and Domingos (2010). The third, DT+L1, combines the features learned by DTSL with those learned by an L1-regularized logistic regression method (L1) proposed by Ravikumar et al. (2009). In an extensive empirical evaluation on 20 data sets, DTSL is comparable to L1 and significantly faster and more accurate than two other baselines. DT-BLM is slower than DTSL, but obtains slightly higher accuracy. DT+L1 combines the strengths of DTSL and L1 to perform significantly better than either of them with only a modest increase in training time.

Keywords: Markov networks, structure learning, decision trees, probabilistic methods

1. Introduction

A Markov network is an undirected, probabilistic graphical model for compactly representing a joint probability distribution over a set of random variables. In general, these variables can be discrete, continuous, or a mix; in this paper, we consider the case when all variables are discrete. Markov networks have been widely used in a number of domains, including computer vision, computational biology, and natural language processing. The structure of a Markov network defines which direct interactions among the variables are included in the model. This structure can be represented as a set of features, each of which is a Boolean-valued function of a subset of the variables. The parameters of a Markov network define the relative strength of those interactions. Selecting a Markov network structure that

includes the most important interactions in a domain is therefore essential for building an accurate model of that domain.

For some tasks, such as image processing, the structure of the Markov network may be hand crafted to fit the problem. In other problems, the structure is unknown and must be learned from data. These learned structures may be interesting in themselves, since they show the most significant direct interactions in the domain. In many domains, however, the goal is not an interpretable structure but an accurate final model. It is this last scenario that is the focus of our paper: learning the structure of a Markov network in order to accurately estimate marginal and conditional probabilities.

Learning an effective structure is difficult due to the very large structure space—the number of possible sets of conjunctive features is doubly-exponential in the number of variables. As a result, most previous approaches to learning Markov network structure are either very slow or limited to relatively simple features, such as only allowing pairwise interactions. In this paper, we propose to overcome these limitations by using decision tree learners, which are able to quickly learn complex structures involving many variables.

Our first method, DTSL (Decision Tree Structure Learner), learns probabilistic decision trees to predict the value of each variable and then converts the trees into sets of conjunctive features. We propose and evaluate several different methods for performing the conversion. Finally, DTSL merges all learned features into a global model. Weights for these features can be learned using any standard Markov network weight learning method. DTSL is similar in spirit to work by Ravikumar et al. (2010), who learn a sparse logistic regression model for each variable and combine the features from each local model into a global network structure. DTSL can also be viewed as converting a dependency network (Heckerman et al., 2000) with decision trees into a consistent Markov network.

Our second method, DT-BLM (Decision Tree Bottom-Up Learning), builds on DTSL by using the BLM algorithm of Davis and Domingos (2010) to further refine the structure learned by DTSL. This algorithm is much slower, but usually more accurate than DTSL. Furthermore, it serves as an example of how decision trees can be used to improve search-based structure learning algorithms by providing a good initial structure.

Our third method, DT+L1, combines the structure learned by DTSL with the pairwise interactions learned by L1-regularized logistic regression (L1) (Ravikumar et al., 2010). The trees used by DTSL are good at capturing higher-order interactions, but each leaf is mutually exclusive. In contrast, L1 captures many independent interaction terms, but each interaction is between just two variables. Their combination offers the potential to represent both kinds of interaction, leading to better performance in many domains.

We conducted an extensive empirical evaluation on 20 real-world data sets. We found that DTSL offers similar accuracy and speed as L1, performing better on data sets where it finds interesting tree structure and worse on data sets where it does not. Over 90% of the running time was spent learning weights, so there is potential to improve learning times even more with more sophisticated weight learning algorithms. The hybrid DT-BLM algorithm is often more accurate than DTSL, but is also much slower due to the additional refinement step. DT+L1 often has the best overall accuracy and runs much faster than DT-BLM, making it a very good algorithm overall. We also evaluated two other baseline structure learners, but they were not competitive with L1 and the three variants of DTSL.

This journal paper is an extended and improved version of the conference paper (Lowd and Davis, 2010). The extensions include two additional algorithms (DT-BLM and DT+L1) and more extensive experiments, including seven additional data sets and learning curves. The presentation has also been expanded and polished.

2. Markov Networks

This section provides a basic overview about Markov networks.

2.1 Representation

A *Markov network* is a model for the joint probability distribution of a set of variables $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ (Della Pietra et al., 1997). It is often expressed as an undirected graph G and a set of potential functions ϕ_k. The graph has a node for each variable, and the model has a potential function for each clique in the graph. The joint distribution represented by a Markov network is:

$$P(\mathbf{X}{=}\mathbf{x}) = \frac{1}{Z} \prod_k \phi_k(\mathbf{x}_{\{k\}}) \tag{1}$$

where $\mathbf{x}_{\{k\}}$ is the state of the variables that appear in the kth clique, and Z is a normalization constant called the *partition function*.

The graph encodes the following conditional independencies: sets of variables $\mathbf{X}_A$ and $\mathbf{X}_B$ are conditionally independent given evidence $\mathbf{Y}$ if all paths between their corresponding nodes in the graph pass through nodes from $\mathbf{Y}$. Any probability distribution that can be represented as a product of potential functions over the cliques of the graph, as in Equation (1), satisfies these independencies; for positive distributions, the converse holds as well.

One of the limitations of the graph structure is that it says nothing about the structure of the potential functions themselves. The most standard representation of a potential function over discrete variables is a table with one value for each variable configuration, but this requires a number of parameters that is exponential in the size of the clique. To learn an effective probability distribution, we typically need a finer-grained parametrization that permits a compact distribution even when the cliques are relatively large.

Therefore, we focus on learning the *log-linear representation* of a Markov network, in which the clique potentials are replaced by an exponentiated weighted sum of features of the state:

$$P(\mathbf{X}{=}\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_j w_j f_j(\mathbf{x}_{\{j\}})\right).$$

A feature $f_j(\mathbf{x}_{\{j\}})$ may be any real-valued function of the state. For discrete data, a feature typically is a conjunction of tests of the form $X_i = x_i$, where X_i is a variable and x_i is a value of that variable. We say that a feature *matches* an example if it is true of that example. Any positive probability distribution over a discrete domain can be represented as log-linear model with conjunctive features. For example, a product of tabular potential functions could be converted into a log-linear model by constructing one conjunctive feature for each row of each table, using the log of the potential function value as the feature weight.

In this paper, we will refer to this set of conjunctive features as the structure of the Markov network. This detailed structure specifies not only the independencies of the distribution, but also the specific interaction terms that are most significant. If desired, the simpler undirected graph structure can be constructed from the features by adding an edge between each pair of nodes whose variables appear together in a feature.

2.2 Inference

The main inference task in graphical models is to compute the conditional probability of some variables (the query) given the values of some others (the evidence), by summing out the remaining variables. This problem is #P-complete. Thus, approximate inference techniques are required. One widely used method is Markov chain Monte Carlo (MCMC) (Gilks et al., 1996), and in particular Gibbs sampling, which proceeds by sampling each variable in turn given its *Markov blanket*, the variables it appears with in some potential or feature. These samples can be used to answer probabilistic queries by counting the number of samples that satisfy each query and dividing by the total number of samples. Under modest assumptions, the distribution represented by these samples will eventually converge to the true distribution. However, convergence may require a very large number of samples, and detecting convergence is difficult.

2.3 Weight Learning

The goal of weight learning is to select feature weights that maximize a given objective function. One of the most popular objective functions is the log-likelihood of the training data. In a Markov network, the negative log-likelihood is a convex function of the weights, and thus weight learning can be posed as a convex optimization problem. However, this optimization typically requires evaluating the log-likelihood and its gradient in each iteration. This is typically intractable to compute exactly due to the partition function. Furthermore, an approximation may work poorly: Kulesza and Pereira (2007) have shown that approximate inference can mislead weight learning algorithms.

A more computationally efficient alternative, widely used in areas such as spatial statistics, social network modeling, and language processing, is to optimize the pseudo-likelihood or pseudo-log-likelihood (PLL) instead (Besag, 1975). Pseudo-likelihood is the product of the conditional probabilities of each variable given its Markov blanket; pseudo-log-likelihood is the log of the pseudo-likelihood:

$$\log P_w^{\bullet}(\mathbf{X}=\mathbf{x}) = \sum_{j=1}^{V} \sum_{i=1}^{N} \log P_w(X_{i,j}=x_{i,j}|MB_x(X_{i,j})) \tag{2}$$

where V is the number of variables, N is the number of examples, $x_{i,j}$ is the value of the jth variable of the ith example, $MB_x(X_{i,j})$ is the state of $X_{i,j}$'s Markov blanket in the data. PLL and its gradient can be computed efficiently and optimized using any standard convex optimization algorithm, since the negative PLL of a Markov network is also convex.

3. Structure Learning in Markov Networks

Our goal in structure learning is to find a succinct set of features that can be used to accurately represent a probability distribution in a domain of interest. Other goals include learning the independencies or causal structure in the domain, but we focus on accurate probabilities. In this section, we briefly review four categories of approaches for Markov network structure learning, along with their strengths and weaknesses.

Global Search-Based Learning. One of the common approaches is to perform a global search for a set of features that accurately captures high-probability regions of the instance space (Della Pietra et al., 1997; McCallum, 2003). The algorithm of Della Pietra et al. (1997) is the most canonical example of this approach. The algorithm starts with a set of atomic features, each consisting of one state of one variable. It creates candidate features by conjoining each feature to each other feature, including the original atomic features. It calculates the weight for each candidate feature by assuming that all other feature weights remain unchanged, which is done for efficiency reasons. It uses Gibbs sampling for inference when setting the weight. Then, it evaluates each candidate feature f by estimating how much adding f would increase the log-likelihood. It adds the feature that results in the largest gain to the feature set. This procedure terminates when no candidate feature improves the model's score.

Recently, Davis and Domingos (2010) proposed an alternative bottom-up approach, called Bottom-up Learning of Markov Networks (BLM), for learning the structure of a Markov network. BLM starts by treating each complete example as a long feature in the Markov network. The algorithm repeatedly iterates through the feature set. It considers generalizing each feature to match its k nearest previously unmatched examples by dropping variables. If incorporating the newly generalized feature improves the model's score, it is retained in the model. The process terminates when no generalization improves the score.

These discrete search approaches are often slow due to the exponential number of possible features, leading to a doubly-exponential space of possible structures. Even a greedy search through this space must use the training data to repeatedly evaluate many candidates.

Optimization-Based Learning. Instead of performing a discrete search through possible structures, other recent work has framed the search as a continuous weight optimization problem with L1 regularization for sparsity (Lee et al., 2007; Schmidt and Murphy, 2010). The final structure consists of all features that are assigned non-zero weights. These methods are somewhat more efficient, but are typically limited to relatively short features. For example, in the approach of Lee et al. (2007), the set of candidate features must be specified in advance, and must be small enough that the gradient of all feature weights can be computed. Even including interactions among three variables requires a cubic number of features. Learning higher-order interactions quickly becomes infeasible. Schmidt and Murphy (2010) propose an algorithm that can learn longer features, as long as they satisfy a hierarchical constraint: longer features are only included when all subsets of the feature have been assigned non-zero weights. In experiments, this method does identify some longer features, but most features are short.

Independence Test Based Learning. Another line of work attempts to identify the Markov network structure directly by performing independence tests (Spirtes et al., 1993).

The basic idea is that if two variables are conditionally independent given some other variables then there should be no edge between them in the Markov network. Thus, instead of searching for interactions among the variables, these methods search for independencies. The challenge is the large number of conditional independencies to test: simply testing for marginal independence among each pair of variables is quadratic in the number of variables, and the complexity grows exponentially with the size of the separating set. Some variants of this approach search for the Markov blanket of each variable, the minimal set of variables that renders it conditionally independent from all others (Bromberg et al., 2009). Using independencies in the data to infer additional independencies can speed up this search, but many tests are still required. Furthermore, reliably recovering the independencies may not necessarily lead to the most accurate probabilistic model, since that is not the primary goal of these methods.

Learning Local Models. Ravikumar et al. (2010) proposed the alternative idea of learning a local model for each variable and then combining these models into a global model. Their method learns the structure by trying to discover the Markov blanket of each variable. It considers each variable X_i in turn and builds an L1-regularized logistic regression model to predict the value of X_i given the remaining variables. L1 regularization encourages sparsity, so that most of the variables end up with a weight of zero. The Markov blanket of X_i is all variables that have non-zero weight in the logistic regression model. Under certain conditions, this is a consistent estimator of the structure of a pairwise Markov network. In practice, when learned from real-world data, these Markov blankets are often incompatible with each other; for example, X_i may be in the inferred Markov blanket of X_j while the reverse does not hold. There are two methods for resolving these conflicts. One is to include an edge if either X_i is in X_j's Markov blanket or X_j is in X_i's Markov blanket. The other method is to include an edge only if X_i is in X_j's Markov blanket and X_j is in X_i's Markov blanket. In the final model, if there is an edge between X_i and X_j then the log-linear model includes a pairwise feature involving those two variables. All weights are then learned globally using any standard weight learning algorithm. While this approach greatly improves the tractability of structure learning, it is limited to modeling pairwise interactions, ignoring all higher-order effects. Furthermore, it still exhibits long run times for domains that have large numbers of variables.

4. Decision Tree Structure Learning (DTSL)

We now describe our method for learning Markov network structure from data, decision tree structure learning (DTSL). Algorithm 1 outlines our basic approach. For each variable X_i, we learn a probabilistic decision tree to represent the conditional probability of X_i given all other variables, $P(X_i|\mathbf{X} - X_i)$. Each tree is converted to a set of conjunctive features capable of representing the same probability distribution as the tree. Finally, all features are taken together in a single model and weights are learned globally using any standard weight learning algorithm.

This is similar in spirit to learning a dependency network (Heckerman et al., 2000): Both dependency networks (with tree distributions) and DTSL learn a probabilistic decision tree for each variable and combine the trees to form a probabilistic model. However, a dependency network may not represent a consistent probability distribution, and inference

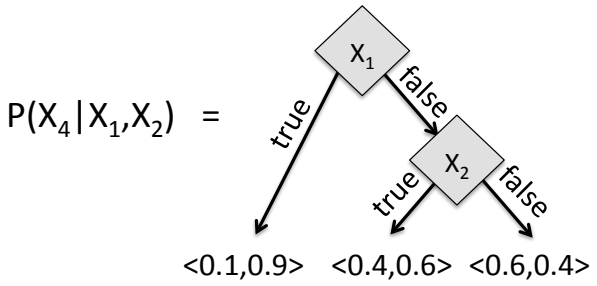


Figure 1: Example of a probabilistic decision tree.

can only be done by Gibbs sampling. In contrast, the Markov networks learned by DTSL always represent consistent probability distributions and allow inference to be done by any standard technique, such as loopy belief propagation (Murphy et al., 1999), mean field, or MCMC.

We now describe each step of DTSL in more detail.

Algorithm 1 The DTSL Algorithm

```
function DTSL(training examples D, variables X)
F ← ∅
for all X_i ∈ X do
  T_i ← LearnTree(D, X_i)
  F_i ← GenerateFeatures(T_i)
  F ← F ∪ F_i
end for
M ←LearnWeights(F, D)
return M
```

4.1 Learning Trees

A probabilistic decision tree represents a probability distribution over a target variable, X_i, given a set of inputs. Each interior node tests the value of an input variable and each of its outgoing edges is labeled with one of the outcomes of that test (e.g., true or false). Each leaf node contains the conditional distribution (e.g., multinomial) of the target variable given the test outcomes specified by its ancestor nodes and edges in the tree. We focus on discrete variables and consider tests of the form $X_j = x_j$, where X_j is a variable and x_j is value of that variable. Each conditional distribution is represented by a multinomial. Figure 1 contains an example of a probabilistic decision tree.

We can learn a probabilistic decision tree from data in a depth-first manner, one split at a time. We select a split at the root, partition the training data into the sets matching each outgoing branch, and recurse. We select each split to maximize the conditional log-likelihood of the target variable. This is very similar to using information gain as the split criterion. We used multinomials as the leaf distributions with a Dirichlet prior ($\alpha = 1$) for smoothing. In order to help avoid overfitting, we used a structure prior $P(S) \propto \kappa^p$,

where p is the number of parameters and $\kappa < 1$ represents a multiplicative penalty for each additional parameter in the model, as in Chickering et al. (1997). To further avoid overfitting, we set the minimum number of examples at each leaf to 10. Any splits that would result in fewer examples in a leaf are rejected.

Pseudocode for the tree learning subroutine is in Algorithm 2.

Algorithm 2 DTSL Tree Learning Subroutine

```
function LearnTree(training examples D, variable X_i)
best_split ← ∅
best_score ← 0
for all X_j ∈ X − X_i do
  for all x_j ∈ Val(X_j) do
    S ← (X_j = x_j)
    if Score(S, X_i, D) > best_score then
      best_split ← S
      best_score ←Score(S, X_i, D)
    end if
  end for
end for
if best_score > log κ then
  (D_t, D_f) ←SplitData(D, best_split)
  T_L ←LearnTree(D_t, X_i)
  T_R ←LearnTree(D_f, X_i)
  return new TreeVertex(best_split, T_L, T_R)
else
  Use D to estimate P(X_i)
  return new TreeLeaf(P(X_i))
end if
```

4.2 Generating Features

While decision trees are not commonly thought of as a log-linear model, any decision tree can be converted to a set of conjunctive features. In addition to a direct translation (Default), we explored four modifications (Prune, Prune-10, Prune-5, and Nonzero) which could yield structures with easier weight learning or better generalization.

The Default feature generation method is a direct translation of a probabilistic decision tree to an equivalent set of features. For each state of the target variable, we generate a feature for each path from the root to a leaf. The feature's conditions specify a single state of the target variable and all variable tests along the path in the decision tree. For example, to convert the decision tree in Figure 1 to a set of rules, we generate two features for each leaf, one where X_4 is true and one where X_4 is false. The complete list of features is as follows:

1. $X_1 = T \wedge X_4 = T$
2. $X_1 = T \wedge X_4 = F$

3. $X_1 = F \wedge X_2 = T \wedge X_4 = T$
4. $X_1 = F \wedge X_2 = T \wedge X_4 = F$
5. $X_1 = F \wedge X_2 = F \wedge X_4 = T$
6. $X_1 = F \wedge X_2 = F \wedge X_4 = F$

By using the log probability at the leaf as the rule's weight, we obtain a log linear model representing the same distribution. By applying this transformation to all decision trees, we obtain a set of conjunctive features that comprise the structure of our Markov network. However, their weights may be poorly calibrated (e.g., due to the same feature appearing in several decision trees), so weight learning is still necessary.

The Prune method expands the set of features generated by Default in order to make learning and inference easier. One disadvantage of the Default procedure is that it generates very long features with many conditions when the source trees are deep. Intuitively, we would like to capture the coarse interactions with short features and the finer interactions with longer features, rather than representing everything with long features. In the Prune method, we generate additional features for each path from the root to an interior node, not just paths from the root to a leaf. Each feature's conditions specify a single state of the target variable and all variable tests along the path in the decision tree. However, we include paths ending at any node, not just at leaves.

Specifically, for each state of the target variable and node in the tree (leaf or non-leaf), we generate a feature that specifies the state of the target variable and contains a condition for each ancestor of the node. This is equivalent to applying the Default feature generation method to all possible "pruned" versions of a decision tree, that is, where one or more interior nodes are replaced with leaves. This yields four additional rules, in addition to those enumerated above:

1. $X_4 = T$
2. $X_4 = F$
3. $X_1 = F \wedge X_4 = T$
4. $X_1 = F \wedge X_4 = F$

The Prune-10 and Prune-5 methods begin with the features generated by Prune and remove all features with more than 10 and 5 conditions, respectively. This can help avoid overfitting.

Our final feature generation method, Nonzero, is similar to Default, but removes all false variable constraints in a post-processing step. For example, the decision tree in Figure 1 would be converted to the following set of rules:

1. $X_1 = T \wedge X_4 = T$
2. $X_1 = T$
3. $X_2 = T \wedge X_4 = T$
4. $X_2 = T$
5. $X_4 = T$

This simplification is designed for sparse binary domains such as text, where a value of false or zero contains much less information than a value of true or one.

4.3 Asymptotic Complexity

Next, we explore DTSL's efficiency by analyzing its asymptotic complexity. Let n be the number of variables, m be the number of training examples, and l be the number of values per variable. The complexity of selecting the first split is $O(lmn)$, since we must compute statistics for each of the l values of each of the n variables using all of the m examples. At the next level, we now have two splits to select: one for the left child and one for the right child of the original split. However, since the split partitions the training data into two sets, each of the m examples is only considered once, either for the left split or the right split, leading to a total time of $O(lmn)$ at each level. If each split assigns a fraction of at least $1/k$ examples to each child, then the depth is at most $O(\log_k(m))$, yielding a total complexity of $O(lmn \log_k(m))$ for one tree, and $O(lmn^2 \log_k(m))$ for the entire structure. Depending on the patterns present in the data, the depth of the learned trees could be much less than $\log_k(m)$, leading to faster run times in practice. For large data sets or streaming data, we can apply the Hoeffding tree algorithm, which uses the Hoeffding bound to select decision tree splits after enough data has been seen to make a confident choice, rather than using all available data (Domingos and Hulten, 2000).

5. Decision Tree Bottom-Up Learning (DT-BLM)

The DTSL algorithm works by first using a decision tree learner to generate a set of conjunctive features and then learning weights for those features. In this section, we propose using decision trees within the context of the BLM Markov network structure learning algorithm (Davis and Domingos, 2010), which is described in Section 3.

BLM starts with a large set of long (i.e., specific) features and simplifies them to make them more general. The standard BLM algorithm uses the set of all training examples as the initial feature set. However, BLM could, in principle, generalize any set of initial features. The key idea of DT-BLM is to run BLM on the features from DTSL.

Algorithm 3 outlines the DT-BLM algorithm. DT-BLM receives a set of training examples, D, a set of variables, $\mathbf{X}$, and a set of integers, K, as input. It begins by running DTSL. Of the five feature conversion methods, it selects whichever one results in the best scoring model on validation data. Then DT-BLM employs the standard BLM learning procedure, but uses the features learned by DTSL as the initial feature set. The main loop in BLM involves repeatedly iterating through the feature set, calling the GeneralizeFeature method on each feature f in the feature set F.

The GeneralizeFeature method, outlined in Algorithm 4, proposes and scores several candidate feature generalizations. Specifically, it creates one generalization, f', for each $k \in K$ by finding the set of examples U_k which are f's k nearest unmatched examples. Next, it creates f' by dropping each variable-value test in f that does not match all examples in U_k, which has the effect of generalizing f' to match all examples in this set. It also scores the effect of removing f from F.

DT-BLM measures the distance between a feature and an example using the generalized value difference metric (GVDM), which tends to perform better in practice than the simpler

Hamming distance (Davis and Domingos, 2010). Formally, the distance, $D(f, e)$ is:

$$D(f, e) = \sum_{c \in f} GVDM(f, e, c)$$

where f is a feature, e is an example, c ranges over the variables in f and

$$GVDM(f, e, c) = \sum_{h} \sum_{f_i \in f, f_i \neq c} |P(c = h|f_i) - P(c = h|e_{f_i})|^Q$$

where h ranges over the possible values of variable c, f_i is the value of the ith variable in f, e_{f_i} is the value of the attribute referenced by f_i in e, and Q is an integer. For a variable c that appears in f, GVDM measures how well the other variables in f predict c. The intuition is that if c appears in a feature then the other variables should be good predictors of c.

Each generalization is evaluated by replacing f with the generalization in the model, relearning the weights of the modified feature set, and scoring the new model as:

$$S(D, F', \alpha) = PLL(D, F') - \alpha \sum_{f_i \in F'} |f_i|$$

where $PLL(D, F')$ is the train set pseudo-likelihood of feature set F', α is a penalty term to avoid overfitting, and $|f_i|$ is the number of variable-value tests in feature f_i. The procedure returns the best scoring generalized feature set, F'. DT-BLM updates F to F' if F' has a better score. The process terminates after making one full loop over the feature set without changing F by accepting a generalization.

The advantage of DT-BLM over DTSL is that it can refine individual features based on their global contribution to the pseudo-likelihood. This can lead to simpler models in terms of both the number and length of the features. One advantage of DT-BLM over BLM is that the features selected by DTSL are already very effective, so BLM is less likely to end up in a bad local optimum. A second advantage is that it removes the restriction that BLM can never learn a model that has more features than examples. This is valuable for domains which are most effectively modeled with a large number of short features (e.g., text). The principle disadvantage of DT-BLM is speed: it can be much slower than DTSL, since it is doing a secondary search over feature simplifications. We have also found that DT-BLM is sensitive to the Gaussian weight prior used during the BLM structure search, unlike the standard BLM algorithm. This means that DT-BLM requires more tuning time than BLM, as discussed more extensively in our empirical evaluation in Section 7.

Note that DT-BLM is similar in spirit to Bayesian network structure learning algorithms that combine independence-test and search-based learning techniques (Tsamardinos et al., 2006). These algorithms work in two phases. In the first step, they identify a superset of the edges that could be included in the network using independence tests. In the second step, a search through the space of possible structures is performed, but it is restricted to only consider including candidate edges identified in the first step. Typically, a greedy, general-to-specific search is employed. These algorithms differ from DT-BLM in three key ways: DT-BLM searches for features and not edges; DT-BLM uses decision trees to identify candidate features and not independence tests; and DT-BLM uses a specific-to-general search and not a general-to-specific search to refine the structure.

Algorithm 3 The DT-BLM Algorithm

function DT-BLM(training examples D, variables $\boldsymbol{X}$, K)
$F \leftarrow \emptyset$
for all $X_i \in \boldsymbol{X}$ **do**
  $T_i \leftarrow$ LearnTree(X_i, D)
  $F_i \leftarrow$ GenerateFeatures(T_i)
  $F \leftarrow F \cup F_i$
end for
repeat
  for all features $f \in F$ **do**
    $F' \leftarrow$ GeneralizeFeature(D, F, f, K)
    if Score$(TS, F') >$ Score(TS, F) **then**
      $F \leftarrow F'$
    end if
  end for
until no generalization improves the score
return M

Algorithm 4 Feature generalization subroutine used by DT-BLM.

function GeneralizeFeature(training examples D, feature set F, feature f, K)
$F_{best} = F$
for all $k \in K$ **do**
  $U_k = k$ nearest examples to f that do not match f
  $f' = f$ excluding each test in f that does not match all examples in U_k.
  $F' \leftarrow F$ with f replaced by f'
  if Score$(D, F') >$ Score(D, F_{best}) **then**
    $F_{best} \leftarrow F'$
  end if
end for
$F' = F$ without f
if Score$(D, F') >$ Score(D, F_{best}) **then**
  $F_{best} \leftarrow F'$
end if
return F_{best}

6. Combining DTSL and L1 (DT+L1)

In this section we propose DT+L1, a very simple way to combine DTSL and Ravikumar et al.'s L1 algorithm. DT+L1 works as follows. First, DTSL and L1 are run normally. That is, each method is run to completion (i.e., learning both the features and weights) to find the best model. Second, DT+L1 takes the union of the best DTSL feature set and the best L1 feature set. Third, DT+L1 learns the weights for the combined feature set and returns this as the final model.

The advantage of this approach is that it combines the strengths of both algorithms. DTSL excels at learning long features that capture complex interactions. Ravikumar et al.'s L1 approach only learns pairwise features, which occur less frequently in DTSL's learned models. One disadvantage is that DT+L1 will be more time intensive than either approach independently. DT+L1 involves performing parameter tuning to select the best model for both DTSL and L1 and then another run of weight learning, including parameter tuning, on the combined feature set. Another potential problem is that the combined model will have more features, which may lead to a more complex inference task.

7. Empirical Evaluation

We evaluate our algorithms on 20 real-world data sets. The goals of our experiments are three-fold. First, we want to determine how the different feature generation methods affect the performance of DTSL and DT-BLM (Section 7.3). Second, we want to compare the accuracy of DTSL, DT-BLM, and DT+L1 to each other as well as to several state-of-the-art Markov network structure learners: the algorithm of Della Pietra et al. (1997), which we refer to as DP; BLM (Davis and Domingos, 2010); and L1-regularized logistic regression (Ravikumar et al., 2010) (Section 7.4). Finally, we want to compare the running time of these learning algorithms, since this greatly affects their practical utility (Section 7.5).

7.1 Methodology

We used DTSL, DT-BLM, DT+L1, and each of the baselines to learn structures on 20 data sets.

DTSL was implemented in OCaml. For both BLM and DP, we used the publicly available code of Davis and Domingos (2010). Since DT-BLM is built on both DTSL and BLM, it used a combination of the DTSL and BLM code as well. For Ravikumar et al.'s approach, we tried both the OWL-QN (Andrew and Gao, 2007) and LIBLINEAR (Fan et al., 2008) software packages. Our initial experiments and evaluation were done using OWL-QN, but we later discovered that LIBLINEAR was much faster with nearly identical accuracy. Therefore, to be as fair as possible to L1, we report the running times for LIBLINEAR.

The output of each structure learning algorithm is a set of conjunctive features. To learn weights, we optimized the pseudo-likelihood of the data via the limited-memory BFGS algorithm (Liu and Nocedal, 1989) since optimizing the likelihood of the data is prohibitively expensive for the domains we consider.

Like Lee et al. (2007), we evaluated our algorithm using test set conditional marginal log-likelihood (CMLL). To make results from different data sets more comparable, we report normalized CMLL (NCMLL), which is CMLL divided by the number of variables in the domain. Calculating the NCMLL requires dividing the variables into a query set Q and an evidence set E. Then, for each test example we computed:

$$\text{NCMLL}(X = x) = \frac{1}{|X|} \sum_{i \in Q} \log P(X_i = x_i | E).$$

For each domain, we divided the variables into four disjoint sets. One set served as the query variables while the remaining three sets served as evidence. We repeated this procedure so

that each set served as the query variables once. We computed the conditional marginal probabilities using Gibbs sampling, as implemented in the open-source Libra toolkit.[1] For all domains, we ran 10 independent chains, each with 100 burn-in samples and followed by 1,000 samples for computing the probability. CMLL is related to PLL (Equation 2), since both measure the ability of the model to predict individual variables given evidence. However, we used approximately 75% of the variables as evidence when computing CMLL, while PLL always uses all-but-one variable as evidence.

We tuned all algorithms using separate validation sets, the same validation sets used by Van Haaren and Davis (2012). For DTSL, we selected the structure prior κ for each domain that maximized the total log-likelihood of all probabilistic decision trees on the validation set. The values of κ we used were powers of 10, ranging from 0.0001 to 1.0. When learning the weights for each feature generation method, we placed a Gaussian prior with mean 0 on each feature weight and then tuned the standard deviation to maximize PLL on the validation set, with values of 100, 10, 1, and 0.1. For comparisons to other algorithms, we selected the DTSL model with the best pseudo-likelihood on the validation set. We chose to use pseudo-likelihood for tuning instead of CMLL because it is much more efficient to compute.

For L1, on each data set we tried the following values of the LIBLINEAR tuning parameter C: 0.001, 0.01, 0.05, 0.1, 0.5, 1 and 5.[2] We also tried both methods of making the Markov blankets consistent. These parameter settings allowed us to explore a variety of different models, ranging from those containing all pairwise interactions to those that were very sparse. We also tuned the weight prior as we did with DTSL. Tuning the standard deviation of the Gaussian weight prior allowed us to get better results than reported by Davis and Domingos (2010).

For BLM and DP, we kept the tuning settings used by Davis and Domingos (2010). We tried performing additional tuning of the weight prior for BLM, but it did not lead to improved results. For DT+L1, we combined the DTSL and L1 structures and relearned the final weights, including tuning the Gaussian weight prior on the validation set.

All of our code is available at `http://ix.cs.uoregon.edu/~lowd/dtsl` under a modified BSD license.

7.2 Data Sets

For our experiments, we used the same set of 20 domains as Van Haaren and Davis (2012), 13 of which were previously used by Davis and Domingos (2010).[3] All variables are binary-valued. Basic statistics for all data sets are in Table 1, ordered by number of variables in the domain. "Density" refers to the fraction of non-zero entries. Below, we provide additional information about each data set.

1. Libra is available from `http://libra.cs.uoregon.edu/`.
2. C is the inverse of the L1 regularization weight λ used by OWL-QN (Andrew and Gao, 2007). Larger C values almost always resulted in generating all pairwise features.
3. These data sets are publicly available at `http://alchemy.cs.washington.edu/papers/davis10a`.

Data Set	# Train Ex.	# Tune Ex.	# Test Ex.	# Vars	Density
1. NLTCS	16,181	2,157	3,236	16	0.332
2. MSNBC	291,326	38,843	58,265	17	0.166
3. KDDCup 2000	180,092	19,907	34,955	64	0.008
4. Plants	17,412	2,321	3,482	69	0.180
5. Audio	15,000	2,000	3,000	100	0.199
6. Jester	9,000	1,000	4,116	100	0.608
7. Netflix	15,000	2,000	3,000	100	0.541
8. Accidents	12,758	1,700	2,551	111	0.291
9. Retail	22,041	2,938	4,408	135	0.024
10. Pumsb Star	12,262	1,635	2,452	163	0.270
11. DNA	1,600	400	1,186	180	0.253
12. Kosarek	33,375	4,450	6,675	190	0.020
13. MSWeb	29,441	3,270	5,000	294	0.010
14. Book	8,700	1,159	1,739	500	0.016
15. EachMovie	4,524	1,002	591	500	0.059
16. WebKB	2,803	558	838	839	0.064
17. Reuters-52	6,532	1,028	1,540	889	0.036
18. 20 Newsgroups	11,293	3,764	3,764	910	0.049
19. BBC	1,670	225	330	1,058	0.078
20. Ad	2,461	327	491	1,556	0.008

Table 1: Data set characteristics.

We used four clickstream prediction domains: KDDCup 2000, MSNBC, Anonymous MSWeb,[4] and Kosarek.[5] Each data point was a single session, with one binary-valued variable for each page, area, or category of the site, indicating if it was visited during that session or not. For KDD Cup 2000 (Kohavi et al., 2000), we used the subset of Hulten and Domingos (2002), which consisted of 65 page categories. We dropped one category that was never visited in the training data. The MSNBC anonymous web data contains information about which top-level MSNBC pages were visited during a single session. The MSWeb anonymous web data contains visit data for 294 areas (Vroots) of the Microsoft web site, collected during one week in February 1998. Kosarek is clickstream data from a Hungarian online news portal.

Five of our domains were from recommender systems: Audio, Book, EachMovie, Jester and Netflix. The Audio data set consists of information about how often a user listened to a particular artist.[6] The data was provided by the company Audioscrobbler before it was acquired by Last.fm. We focused on the 100 most listened-to artists. We used a random subset of the data and reduced the problem to "listened to" or "did not listen to." The

4. KDDCup 2000, MSNBC, and Anonymous MSWeb, are available from the UCI machine learning repository (Blake and Merz, 2000).
5. The Kosarek, Pumsb Star, Accidents, and Retail data sets are available at `http://fimi.ua.ac.be/data/`.
6. The Audio data set is available at `http://www-etud.iro.umontreal.ca/~bergstrj/audioscrobbler_data.html`.

Book Crossing (Book) data set (Ziegler et al., 2005) consists of a user's rating of how much they liked a book. We considered the 500 most frequently rated books. We reduced the problem to "rated" or "not rated" and considered all people who rated at least two of these books. EachMovie[7] is a collaborative filtering data set in which users rate movies they have seen. We focused on the 500 most-rated movies, and reduced each variable to "rated" or "not rated". The Jester data set (Goldberg et al., 2001) consists of users' real-valued ratings for 100 jokes. For Jester, we selected all users who had rated all 100 jokes, and reduced their preferences to "like" and "dislike" by thresholding the real-valued preference ratings at zero. Finally, we considered a random subset of the Netflix challenge data and focused on the 100 most frequently rated movies. We reduced the problem to "rated" or "not rated."

We used four text domains: 20 Newsgroups, Reuters-52, WebKB,[8] and BBC.[9] For 20 Newsgroups, we only considered words that appeared in at least 200 documents. For Reuters, WebKB, and BBC, we only considered words that appeared in at least 50 documents. For all four data sets, we created one binary feature for each word. The text domains contained roughly a 50-50 train-test split, whereas all other domains used around 75% of the data for the training, 10% for tuning, and 15% for testing. Thus we split the test set of these domains to make the proportion of data devoted to each task more closely match the other domains used in the empirical evaluation.

The remaining seven data sets have no unifying theme. Plants consists of different plant types and locations where they are found.[4] We constructed one binary feature for each location, which is true if the plant is found there. DNA[10] is DNA sequences for primate splice-junctions; we used the binary-valued encoding provided. The National Long Term Care Survey (NLTCS) data consist of binary variables that measure an individual's ability to perform different daily living activities.[11] Pumsb Star contains census data for population and housing.[5] Accidents contains anonymized traffic incident data.[5] Retail is market basket data from a Belgian retail store.[5]

7.3 Feature Generation Methods

First, we compared the accuracy of DTSL with different feature generation methods: Default, Prune, Prune-10, Prune-5, and Nonzero. Table 2 lists the NCMLL of each method on each data set. For each data set, the method with the best NCMLL on the test set is in bold, and the method with the best PLL on the validation set is underlined. The results are shown graphically in the top half of Figure 2. The data sets are shown in the same order as in Table 1. Each bar represents the *negative* NCMLL of DTSL with one feature generation method on one data set. Lower is better. To make the differences easier to see, we subtracted the negative NCMLL for Default from each bar, so that positive values (above the x-axis) are worse than Default and negative values (below the x-axis) are better than Default.

7. The EachMovie data set was provided by Compaq at `http://research.compaq.com/SRC/eachmovie/`; as of October 2004, it is no longer available for download.
8. 20 Newsgroups and Reuters-52 are available at `http://web.ist.utl.pt/~acardoso/datasets/`.
9. The BBC data set is available at `http://mlg.ucd.ie/datasets/bbc.html`.
10. The DNA data set is available at `http://www.cs.sfu.ca/~wangk/ucidata/dataset/DNA/`.
11. NLTCS is available at `http://lib.stat.cmu.edu/datasets/`.

	DTSL					DT-BLM				
Data Set	Default	Nonzero	Prune	Prune-10	Prune-5	Default	Nonzero	Prune	Prune-10	Prune-5
NLTCS	-0.328	-0.326	-0.326	**-0.325**	-0.326	**-0.324**	-0.326	**-0.324**	**-0.324**	-0.325
MSNBC	**-0.336**	-0.344	**-0.336**	-0.340	-0.358	**-0.336**	-0.344	**-0.336**	-0.339	-0.357
KDDCup 2000	**-0.032**	-0.033	**-0.032**	**-0.032**	**-0.032**	**-0.032**	-0.033	**-0.032**	**-0.032**	**-0.032**
Plants	-0.146	-0.148	**-0.143**	-0.145	-0.153	-0.142	-0.148	**-0.141**	-0.144	-0.153
Audio	-0.380	**-0.375**	-0.379	-0.379	-0.384	-0.375	**-0.373**	-0.375	-0.375	-0.380
Jester	-0.512	**-0.505**	-0.511	-0.511	-0.514	-0.507	**-0.503**	-0.507	-0.507	-0.509
Netflix	-0.543	**-0.532**	-0.541	-0.541	-0.545	-0.540	**-0.532**	-0.539	-0.539	-0.542
Accidents	-0.150	**-0.147**	-0.150	-0.150	-0.169	-0.148	-0.146	-0.146	**-0.145**	-0.167
Retail	**-0.078**	-0.079	-0.079	**-0.078**	-0.079	**-0.078**	-0.079	**-0.078**	**-0.078**	-0.079
Pumsb Star	-0.104	**-0.098**	-0.101	-0.102	-0.108	-0.105	**-0.099**	-0.100	**-0.099**	-0.110
DNA	**-0.384**	-0.385	**-0.384**	**-0.384**	**-0.384**	-0.384	-0.385	**-0.383**	**-0.383**	-0.384
Kosarek	**-0.053**	**-0.053**	**-0.053**	**-0.053**	**-0.053**	**-0.053**	**-0.053**	**-0.053**	**-0.053**	**-0.053**
MSWeb	**-0.029**	-0.030	**-0.029**	-0.030	-0.030	**-0.029**	-0.030	**-0.029**	**-0.029**	-0.030
Book	**-0.069**	-0.070	**-0.069**	**-0.069**	**-0.069**	**-0.068**	-0.069	**-0.068**	**-0.068**	-0.069
EachMovie	-0.109	-0.104	**-0.102**	**-0.102**	-0.105	-0.101	-0.102	**-0.100**	**-0.100**	-0.102
WebKB	**-0.179**	**-0.179**	**-0.179**	**-0.179**	-0.180	-0.178	-0.177	**-0.177**	**-0.177**	-0.178
Reuters-52	**-0.092**	**-0.092**	**-0.092**	-0.094	-0.093	-0.093	-0.092	**-0.091**	**-0.091**	-0.092
20 Newsgroups	-0.170	-0.169	**-0.166**	**-0.166**	-0.167	**-0.163**	-0.165	-0.165	-0.165	-0.165
BBC	-0.238	**-0.237**	-0.240	-0.240	-0.241	**-0.237**	**-0.237**	**-0.237**	**-0.237**	**-0.237**
Ad	**-0.012**	-0.144	-0.016	-0.016	-0.017	**-0.016**	-0.151	-0.020	-0.019	-0.022

Table 2: NCMLL of DTSL (left) and DT-BLM (right) with different conversion methods. For each algorithm and data set, the method with the best test set NCMLL is in bold and the method with the best validation set PLL is underlined.

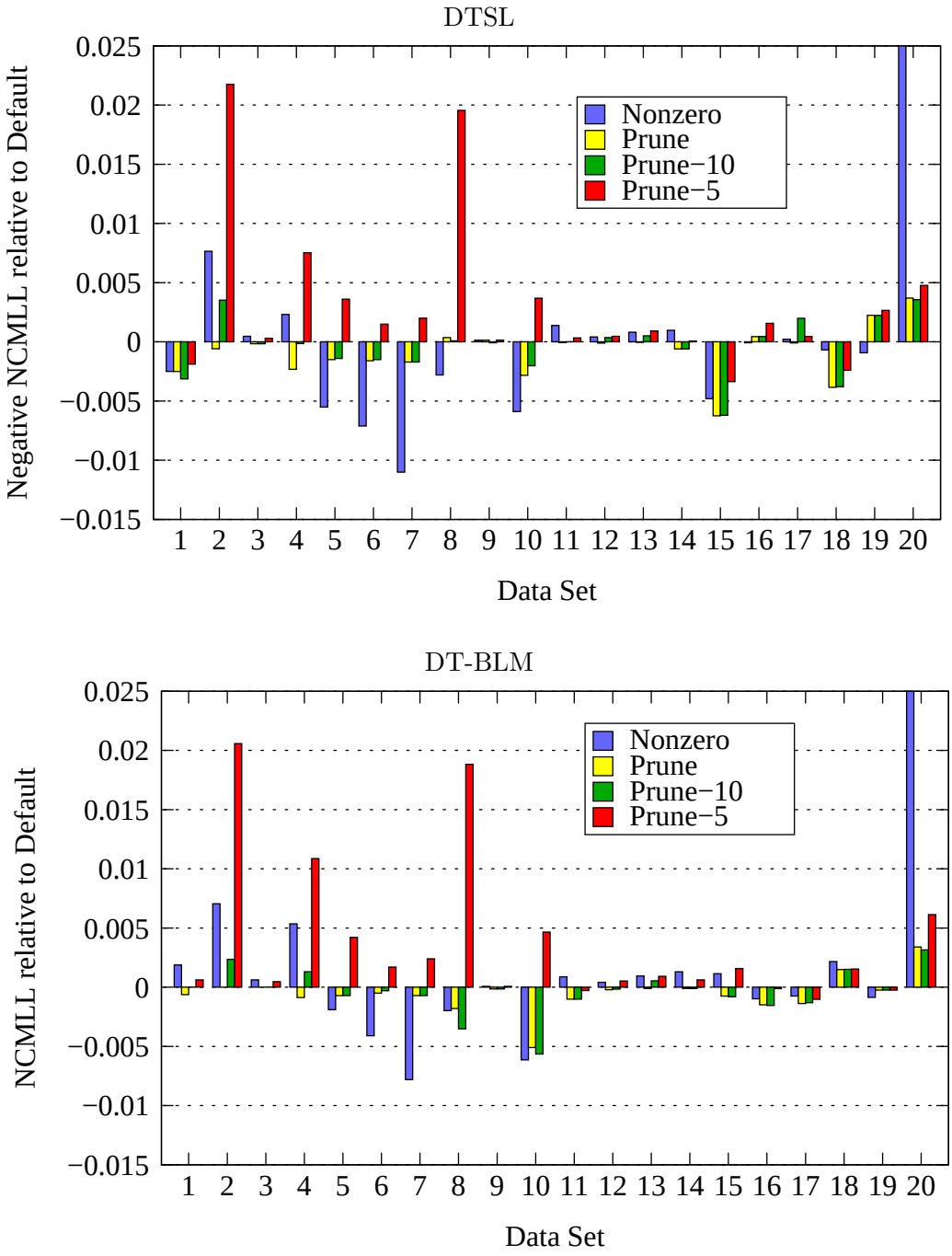


Figure 2: Performance of different feature conversion methods with DTSL (top) and DT-BLM (bottom), relative to Default. Lower values indicate better performance. Positive values (above the x-axis) indicate methods that performed worse than Default, and negative values (below the x-axis) indicate methods that performed better than Default.

For DTSL, Prune is more accurate than Default on 15 data sets. Prune-10 rarely improved on the accuracy of Prune and Prune-5 often did worse. Nonzero was the most accurate method on seven data sets for DTSL. Overall, Prune did better on more data sets, but Nonzero worked especially well on Audio, Jester, and Netflix, three relatively dense collaborative filtering data sets. When we investigated these data sets further, we found that Default, Prune, and Prune-10 were overfitting, since they obtained better PLLs than Nonzero on the training data but worse PLLs on the validation data. Prune-5 was underfitting, obtaining worse PLLs than Nonzero on both training and validation data. We hypothesize that Nonzero provides beneficial regularization by removing many features. Long features are more likely to have one or more false variable constraints, and are therefore more likely to be removed by Nonzero. If these longer features are the source of the overfitting problems, then placing a stricter prior on the weights of longer features might offer a similar benefit.

The results on DT-BLM are similar, as shown in the right side of Table 2 and the bottom half of Figure 2. For DT-BLM, Prune is more accurate on 18 data sets, although many of these differences are very small. Nonzero is most accurate on five data sets, but Prune is relatively close on three of them. On average, the additional feature refinement done by DT-BLM seems to render it somewhat less sensitive to the choice of feature generation method.

Our tuning procedure uses the PLL of the validation set for model selection. Thus, the model we select may be different from the one with the best NCMLL on the test set, since it is selected according to a different metric on different evaluation data. For both DTSL and DT-BLM, the method selected with the validation set (underlined in Table 2) is often the same as the one with the best NCMLL (bold in Table 2). When they are different, the NCMLL of the alternative model is very close. This suggests that PLL does a reasonably good job of model selection for DTSL and DT-BLM on these data sets.

For DTSL, additional characteristics of the features generated by each method are shown in Figures 3 and 4. "Average feature length" is the average number of conditions per feature. The Prune method leads to roughly twice as many features as Default, which is what one would expect, since half of the nodes in a balanced binary tree are leaves and the other half are interior nodes. Nonzero typically yields the shortest and the fewest rules, as expected.

7.4 Accuracy

We then compared DTSL, DT-BLM, and DT+L1 to three standard Markov network structure learners: L1-regularized logistic regression (Ravikumar et al., 2010), BLM (Davis and Domingos, 2010), and DP (Della Pietra et al., 1997). For DTSL and DT-BLM, we used the feature generation method that performed best on the validation set. In some cases, such as KDDCup 2000, this was not the method that performed best on the test data.

Figure 5 shows how DTSL, DT-BLM, DT+L1, and the three baselines compare in terms of NCMLL. For each data set, bars above the x-axis indicate algorithms that perform worse than DTSL, and bars below the x-axis indicate algorithms that perform better. Raw numbers for NCMLL and NPLL can be found in Table 3. NPLL is the pseudo-log-likelihood divided by the number of variables in the domain. The NPLL results are qualitatively similar to the NCMLL results, except for Pumsb Star and DNA, where L1 ranks better according

	Test set CMLL						Test set NPLL					
Data Set	DP	BLM	L1	DTSL	DT-BLM	DT+L1	DP	BLM	L1	DTSL	DT-BLM	DT+L1
NLTCS	-0.326	-0.328	-0.327	-0.325	**-0.324**	-0.325	-0.307	-0.311	-0.309	-0.309	**-0.307**	-0.308
MSNBC	-0.349	-0.344	-0.369	-0.337	**-0.336**	**-0.336**	-0.299	-0.288	-0.356	**-0.252**	**-0.252**	**-0.252**
KDDCup 2000	-0.033	**-0.032**	-0.033	**-0.032**	**-0.032**	**-0.032**	-0.034	-0.032	-0.032	-0.032	**-0.031**	**-0.031**
Plants	-0.159	-0.151	-0.156	-0.143	**-0.141**	**-0.141**	-0.135	-0.133	-0.136	-0.124	**-0.122**	**-0.122**
Audio	-0.392	-0.375	**-0.370**	-0.375	-0.373	**-0.370**	-0.385	-0.368	**-0.362**	-0.370	-0.367	-0.366
Jester	-0.537	-0.530	**-0.496**	-0.505	-0.503	-0.504	-0.528	-0.526	**-0.488**	-0.498	-0.497	-0.492
Netflix	-0.574	-0.565	**-0.523**	-0.532	-0.532	-0.525	-0.561	-0.560	**-0.511**	-0.523	-0.524	-0.514
Accidents	-0.270	-0.339	-0.149	-0.147	-0.146	**-0.141**	-0.240	-0.296	-0.112	**-0.105**	-0.106	**-0.105**
Retail	-0.079	-0.079	-0.079	**-0.078**	**-0.078**	**-0.078**	-0.075	-0.077	**-0.076**	**-0.076**	**-0.076**	**-0.076**
Pumsb Star	-0.183	-0.623	-0.085	-0.101	-0.100	**-0.084**	-0.145	-0.176	**-0.059**	**-0.059**	**-0.059**	**-0.059**
DNA	-0.541	-0.554	**-0.384**	**-0.384**	**-0.384**	**-0.384**	-0.523	-0.545	-0.326	**-0.323**	**-0.323**	**-0.323**
Kosarek	-0.057	-0.054	-0.054	-0.053	-0.053	**-0.052**	-0.056	-0.053	-0.052	-0.052	**-0.051**	**-0.051**
MSWeb	-0.031	-0.030	-0.030	**-0.029**	**-0.029**	**-0.029**	-0.030	-0.029	-0.030	**-0.027**	**-0.027**	**-0.027**
Book	-0.078	-0.069	-0.073	-0.069	**-0.068**	**-0.068**	-0.076	-0.068	-0.073	-0.068	**-0.067**	**-0.067**
EachMovie	-0.134	-0.117	-0.104	-0.102	**-0.100**	**-0.100**	-0.132	-0.116	-0.101	-0.102	**-0.099**	**-0.099**
WebKB	-0.210	-0.196	-0.179	-0.179	-0.177	**-0.176**	-0.201	-0.193	-0.175	-0.175	-0.173	**-0.172**
Reuters-52	-0.119	-0.102	**-0.091**	-0.092	-0.092	**-0.091**	-0.130	-0.101	-0.090	-0.091	**-0.089**	-0.090
20 Newsgroups	-0.188	-0.176	**-0.165**	-0.169	**-0.165**	-0.166	-0.172	-0.175	**-0.163**	-0.167	**-0.163**	-0.166
BBC	-0.258	-0.251	-0.245	**-0.237**	**-0.237**	-0.240	-0.250	-0.247	-0.246	-0.236	**-0.235**	-0.241
Ad	-0.033	-0.025	**-0.006**	-0.017	-0.022	**-0.006**	-0.033	-0.008	**-0.004**	-0.008	-0.008	**-0.004**

Table 3: Test set NCMLL and NPLL for all algorithms. The DTSL feature generation method was selected using the validation set. The best result for each metric is shown in bold. The method with the best validation set PLL is underlined.

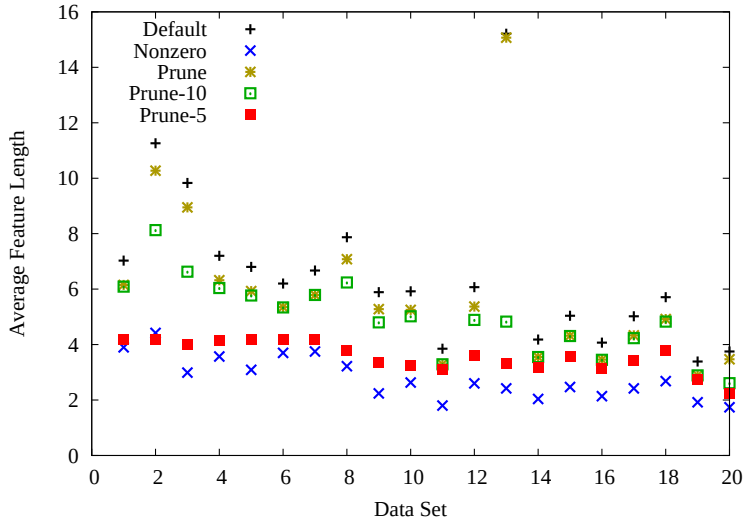


Figure 3: Average feature length for each DTSL feature generation method on each data set.

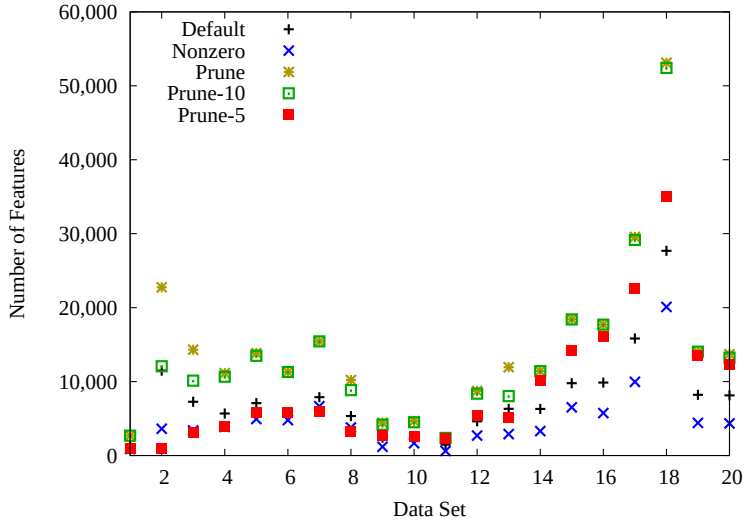


Figure 4: Number of features for each DTSL feature generation method on each data set.

to NCMLL than NPLL, and 20 Newsgroups and BBC, where BLM ranks worse according to NCMLL than NPLL. Table 3 also shows which model has the best validation set NPLL. Note that in 19 out of the 20 data sets this corresponds to the model with the best NCMLL, indicating that PLL is a good objective function to optimize for this evaluation metric. We focus our subsequent discussion on the NCMLL results, which we believe to be a better measure of model accuracy than NPLL for typical queries.

Overall, DP and BLM are fairly inaccurate, DTSL and L1 are roughly comparable, and DT-BLM is slightly better than DTSL. DT+L1 usually does at least as well as both DTSL and L1, making it the most reliably accurate algorithm overall. DTSL is always

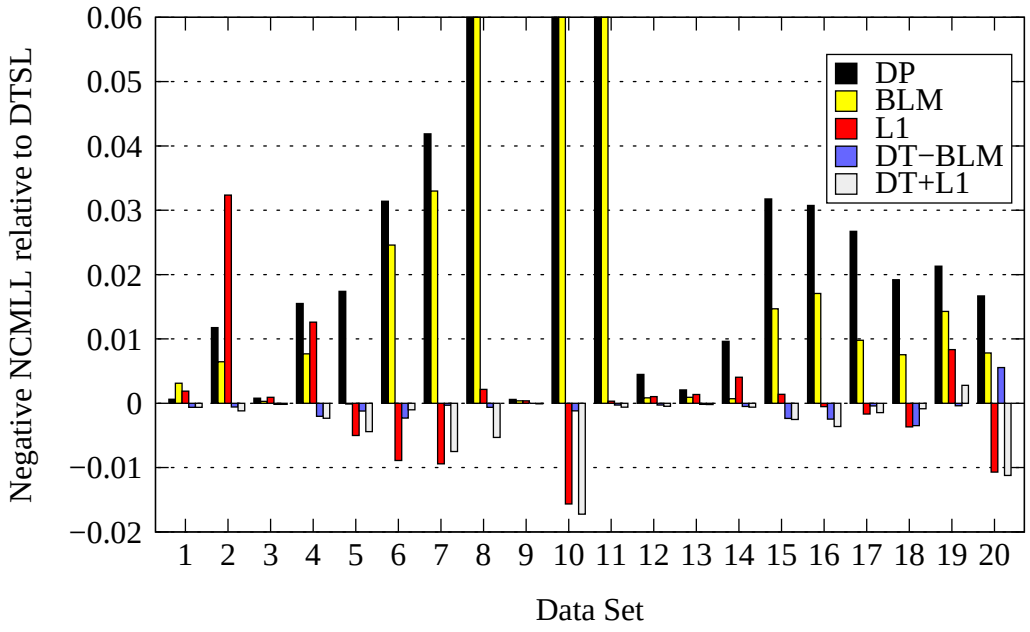


Figure 5: Normalized negative CMLL, relative to DTSL. Lower values indicate better performance. Positive values (above the x-axis) indicate methods that performed worse than DTSL, and negative values (below the x-axis) indicate methods that performed better than DTSL.

more accurate than DP and BLM, except for three data sets where it is tied with BLM. DTSL is significantly more accurate than both DP and BLM ($p < 0.001$) according to a Wilcoxon signed-ranks test in which the test set NCMLL of each data set appears as one sample in the significance test. DT-BLM represents a modest improvement in accuracy over DTSL, performing slightly better than DTSL on 11 data sets and worse on only one. On the remaining eight data sets, the difference in NCMLL was less than 0.001. A Wilcoxon signed-ranks test indicates that DT-BLM is significantly more accurate than DTSL ($p < 0.05$). Since DT+L1 includes the features from both DTSL and L1, it usually does at least as well as both methods, and sometimes better. DT+L1 is the most accurate method (including ties) on 15 out of 20 data sets, while DT-BLM is one of the most accurate methods on only 11 data sets, DTSL on five, and L1 on seven. DT+L1 is more accurate than both L1 and DTSL ($p < 0.05$) according to a Wilcoxon signed-ranks test.

Comparisons between DTSL or DT-BLM and L1 are interesting because they demonstrate the relative strengths of using trees versus logistic regression for generating features. DTSL performs better than L1 on 11 data sets and worse on seven. Similarly, DT-BLM performs better than L1 on 12 data sets and worse on six. These differences are not significant according to a Wilcoxon signed-ranks test. The relative performance of DTSL and L1 seems to vary greatly from data set to data set. To better understand what makes DTSL perform better or worse than L1, we examined the average length of the features learned by DTSL. For the domains where DTSL performs worse than L1, the average feature length

is 3.22, indicating relatively shallow trees with simple features. For the domains where DTSL performs better, the average feature length is 6.04. This supports the hypothesis that DTSL does better on domains with higher-order interactions that can be discovered by decision trees, while L1 does better on domains with many low-order interactions that can be modeled as pairwise features. DT-BLM's features show a similar trend. Figure 6 shows the average feature length for each algorithm on each data set.

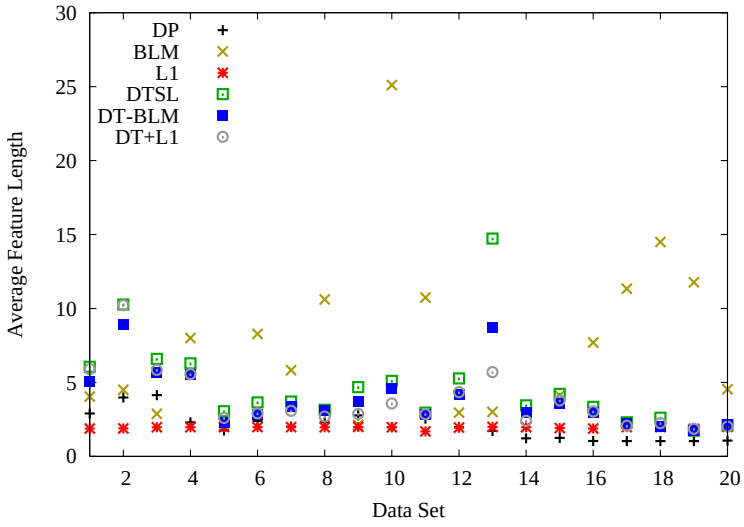


Figure 6: Average feature length for each algorithm on each data set.

We also examined the number of features learned by each algorithm (see Figure 7). When DTSL learned many more features than L1, it typically did better than L1 (NLTCS, MSNBC, KDDCup, Plants, Kosarek, EachMovie) or the same (DNA, WebKB). However, when L1 learned many more features than DTSL, it sometimes did better (Reuters-52, Ad) and sometimes did worse (Retail, MSWeb, Book). Thus, the number of features learned does not appear to correlate strongly with the relative performance of these two algorithms.

Figures 8 and 9 contain learning curves comparing DTSL and DT-BLM to L1. For the most part, all three algorithms exhibit a similar dependence on the amount of training data. The exceptions to this are Pumsb Star and Ad, where DTSL shows signs of overfitting. The relative performance of L1 goes up and down somewhat in NLTCS, DNA, EachMovie, and 20 Newsgroups, but in most data sets the ranking of the methods is stable across all amounts of training data. For some data sets (MSNBC, KDDCup 2000, Plants), L1 appears to have converged to a different asymptotic error rate than DTSL, due to its different learning bias. This is consistent with the hypothesis that some data sets are simply better suited to the learning bias of L1, and some are better suited to the longer conjunctive features representable by trees.

7.5 Learning Time

A comparison of running times is shown in Figure 10 (excluding DP), with raw numbers in Table 4. The timing results shown include parameter tuning. L1 was fastest on nine data sets; DTSL was fastest on 10; and BLM was fastest on one data set (Ad). Results excluding

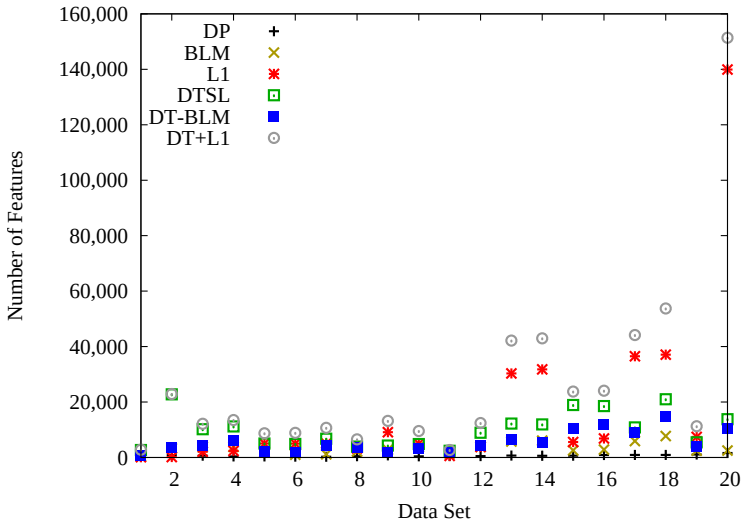


Figure 7: Number of features for each algorithm on each data set.

tuning time are similar—L1 is fastest on six data sets; DTSL on 13; and BLM on one. For DT+L1, we report the total time required for learning both the DTSL and L1 structures, as well as the additional time required for weight learning.

We use the geometric mean running time of each algorithm to summarize performance over all data sets. On average, DTSL is 5% slower than L1, 4.6 times faster than BLM, and 21.7 times faster than DP. DTSL is significantly faster than both BLM and DP according to a Wilcoxon signed-ranks test on the log of the training time ($p < 0.05$). Although the geometric mean running time of DTSL is slightly worse than L1's, its arithmetic mean is slightly better, mainly because DTSL tends to be faster on the larger, slower data sets such as 20 Newsgroups and BBC.

DT-BLM is 19.7 times slower than DTSL and 10.6 times slower than BLM; these differences are also significant under the same test ($p < 0.01$). DT-BLM is slower than BLM for two reasons. First, some data sets have more DTSL features than examples in the original training data. For example, BBC has only 1,670 examples but results in 5,475 features in DTSL's model. Since DT-BLM runs on DTSL features instead of the original training data, more features means a longer running time. Second, DT-BLM is more sensitive to the width of the Gaussian prior than BLM is, so DT-BLM has one extra parameter to tune. If we instead use the best Gaussian prior width from DTSL, the algorithm runs roughly four times faster, but remains slower than BLM.

Table 5 shows a division of the total learning time, including tuning, divided into the time spent learning the structure (i.e., the features) and the time spent on weight learning. With respect to the number of variables, DTSL has better scaling characteristics for feature generation than Ravikumar et al.'s L1 approach. The number of variables seems to have a greater effect on run time than the number of examples. Each additional variable requires learning one extra model. Furthermore, each individual learning task is more complex because the target variable can depend on one additional input variable. The most striking observation is that the majority of time is spent learning the feature weights. For L1, on

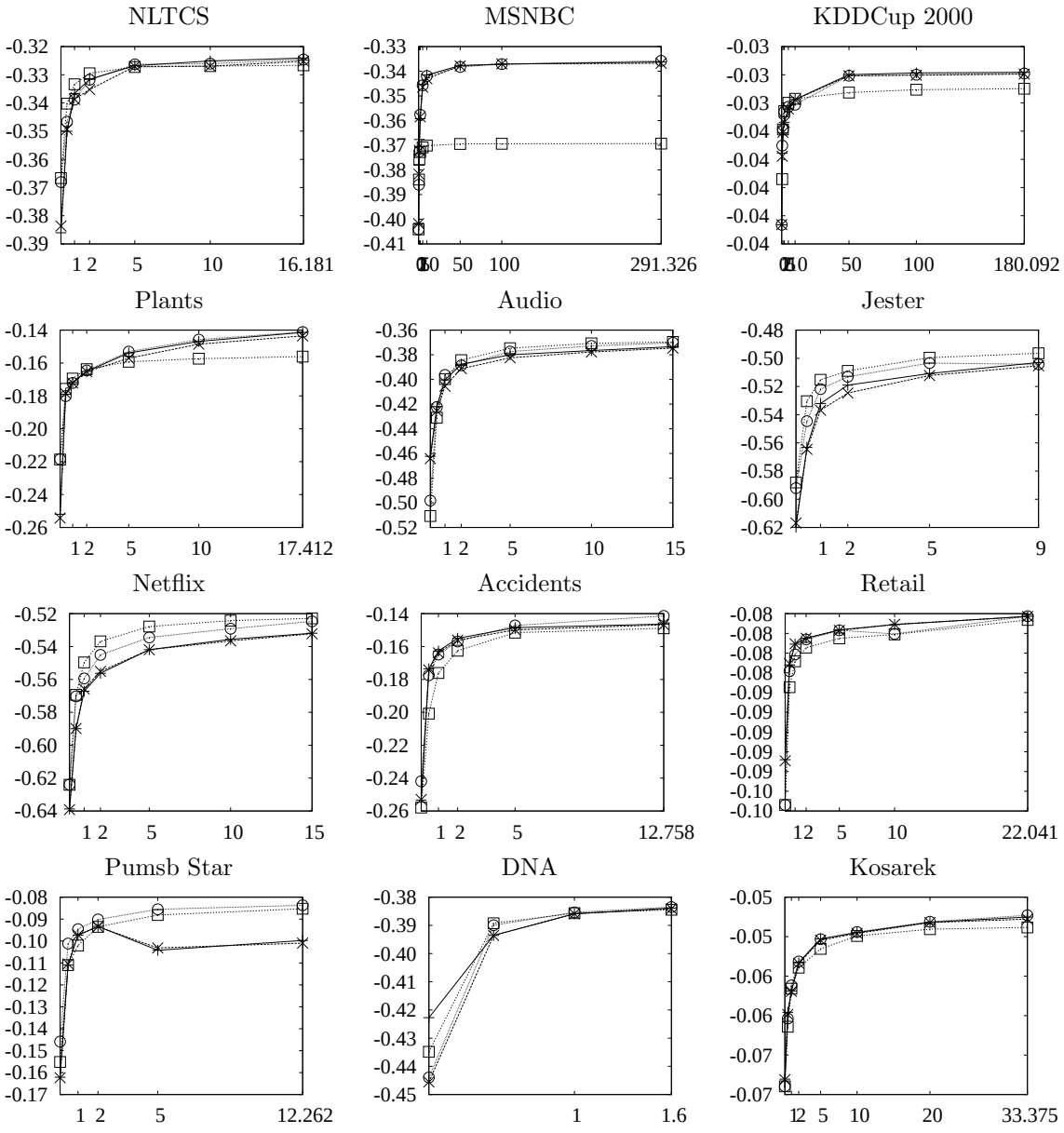


Figure 8: NCMLL vs. thousands of training examples for DT+L1 (circles), DT-BLM (pluses), DTSL (x marks), and L1 (boxes) on the first 12 data sets. Higher is better.

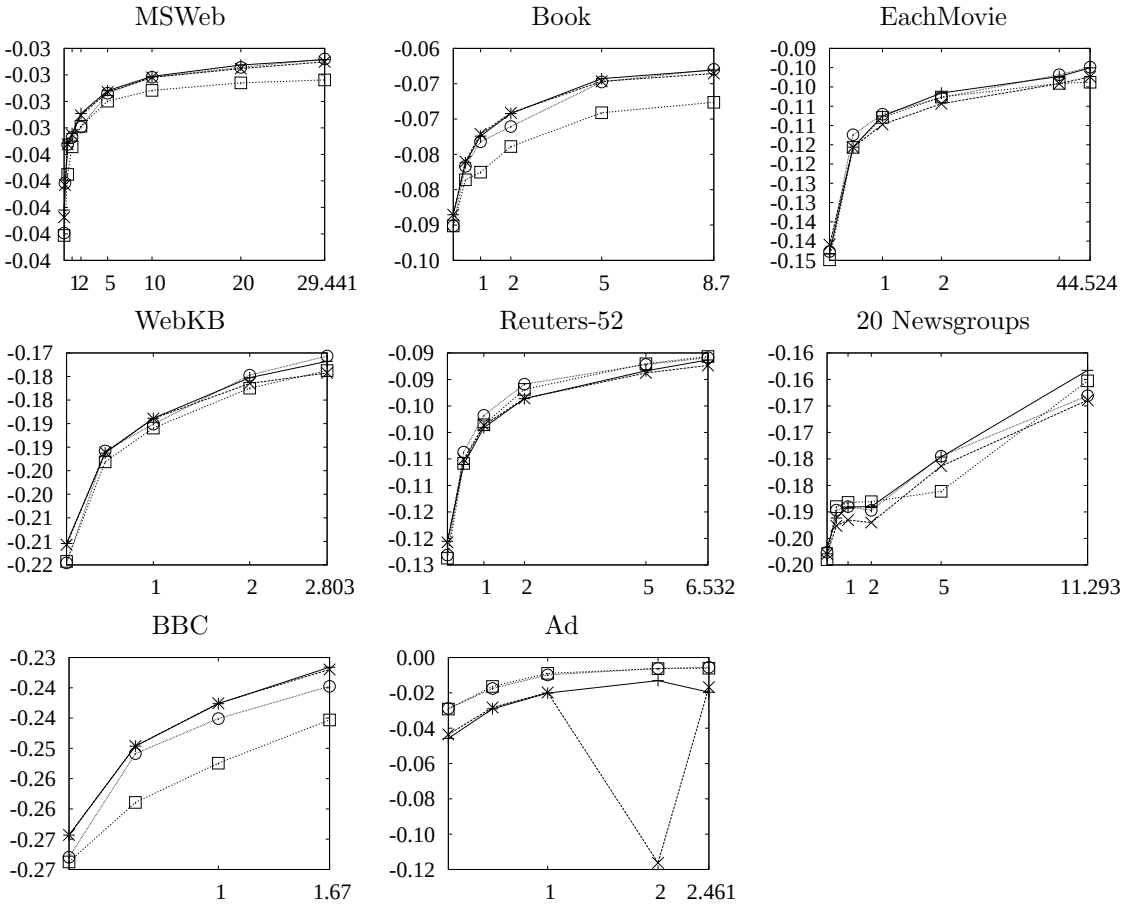


Figure 9: NCMLL vs. thousands of training examples for DT+L1 (circles), DT-BLM (pluses), DTSL (x marks), and L1 (boxes) on the last 8 data sets. Higher is better.

average, weight learning accounts for 93.8% of the total run time. For DTSL, this rises to 99.1% of the total run time. The factor that most influences weight learning time is the number of features: models with more features lead to longer weight learning times. Thus, more efficient weight learning techniques would substantially improve the running time of both structure learning algorithms.

7.6 Discussion

Both DTSL and L1 are typically faster and more accurate than DP and BLM. DTSL excels in domains that depend on higher-order interactions, while L1 performs better in domains that require many pairwise interactions. Therefore, neither algorithm dominates or subsumes the other; rather, they discover complementary types of structure.

DTSL has two weaknesses. The first is a higher risk of overfitting, since it often generates many very specialized features. For the most part, this can be remedied with careful tuning

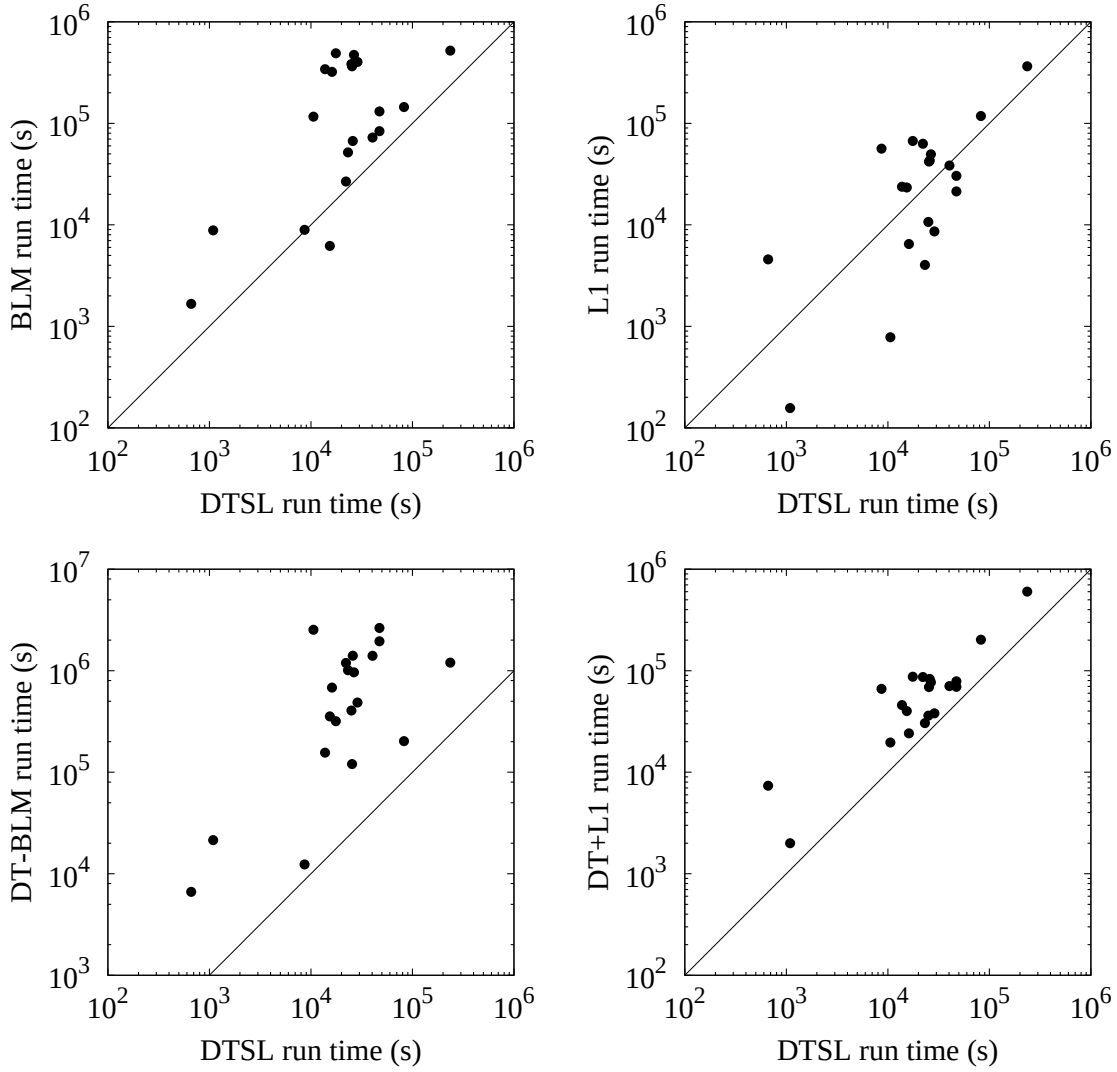


Figure 10: Running time on each data set in seconds (y-axis) relative to DTSL's running time (x-axis). Points below the line y=x represent data sets where DTSL was slower than the other algorithm. All times include tuning.

on a validation set. The second is a limited ability to capture many independent interactions. For instance, to capture pairwise interactions between a variable and k other variables would require a decision tree with 2^k leaves, even though such interactions could be represented exactly by $O(k)$ features.

DT-BLM extends DTSL by further refining the features it generates, merging them into a smaller set of more essential features. This leads to modest but consistent gains in accuracy over DTSL at the cost of significantly longer learning times.

Data Set	DP	BLM	L1	DTSL	DT-BLM	DT+L1
NLTCS	1,934	8,836	**157**	1,089	21,430	2,001
MSNBC	438,611	116,315	**782**	10,597	2,533,367	19,606
KDDCup 2000	335,398	51,744	**4,036**	23,201	1,012,635	30,463
Plants	423,593	321,880	**6,474**	16,118	682,405	24,145
Audio	653,111	402,441	**8,611**	28,738	155,970	45,829
Jester	583,969	341,830	23,769	**13,771**	485,892	38,078
Netflix	653,099	472,792	49,582	**26,560**	962,730	76,944
Accidents	637,574	364,282	41,982	**25,436**	120,434	69,045
Retail	279,507	384,011	**10,698**	25,030	405,059	36,090
Pumsb Star	707,530	489,854	66,971	**17,578**	318,017	87,292
DNA	197,406	1,666	4,570	**661**	6,635	7,365
Kosarek	626,336	130,962	**21,353**	47,363	1,947,259	69,363
MSWeb	426,199	83,727	**30,393**	47,363	2,635,238	78,579
Book	641,338	72,398	**38,444**	40,409	1,405,667	82,966
EachMovie	694,509	66,983	42,923	**25,873**	1,401,549	70,613
WebKB	715,885	26,754	63,010	**22,113**	1,192,672	86,727
Reuters-52	790,165	144,641	118,166	**82,467**	202,551	202,237
20 Newsgroups	792,268	520,361	364,512	**235,778**	1,201,506	601,137
BBC	864,000	8,944	56,293	**8,661**	12,352	66,197
Ad	719,893	**6,196**	23,375	15,373	354,843	40,083
Arith. Mean	559,116	200,831	48,805	**35,709**	852,911	86,738
Geom. Mean	416,573	87,876	**18,268**	19,172	378,405	48,662

Table 4: Run time in seconds, including parameter tuning. The best run time is shown in bold.

DT+L1 extends DTSL by including the features from L1, allowing it to capture many pairwise interactions and some higher-order interactions in the same model. As a result, DT+L1 does well overall across all data sets. DT+L1 is slower than DTSL but still much faster than DT-BLM. The accuracy of DT+L1 could perhaps be improved by performing additional tuning, rather than simply combining the models learned by DTSL and L1, or by using the features from DT-BLM. However, these modifications would also increase the learning time. The main risk of the expanded feature set used by DT+L1 is overfitting, which could explain its slightly worse performance on Jester and BBC.

8. Conclusions and Future Work

In this paper, we presented three new methods for using decision trees to learn the structure of Markov networks: DTSL, DT-BLM, and DT+L1.

DTSL is similar to the approach of Ravikumar et al. (2010), except that it uses decision trees in place of L1-regularized logistic regression. This allows it to learn longer features capturing interactions among more variables, which yields substantially better performance in several domains. DTSL is also similar to methods for learning dependency networks with tree conditional probability distributions (Heckerman et al., 2000). However, dependency networks may not represent consistent probability distributions and require that inference

	L1 Learning Times			DTSL Learning Times		
Data Set	Structure	Weights	Total	Structure	Weights	Total
NLTCS	7	151	157	2	1,087	1,089
MSNBC	155	626	782	126	10,471	10,597
KDDCup 2000	1,469	2,567	4036	780	22,421	23,201
Plants	126	6,348	6,474	34	16,084	16,118
Audio	138	8,473	8,611	49	28,688	28,738
Jester	79	23,690	23,769	27	13,745	13,771
Netflix	157	49,426	49,582	47	26,513	26,560
Accidents	213	41,769	41,982	43	25,393	25,436
Retail	306	10,392	10,698	112	24,918	25,030
Pumsb Star	293	66,679	66,971	43	17,534	17,578
DNA	38	4,532	4,570	6	655	661
Kosarak	1,210	20,143	21,353	238	47,125	47,363
MSWeb	2,254	28,140	30,393	485	46,878	47,363
Book	1,694	36,751	38,444	256	40,153	40,409
EachMovie	1,321	41,602	42,923	191	25,681	25,873
WebKB	2,052	60,959	63,010	212	21,901	22,113
Reuters-52	5,713	112,453	118,166	589	81,878	82,467
20 Newsgroups	9,971	354,541	364,512	1,455	234,323	235,778
BBC	1,555	54,738	56,293	173	8,488	8,661
Ad	4,510	18,865	23,375	740	14,634	15,373
Arith. Mean	1,663	47,142	48,805	280	35,428	35,709
Geom. Mean	507	17,060	18,267	110	18,987	19,173

Table 5: Run time for Ravikumar et al.'s algorithm and DTSL divided into time spent on structure learning and weight learning. Time is in seconds and includes parameter tuning.

be done with Gibbs sampling, while the Markov networks learned by DTSL have neither of those limitations.

In terms of speed, we found that DTSL and L1-regularized logistic regression (Ravikumar et al., 2010) had similar speed, while BLM (Davis and Domingos, 2010) and Della Pietra et al. (1997) were significantly slower. With a faster weight learning method, this comparison would be even more favorable to DTSL and L1, since most of their time was spent on the final weight learning step. In terms of accuracy, DTSL is comparable in accuracy to other approaches, placing ahead of all three baselines on nine out of 20 data sets.

The other two methods are extensions of DTSL that combine it with other structure learning algorithms. DT-BLM builds on DTSL by running the BLM bottom-up structure learning algorithm on the features generated by DTSL. This usually leads to slightly better accuracy, but is also much slower. DT+L1 extends DTSL by adding the features learned by L1-regularized logistic regression. This hybrid approach is very effective: DT+L1 is one of the most accurate methods on 15 out of 20 data sets and runs much faster than DT-BLM.

Future work includes exploring other methods of learning local structure, such as rule sets, boosted decision trees, and neural networks; determining sufficient conditions for the asymptotic consistency of local learning; further improving speed, perhaps by using frequent itemsets; and incorporating faster methods for weight learning, since structure learning is no longer the bottleneck.

Acknowledgments

The authors would like to thank the anonymous reviewers for many helpful suggestions. We also thank Jan Van Haaren for his valuable feedback on the article. DL is partly supported by ARO grant W911NF-08-1-0242 and NSF grant IIS-1118050. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO or the United States Government. JD is partially supported by the research fund KU Leuven (CREA/11/015 and OT/11/051), and EU FP7 Marie Curie Career Integration Grant (#294068).

References

G. Andrew and J. Gao. Scalable training of l1-regularized log-linear models. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, pages 33–40. ACM Press, 2007.

J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195, 1975.

C. Blake and C. J. Merz. UCI repository of machine learning databases. Machine-readable data repository, Department of Information and Computer Science, University of California at Irvine, Irvine, CA, 2000. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

F. Bromberg, D. Margaritis, and V. Honavar. Efficient Markov network structure discovery using independence tests. *Journal of Artificial Intelligence Research*, 35(2):449–484, 2009.

D. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 80–89, Providence, RI, 1997. Morgan Kaufmann.

J. Davis and P. Domingos. Bottom-up learning of Markov network structure. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pages 271–278, Haifa, Israel, 2010. ACM Press.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–392, 1997.

P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, Boston, MA, 2000. ACM Press.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, (9):1871–1874, 2008.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London, UK, 1996.

K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

G. Hulten and P. Domingos. Mining complex models from arbitrarily large databases in constant time. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 525–531, Edmonton, Canada, 2002. ACM Press.

R. Kohavi, C. Brodley, B. Frasca, L. Mason, and Z. Zheng. KDD-Cup 2000 organizers' report: Peeling the onion. *SIGKDD Explorations*, 2(2):86–98, 2000.

A. Kulesza and F. Pereira. Structured learning with approximate inference. In *Advances in Neural Information Processing Systems 20*, pages 785–792, 2007.

S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L1-regularization. In *Advances in Neural Information Processing Systems 19*, pages 817–824. MIT Press, 2007.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.

D. Lowd and J. Davis. Learning Markov network structure with decision trees. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*, pages 334–343, Sydney, Australia, 2010. IEEE Computer Society Press.

A. McCallum. Efficiently inducing features of conditional random fields. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 403–410, Acapulco, Mexico, 2003. Morgan Kaufmann.

K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 467–475. Morgan Kaufmann, Stockholm, Sweden, 1999.

P. Ravikumar, M. J. Wainwright, and J. Lafferty. High-dimensional ising model selection using L1-regularized logistic regression. *Annals of Statistics*, 38(3):1287–1319, 2010.

M. Schmidt and K. Murphy. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 709–716, 2010.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Springer, New York, NY, 1993.

I. Tsamardinos, L. Brown, and C. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

J. Van Haaren and J. Davis. Markov network structure learning: A randomized feature generation approach. In *Proceedings of the Twenty-Sixth National Conference on Artificial Intelligence*, pages 1148–1154. AAAI Press, 2012.

C. Ziegler, S. McNee, J. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the Fourteenth International World Wide Web Conference*, pages 22–32, 2005.

Journal of Machine Learning Research 15 (2014) 533-564 Submitted 10/11; Revised 8/13; Published 2/14

Ground Metric Learning

Marco Cuturi MCUTURI@I.KYOTO-U.AC.JP
David Avis AVIS@I.KYOTO-U.AC.JP
Graduate School of Informatics
Kyoto University
36-1 Yoshida-Honmachi, Sakyo-ku
Kyoto 606-8501, Japan

Editor: Gert Lanckriet

Abstract

Optimal transport distances have been used for more than a decade in machine learning to compare histograms of features. They have one parameter: the *ground metric*, which can be any metric between the features themselves. As is the case for all parameterized distances, optimal transport distances can only prove useful in practice when this parameter is carefully chosen. To date, the only option available to practitioners to set the ground metric parameter was to rely on *a priori* knowledge of the features, which limited considerably the scope of application of optimal transport distances. We propose to lift this limitation and consider instead algorithms that can learn the ground metric using only a training set of labeled histograms. We call this approach ground metric learning. We formulate the problem of learning the ground metric as the minimization of the difference of two convex polyhedral functions over a convex set of metric matrices. We follow the presentation of our algorithms with promising experimental results which show that this approach is useful both for retrieval and binary/multiclass classification tasks.

Keywords: optimal transport distance, earth mover's distance, metric learning, metric nearness

1. Introduction

We consider in this paper the problem of learning a distance for normalized histograms. Normalized histograms, namely finite-dimensional vectors with nonnegative coordinates whose sum is equal to 1, arise frequently in natural language processing, computer vision, bioinformatics and more generally areas involving complex datatypes. Objects of interest in such areas are usually simplified and are represented as a bag of smaller features. The occurrence frequencies of each of these features in the considered object can be then represented as a histogram. For instance, the representation of images as histograms of pixel colors, SIFT or GIST features (Lowe 1999, Oliva and Torralba 2001, Douze et al. 2009); texts as bags-of-words or topic allocations (Joachims 2002, Blei et al. 2003, Blei and Lafferty 2009); sequences as n-grams counts (Leslie et al. 2002) and graphs as histograms of subgraphs (Kashima et al. 2003) all follow this principle.

Various distances have been proposed in the statistics and machine learning literatures to compare two histograms (Amari and Nagaoka 2001, Deza and Deza 2009, §14). Our focus is in this paper is on the family of optimal transport distances, which is both well

motivated theoretically (Villani 2003, Rachev 1991) and works well empirically (Pele and Werman 2009). Optimal transport distances are particularly popular in computer vision, where, following the influential work of Rubner et al. (1997), they were called *Earth Mover's Distances* (EMD).

Optimal transport distances can be thought of as meta-distances that build upon a *metric on the features* to form a *distance on histograms of features*. Such a metric between features, which is known in the computer vision literature as the *ground metric*,[1] is the only parameter of optimal transport distances. In their seminal paper, Rubner et al. (2000) argue that, *"in general, the ground distance can be any distance and will be chosen according to the problem at hand"*. As a consequence, the earth mover's distance has only been applied to histograms of features when a good candidate for the ground metric was available beforehand. We argue that this is problematic in two senses: first, this restriction limits the application of optimal transport distances to problems where such a knowledge exists. Second, even when such an *a priori* knowledge is available, we argue that there cannot be a "universal" ground metric that will be suitable for all learning problems involving histograms on such features. As with all parameters in machine learning algorithms, the ground metric should be selected adaptively using data samples. The goal of this paper is to propose *ground metric learning* algorithms to do so.

This paper is organized as follows: after providing background and a few results on optimal transport distances in Section 2, we propose in Section 3 a criterion to select a ground metric given a training set of labeled histograms. We then show how to obtain a local minimum for that criterion using a projected subgradient descent algorithm in Section 4. We provide a review of other relevant distances and metric learning techniques in Section 5, in particular Mahalanobis metric learning techniques (Xing et al. 2003, Weinberger et al. 2006, Weinberger and Saul 2009, Davis et al. 2007) which have inspired much of this work. We provide empirical evidence in Section 6 that the metric learning framework proposed in this paper compares favorably to competing tools in terms of retrieval and classification performance. We conclude this paper in Section 7 by providing a few research avenues that could alleviate the heavy computational price tag of these techniques.

Notations: We consider throughout this paper histograms of length $d \geq 1$. We use upper case letters $A, B, \ldots$ for $d \times d$ matrices. Bold upper case letters $\mathbf{A}, \mathbf{B}, \ldots$ stand for larger matrices; lower case letters $r, c, \ldots$ are used for vectors of $\mathbb{R}^d$ or simply scalars in $\mathbb{R}$. An upper case letter M and its bold lower case $\mathbf{m}$ stand for the same matrix written in $d \times d$ matrix form or d^2 vector form by stacking successively all its column vectors from the left-most on the top to the right-most at the bottom. The notations $\overline{\mathbf{m}}$ and $\underline{\mathbf{m}}$ stand respectively for the strict upper and lower triangular parts of M expressed as vectors of size $\binom{d}{2}$. The order in which these elements are enumerated must be coherent in the sense that the upper triangular part of M^T expressed as a vector must be equal to $\underline{\mathbf{m}}$. Finally, we use the Frobenius dot-product for both matrix and vector representations, written as $\langle A, B \rangle \stackrel{\text{def}}{=} \operatorname{tr}(A^T B) = \mathbf{a}^T \mathbf{b}$.

1. Since the terms *metric* and *distance* are interchangeable mathematically speaking, we will always use the term *metric* for a metric between features and the term *distance* for the resulting transport distance between histograms, or more generally any other distance on histograms.

2. Optimal Transport Between Histograms

We recall in this section a few facts about optimal transport between two histograms. A more general and technical introduction is provided by Villani (2003, Introduction and §7); practical insights and motivation for the application of optimal transport distances in machine learning can be found in Rubner et al. (2000); a recent review of extensions and acceleration techniques to compute the EMD can be found in Pele and Werman (2009, §2).

Our interest in this paper lies in defining distances for pairs of probability vectors, namely on two nonnegative vectors r and c with the same sum. We consider in the following vectors of length d, and define the probability simplex accordingly:

$$\Sigma_d \stackrel{\text{def}}{=} \{u \in \mathbb{R}_+^d \mid \sum_{i=1}^{d} u_i = 1\}.$$

Optimal transport distances build upon two ingredients: (1) a $d \times d$ metric matrix, known as the ground metric parameter of the distance; (2) a feasible set of $d \times d$ matrices known as the transport polytope. We provide first an intuitive description of optimal transport distances in Section 2.1 (which can be skipped by readers familiar with these concepts) and follow with a more rigorous exposition in Section 2.2.

2.1 The Intuition behind Optimal Transport

The fundamental idea behind optimal transport distances is that they can be used to compare histograms of features, when the features lie in a metric space and can therefore be compared one with the other. To illustrate this idea, suppose we wish to compare images of $10 \times 10 = 100$ pixels. Suppose further, for the sake of simplicity, that these pixels can only take values in a range of 4 possible colors, **dark red**, light red, **dark blue** and light blue, and that each image is represented as a histogram of 4 colors as in Figure 1.

So called *bin-to-bin* distances (we provide a formal definition in Section 5.1) would compute the distance between a and b by comparing for each given index i their coordinates a_i and b_i one at a time. For instance, computing the Manhattan distances (the l_1 norm of the difference of two vectors) of three histograms a, b and c in Figure 1, we obtain that a is equidistant to b and c. However, upon closer inspection, assuming that dark and light red have more in common than, say, dark red and dark blue, one may have the intuition that c should be closer to a than it is to b. Optimal transport theory implements this intuition by carrying out an optimization procedure to compute a distance between histograms. Such an optimization procedure builds upon a set of feasible solutions (transport mappings) and a cost function (a linear cost), to define an optimal transport.

$$X = \begin{array}{c} \begin{array}{cccc} \textbf{13} & \textbf{7} & \textbf{56} & \textbf{24} \end{array} \\ \left[\begin{array}{cccc} 10 & 4 & 40 & 6 \\ 1 & 1 & 11 & 7 \\ 1 & 2 & 4 & 7 \\ 1 & 0 & 1 & 4 \end{array}\right] \end{array} \begin{array}{c} \textbf{60} \\ \textbf{20} \\ \textbf{14} \\ \textbf{6} \end{array} \tag{1}$$

Mapping a to b: An assignment between a and b assigns to each of the 100 colored pixels of a one of the 100 colored pixels of b. By grouping these assignments according to the 4×4

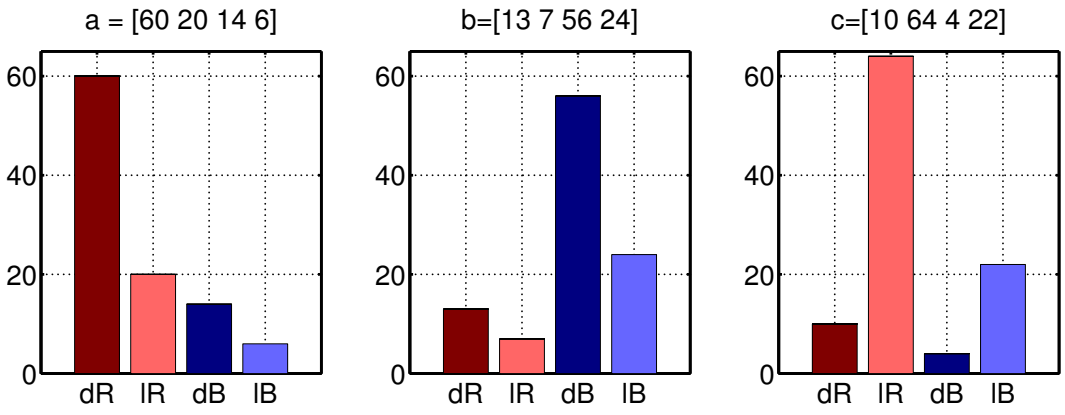


Figure 1: Three color histograms summing to 100. Although a and c are arguably closer to each other because of their overlapping dominance in red colors, the Manhattan distance cannot consider such an overlap and treats all colors separately. As a result, in this example, a is equidistant from b and c, $\|a-b\|_1 = \|a-c\|_1 = 120$.

possible color pairs, we obtain a 4×4 matrix which details, for each possible pair of colors (i, j), the overall amount x_{ij} of pixels of color i in a which have been morphed into pixels of color j in b. Because such a matrix representation only provides *aggregated* assignments and does not detail the actual individual assignments, such matrices are known as transport plans. A transport plan between a and b must be such that its row and column sums match the quantities detailed in a and b, as highlighted on the top and right side of an example matrix X in Equation 1.

$$M = \begin{array}{c} \begin{matrix} \bullet & \bullet & \bullet & \bullet \end{matrix} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{matrix} \tag{2}$$

A Linear Cost for Transport Plans: A cost matrix M quantifies all 16 possible costs m_{ij} of turning a pixel of a given color i into another color j. In the example provided in Equation 3, M states for instance that the cost of turning a dark red pixel into a dark blue pixel is twice that of turning it into a light red pixel; that transferring a colored pixel from a to the *same* color in b has a zero cost for all four colors. The cost of a transport plan X, given the cost matrix M, is defined as the Frobenius dot-product of X and M, namely $\langle X, M \rangle = \sum_{ij} x_{ij} m_{ij} = 169$ in our example.

$$X^\star = \begin{array}{c} \begin{array}{cccc} \mathbf{13} & \mathbf{7} & \mathbf{56} & \mathbf{24} \end{array} \\ \left[\begin{array}{cccc} 13 & & 42 & 5 \\ & 7 & & 13 \\ & & 14 & \\ & & & 6 \end{array}\right] \end{array} \begin{array}{c} \\ \mathbf{60} \\ \mathbf{20} \\ \mathbf{14} \\ \mathbf{6} \end{array} \tag{3}$$

Smallest Possible Total Transport Cost: The transport distance is defined as the lowest cost one could possibly find by considering *all* possible transport plans from a to b. Computing such an optimum involves solving a linear program, as detailed in Section 2.3. For a and b and given M above, solving this program would return an optimal matrix $X^\star$ provided in Equation (3) with an optimum of $\langle X^\star, M \rangle = 120$. When comparing a and c, the distance would, on the other hand, be equal to 72. Comparing these two numbers, we can see that the transport distance agrees with our initial intuition that a is closer to c than b by taking into account a metric on features. We define rigorously the properties of both the cost matrix M and the set of transport plans in the next section.

2.2 The Ingredients of Discrete Optimal Transport

Optimal transport distances between histograms are computed through a mathematical program. The feasible set of that program is a polytope of matrices. Its objective is a linear function parameterized by metric matrices. We define both in the sections below.

2.2.1 Objective: Semimetric and Metric Matrices

Consider d points labeled as $\{1, 2, \ldots, d\}$ in a metric space. Form now the $d \times d$ matrix M where element m_{ij} is equal to the distance between points i and j. Because of the metric axioms, the elements of M must obey three rules: (1) symmetry: $m_{ij} = m_{ji}$ for all pairs of indices i, j; (2) $m_{ii} = 0$ for all indices i and more generally $m_{ij} \geq 0$ for any pair (i, j); (3) triangle inequality: $m_{ij} \leq m_{ik} + m_{kj}$, for all triplets of indices i, j, k. The set of all $d \times d$ matrices that observe such rules, and thus represent hypothetically the pairwise distances between d points taken in any arbitrary metric space, is known as the cone of semimetric matrices,

$$\mathcal{M} \stackrel{\text{def}}{=} \left\{ M \in \mathbb{R}^{d \times d} : \forall\, 1 \leq i, j, k \leq d, m_{ii} = 0, m_{ij} \leq m_{ik} + m_{kj} \right\} \subset \mathbb{R}_+^{d \times d}.$$

Note that the $\binom{d}{2}$ symmetry conditions $m_{ij} = m_{ji}$ and non-negativity conditions $m_{ij} \geq 0$ are contained in the d^3 linear inequalities described in the definition above. $\mathcal{M}$ is a polyhedral set, because it is defined by a finite set of linear equalities and inequalities. $\mathcal{M}$ is also a convex pointed cone as can be visualized in Figure 2 for $d = 3$. Additionally, if a matrix M satisfies conditions (1) and (3) but also has, in addition to (2), the property that $m_{ij} > 0$ whenever $i \neq j$, then we call M a metric matrix. We write $\mathcal{M}_+ \subset \mathcal{M}$ for the set of metric matrices, which is neither open nor closed.

2.2.2 Feasible Set: Transport Polytopes

Consider two vectors r and c in the simplex Σ_d. Let $U(r, c)$ be the set of $d \times d$ nonnegative matrices such that their row and columns sums are equal to r and c respectively, that is,

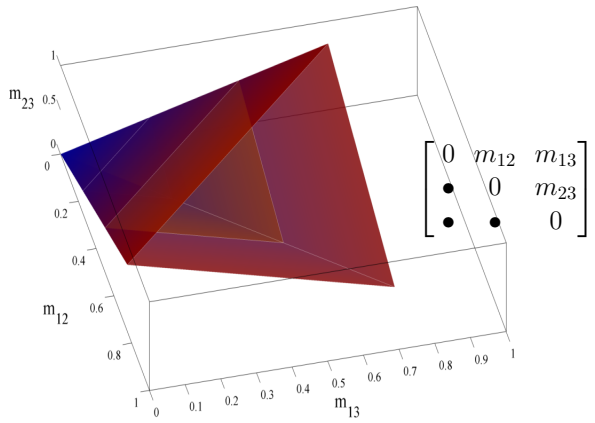


Figure 2: Semimetric cone in 3 dimensions. A $d \times d$ metric matrix for $d = 3$ can be described by 3 positive numbers m_{12}, m_{13} and m_{23} that follow the three triangle inequalities, $m_{12} \leq m_{13} + m_{23}$, $m_{13} \leq m_{12} + m_{23}$, $m_{23} \leq m_{12} + m_{13}$. The set (neither open nor closed) of *positive* triplets (m_{12}, m_{13}, m_{23}) forms the set of metric matrices.

writing $\mathbb{1}_d \in \mathbb{R}^d$ for the column vector of ones,

$$U(r, c) = \{X \in \mathbb{R}_+^{d \times d} \mid X\mathbb{1}_d = r,\ X^\top \mathbb{1}_d = c\}.$$

Because of these constraints, it is easy to see that any matrix $X = [x_{ij}]$ in $U(r, c)$ is such that $\sum_{ij} x_{ij} = 1$. While r and c can be interpreted as two probability measures on the discrete set $\{1, \ldots, d\}$, any matrix X in $U(r, c)$ is thus a probability measure on $\{1, \ldots, d\} \times \{1, \ldots, d\}$, the cartesian product of $\{1, \ldots, d\}$ with itself. $U(r, c)$ can be identified with the set of all discrete probabilities on $\{1, \ldots, d\} \times \{1, \ldots, d\}$ that admit r and c as their first and second marginals respectively.

$U(r, c)$ is a bounded polyhedron (the entries of any X in $U(r, c)$ are bounded between 0 and 1) and is thus a polytope with a finite set of extreme points. This polytope has an effective dimension of $d^2 - 2d + 1$ in the general case where r and c have positive coordinates (Brualdi 2006, §8.1). $U(r, c)$ is known in the operations research literature as the set of transport plans between r and c (Rachev and Rüschendorf 1998). When r and c are integer valued histograms with the same total sum, a transport plan with integral values is also known as a contingency table or a two-way table with fixed margins (Lauritzen 1982, Diaconis and Efron 1985).

2.3 Optimal Transport Distances

Given two histograms r and c of Σ_d and a matrix M, the quantity

$$G(r, c; M) \stackrel{\text{def}}{=} \min_{X \in U(r,c)} \langle M, X \rangle.$$

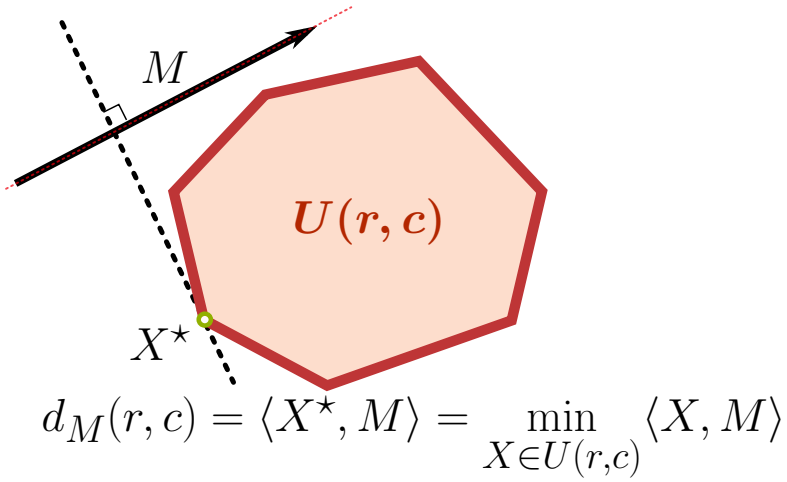


Figure 3: Schematic view of the optimal transport distance. Given a feasible set $U(r,c)$ and a cost parameter $M \in \mathcal{M}_+$, the distance between r and c is the minimum of $\langle X, M \rangle$ when X varies across $U(r,c)$. The minimum is reached here at $X^\star$.

describes the optimum of a linear program whose feasible set is defined by r and c and whose cost is parameterized by M. G is a positive homogeneous function of M, that is $G(r,c;tM) = tG(r,c;M)$ for $t \geq 0$. $G(r,c;M)$ can also be described as minus the support function (Rockafellar 1970, §13) of the polytope $U(r,c)$ evaluated at $-M$. A schematic view of that LP is given in Figure 3.

When M belongs to the cone of metric matrices $\mathcal{M}$, the value of $G(r,c;M)$ is a distance (Villani 2003, §7, p.207) between r and c, parameterized by M. In that case, assuming implicitly that M is fixed and only r and c vary, we will refer to $G(r,c;M)$ as $d_M(r,c)$, the optimal transport distance between r and c.

Theorem 1 *d_M is a distance on Σ_d whenever $M \in \mathcal{M}_+$.*

The fact that $d_M(r,c)$ is a distance is a well known result; a standard proof for continuous probability densities is provided in Villani (2003, Theorem 7.3). A proof often reported in the literature for the discrete case can be found in Rubner et al. (2000). We believe this proof is not very clear, so we provide an alternative proof in the Appendix.

When r and c are, on the contrary, considered fixed, we will use the notation $G_{rc}(M)$ to stress that M is the variable argument of G, as will be mostly the case in this paper. Although using two notations for the same mathematical object may seem cumbersome, these notations will allow us to stress alternatively which of the three variables r, c and M are considered fixed in our analysis.

2.3.1 Extensions of Optimal Transport Distances

The distance d_M bears many names: 1-Wasserstein; Monge-Kantorovich; Mallow's (Mallows 1972, Levina and Bickel 2001) and finally Earth Mover's (Rubner et al. 2000) in the computer vision literature. Rubner et al. (2000) and more recently Pele and Werman (2009)

have also proposed to extend the optimal transport distance to compare unnormalized histograms, that is vectors with nonnegative coordinates which do not necessarily sum to 1. Simply put, these extensions compute a distance between two unnormalized histograms u and v by combining any difference in the total mass of u and v with the optimal transport plan that can carry the whole mass of u onto v if $\|u\|_1 \leq \|v\|_1$ or v onto u if $\|v\|_1 \leq \|u\|_1$. These extensions can also be traced back to earlier work by Kantorovich and Rubinshtein (1958), see Vershik (2006) for a historical perspective. We will not consider such extensions in this work, and will only consider distances for histograms of equal sum.

2.3.2 Relationship with Other Distances

The optimal transport distance bears an interesting relationship with the total variation distance, which is a popular distance between histograms of features in computer vision following early work by Swain and Ballard (1991). As noted by Villani (2003, p.7 & Ex.1.17 p.36), the total variation distance, defined as

$$d_{\mathrm{TV}}(r,c) \stackrel{\text{def}}{=} \frac{1}{2}\|r-c\|_1,$$

can be seen as a trivial instance of optimal transport distances by simply noting that

$$d_{\mathrm{TV}} = d_{M_{\mathbb{1}}},$$

where $M_{\mathbb{1}}$ is the matrix of ones with a zero diagonal, namely $M_{\mathbb{1}}(i,j)$ is equal to 1 if $i = j$ and zero otherwise. The metric on features defined by $M_{\mathbb{1}}$ simply states that all d considered features are equally different, that is their pairwise distances are constant. This relationship between total variation and optimal transport can be compared to the analogous observation that Euclidean distances are a trivial instance of the Mahalanobis family of distances, by setting the Mahalanobis parameter to the identity matrix. Tuning the ground metric M to select an optimal transport distance d_M can thus be compared to the idea of tuning a positive-definite matrix Ω to define a suitable Mahalanobis distance for a given problem: Mahalanobis distances are to the Euclidean distance what optimal transport distances are to the total variation distance, as schematized in Figure 4. We discuss this parallel further when reviewing related work in Section 5.2.

2.3.3 Computing Optimal Transport Distances

The distance d_M between two histograms r and c can be computed as the solution of the following Linear Program (LP),

$$\begin{array}{rll} d_M(r,c) = & \text{minimize} & \sum_{i,j=1}^{d} m_{ij}x_{ij} \\ & \text{subject to} & \sum_{j=1}^{d} x_{ij} = r_i, 1 \leq i \leq d \\ & & \sum_{i=1}^{d} x_{ij} = c_j, 1 \leq j \leq d \\ & & x_{ij} \geq 0, 1 \leq i,j \leq d. \end{array}$$

This program is equivalent to the following program, provided in a more compact form, as:

$$\begin{array}{rll} d_M(r,c) = & \text{minimize} & \mathbf{m}^T\mathbf{x} \\ & \text{subject to} & \mathbf{A}\mathbf{x} = \begin{bmatrix} r \\ c \end{bmatrix}_* \\ & & \mathbf{x} \geq 0, \end{array} \tag{4}$$

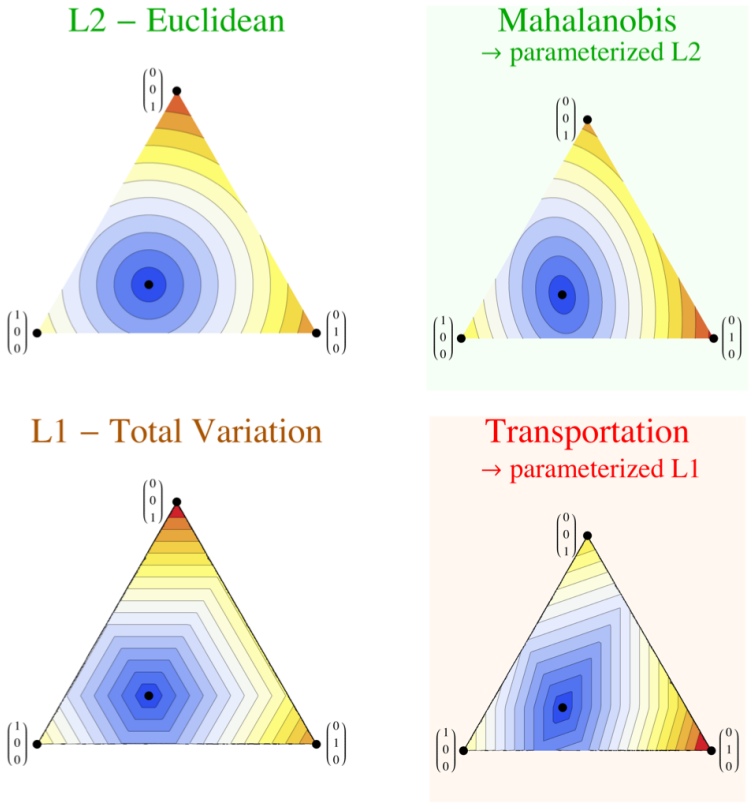


Figure 4: Contour plots of the Euclidean (top-left) and Total variation (bottom-left) of all points in the simplex for $d = 3$ to the point $[0.5, 0.3, 0.2]$, and their respective parameterized equivalents, the Mahalanobis distance (top-right) and the transport distance (bottom-right). The parameter for the Mahalanobis distance has been drawn randomly. The upper right values of the ground metric M are 0.8 and 0.4 on the first row and 0.6 on the second row.

where $\mathbf{A}$ is the $(2d-1) \times d^2$ matrix that encodes the row-sum and column-sum constraints for X to be in $U(r, c)$ as

$$\mathbf{A} = \begin{bmatrix} \mathbb{1}_{1\times d} \otimes I_d \\ I_d \otimes \mathbb{1}_{1\times d} \end{bmatrix}_*,$$

where $\otimes$ is Kronecker's product and the lower subscript $\left[\cdot\right]_*$ in a matrix (resp. a vector) means that its last line (resp. element) has been removed. This modification is carried out to make sure that all constraints described by $\mathbf{A}$ are independent, or equivalently that $\mathbf{A}^T$ is not rank deficient. This LP can be solved using the network simplex (Ford and Fulkerson 1962) or through more specialized minimum-cost network flow algorithms (Ahuja et al. 1993, §9). The computational effort required to compute a single distance between two histograms of dimension d scales typically as $O(d^3 \log(d))$ (Pele and Werman 2009, §2.3) when M has no particular structure.

2.4 Properties of the Optimal Transport Distance Seen As a Function of M

When both its arguments are fixed, the optimal transport distance $d_M(r, c)$ seen as a function G_{rc} of M has three important properties: G_{rc} is piecewise linear; concave; a subgradient of G_{rc} can be directly recovered by considering any optimal solution of the linear program considered to compute G_{rc}. These properties are crucial, because they highlight that for a given pair of histograms (r, c), a gradient direction to increase or decrease $d_M(r, c)$ can be obtained through the optimal transport plan that realizes $d_M(r, c)$, and that maximizing this value is a convex problem.

2.4.1 Concavity and Piecewise-linearity

Because its feasible set $U(r, c)$ is a bounded polytope and its objective is linear, Problem (4) has an optimal solution in the finite set $\mathrm{Ex}(r, c)$ of extreme points of $U(r, c)$ (Bertsimas and Tsitsiklis 1997, Theorem 2.7, p.65). G_{rc} is thus the minimum of a finite collection of linear functions, each indexed by an extreme point, and thus

$$G_{rc}(M) = \min_{X \in U(r,c)} \langle X, M \rangle = \min_{X \in \mathrm{Ex}(r,c)} \langle X, M \rangle, \tag{5}$$

is piecewise linear. G_{rc} is also concave by a standard result stating that the point-wise minimum of a family of affine functions is itself concave (Boyd and Vandenberghe 2004, §3.2.3).

2.4.2 Differentiability

Because the computation of G_{rc} involves a linear program, the gradient ∇G_{rc} of G_{rc} at a given point M is equal to the optimal solution $X^\star$ to Problem (4) whenever this solution is unique,

$$\nabla G_{rc} = X^\star,$$

as stated by Bertsimas and Tsitsiklis (1997, Theorem 5.3). Intuitively, by continuity of all functions involved in Problem (4) and the uniqueness of the optimal solution $X^\star$, one can show that there exists a ball with a positive radius around M for which $G_{rc}(M)$ is locally linear, equal to $\langle X^\star, M \rangle$ on that ball, resulting in the fact that the gradient of $\langle X^\star, M \rangle$ is simply $X^\star$. More generally and regardless of the uniqueness of $X^\star$, *any* optimal solution $X^\star$ of Problem (4) is in the sub-differential $\partial G_{rc}(M)$ of G_{rc} at M (Bertsimas and Tsitsiklis 1997, Lemma 11.4). Indeed, suppose that $Z(p)$ is the minimum of a linear program Z parameterized by a cost vector x, over a bounded feasible polytope with extreme points $\{c_1, \ldots, c_m\}$. $Z(x)$ can in that case be written as

$$Z(x) = \min_{i=1,\ldots,m} u_i + c_i^T x.$$

Then, defining $E(x) = \{i | Z(x) = u_i + c_i^T x\}$, namely the set of indices of extreme points which are optimal for x, Bertsimas and Tsitsiklis (1997, Lemma 11.4) show that for any fixed x and any index i in $E(x)$, c_i is a subgradient of Z at x. More generally, this lemma also shows that the differential of Z at x is exactly the convex hull of those optimal solutions $\{c_i\}_{i \in E(x)}$. If, as in Equation (5), these c_i's describe the set of extreme points of $U(r, c)$, the variable x is the ground metric M, and Z is G_{rc}, this lemma implies that any optimal

transport is necessarily in the subdifferential of $G_{rc}(M)$, and that this subdifferential is exactly the convex hull of all the optimal transports between r and c using cost M.

In summary, the distance $d_M(r, c)$ seen as a function of M ($G_{rc}(M)$ using our notations) can be computed by solving a network flow problem, and any optimal solution of that network flow is a subgradient of the distance with respect to M. This function itself is concave in M. We use extensively these properties in Section 4 when we optimize the criteria considered in the next section.

3. Learning Ground Metrics as an Optimization Problem

We define in this section a family of criteria to quantify the relevance of a ground metric to compare histograms in a given learning task. We use to that effect a training sample of histograms with additional information.

3.1 Training Set: Histograms and Side Information

Suppose that we are given a sample $\{r_1, \ldots, r_n\} \subset \Sigma_d$ of histograms in the canonical simplex along with a family of coefficients $\{\omega_{ij}\}_{1 \leq i,j \leq n}$, which quantify how similar r_i and r_j are. We assume that these coefficients are such that ω_{ij} is positive whenever r_i and r_j describe similar objects and negative for dissimilar objects. We further assume that this similarity is symmetric, $\omega_{ij} = \omega_{ji}$. The similarity of an object with itself will not be considered in the following, so we simply assume that $\omega_{ii} = 0$ for $1 \leq i \leq n$.

In the most simple case, these weights may reflect a labeling of all histograms into multiple classes and be set to $\omega_{ij} > 0$ whenever r_i and r_j come from the same class and $\omega_{ij} < 0$ for two different classes. An ever simpler setting which we consider in our experiments is that of setting $\omega_{ij} = \mathbb{1}_{y_i = y_j}$, where the label y_i of histogram r_i for $1 \leq i \leq n$ is taken in a finite set of labels $\mathcal{L} = \{1, 2, \ldots, L\}$. Let us introduce more notations before moving on to the next section. Since by symmetry $\omega_{ij} = \omega_{ji}$ and $G_{r_i r_j} = G_{r_j r_i}$, we restrict the set of pairs of indices (i, j) we will study to

$$\mathcal{I} \stackrel{\text{def}}{=} \{(i, j) \mid i, j \in \{1, \ldots, n\}, i < j\},$$

and introduce two subsets of $\mathcal{I}$, the subsets of similar and dissimilar histograms:

$$\mathcal{E}_+ \stackrel{\text{def}}{=} \{(i, j) \in \mathcal{I} \mid \omega_{ij} > 0\}; \quad \mathcal{E}_- \stackrel{\text{def}}{=} \{(i, j) \in \mathcal{I} \mid \omega_{ij} < 0\}.$$

Finally, we define the shorthand $G_{ij} \stackrel{\text{def}}{=} G_{r_i r_j}$.

3.2 Feasible Set of Metrics

We propose to formulate the ground metric learning problem as that of finding a metric matrix $M \in \mathcal{M}_+$ such that the corresponding optimal transport distance d_M computed between pairs of points in $(r_1, \ldots, r_n)$ agrees with the weights ω. However, because projectors are not well defined on feasible sets that are not closed, we will consider the whole of the semimetric cone $\mathcal{M}$ as a feasible set instead of considering $\mathcal{M}_+$ directly. We implicitly assume in this paper that, if our algorithms output a matrix that has null off-diagonal elements, such a matrix will be regularized by adding the same arbitrarily small positive

constant to all its off-diagonal elements. Moreover, and as remarked earlier, two histograms r and c define a homogeneous function G_{rc} of M, that is $G_{rc}(tM) = t\,G_{rc}(M)$. To remove this ambiguity on the scale of M, we only consider in the following matrices that lie in the intersection of $\mathcal{M}$ and the unit sphere in $\mathbb{R}^{d\times d}$ of the 1-norm,

$$\mathcal{M}_1 = \mathcal{M} \cap B_1,$$

where $B_1 = \{A \in \mathbb{R}^{d\times d} \mid \|A\|_1 \stackrel{\text{def}}{=} \|\mathbf{a}\|_1 = 1\}$. $\mathcal{M}_1$ is convex as the intersection of two convex sets. In what follows we call matrices in $\mathcal{M}_1$ metric matrices (this is a slight abuse of language since some of these matrices are in fact semimetrics).

3.3 A Local Criterion to Select the Ground Metric

More precisely, this criterion will favor metrics M for which the distance $d_M(r_i, r_j)$ is *small* for pairs of *similar* histograms r_i and r_j ($\omega_{ij} > 0$) and *large* for pairs of *dissimilar* histograms ($\omega_{ij} < 0$). We build such a criterion by considering the family of all $\binom{n}{2}$ pairs

$$\{\,(\omega_{ij}, G_{ij}(M))\,, (i,j) \in \mathcal{I}\}.$$

Given the i^{th} datum of the training set, we consider the subsets $\mathcal{E}_{i+}$ and $\mathcal{E}_{i-}$ of points that share their label with r_i and those that do not respectively:

$$\mathcal{E}_{i+} \stackrel{\text{def}}{=} \{j | (i,j) \text{ or } (j,i) \in \mathcal{E}_+\}, \quad \mathcal{E}_{i-} \stackrel{\text{def}}{=} \{j | (i,j) \text{ or } (j,i) \in \mathcal{E}_-\}.$$

Within these subsets, we consider the sets N_{ik}^+ and N_{ik}^-, which stand for the indices of any k nearest neighbours of r_i using distance d_M and whose indices are taken respectively in the subsets $\mathcal{E}_{i+}$ and $\mathcal{E}_{i-}$. For each index i and corresponding histogram r_i, we can now form the weighted sum of distances to its *similar* and *dissimilar* neighbors

$$S_{ik}^+(M) \stackrel{\text{def}}{=} \sum_{j\in N_{ik}^+} \omega_{ij}\, G_{ij}(M), \quad \text{and} \quad S_{ik}^-(M) \stackrel{\text{def}}{=} \sum_{j\in N_{ik}^-} \omega_{ij}\, G_{ij}(M). \tag{6}$$

Note that N_{ik}^+ and N_{ik}^- are not necessarily uniquely defined. Whenever more than one list of indices can qualify as the k closest neighbors of r_i, we select such a list randomly among all possible choices. We adopt the convention that $N_{ik}^+ = \mathcal{E}_{i+}$ whenever k is larger than the cardinality of $\mathcal{E}_{i+}$, and follow the same convention for N_{ik}^-. We use these two terms to form our final criterion:

$$C_k(M) \stackrel{\text{def}}{=} \sum_{i=1}^{n} S_{ik}^+(M) + S_{ik}^-(M). \tag{7}$$

4. Approximate Minimization of C_k

Since all functions G_{ij} are concave, C_k can be cast as a difference of convex functions

$$C_k(M) = S_k^-(M) - \text{-}S_k^+(M),$$

where both

$$S_k^-(M) \stackrel{\text{def}}{=} \sum_{i=1}^{n} S_{ik}^-(M) \;\text{ and }\; \text{-}S_k^+(M) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \text{-}S_{ik}^+(M)$$

are convex, by virtue of the convexity of each of the terms S_{ik}^- and -S_{ik}^+ defined in Equation (6). This follows in turn from the concavity of each of the distances G_{ij} as discussed in Sections 2.4 and 3.3, and the fact that such functions are weighted by negative factors, ω_{ij} for $(i,j) \in \mathcal{E}_-$ and -ω_{ij} for $(i,j) \in \mathcal{E}_+$. We propose an algorithm to approximate the minimization of C_k defined in Equation (7) that takes advantage of this decomposition.

4.1 Subdifferentiability of C_k

It is easy to see that, using the results on G_{rc} we have recalled in Section 2.4.1, the gradient of C_k computed at a given metric matrix M is

$$\nabla C_k(M) = \nabla S_k^-(M) + \nabla S_k^+(M),$$

where,

$$\nabla S_k^+(M) = \sum_{i=1}^n \sum_{j \in N_{ik}^+} \omega_{ij} X_{ij}^\star, \quad \nabla S_k^-(M) = \sum_{i=1}^n \sum_{j \in N_{ik}^-} \omega_{ij} X_{ij}^\star,$$

whenever all solutions $X_{ij}^\star$ to the linear programs G_{ij} considered in C_k are unique and whenever each of the two sets of k nearest neighbors of each histogram r_i is unique. Also as recalled in Section 2.4.1, any optimal solution $X_{ij}^\star$ is in the sub-differential $\partial G_{ij}(M)$ of G_{ij} at M and we thus have that

$$\sum_{i=1}^n \sum_{j \in N_{ik}^+} \omega_{ij} X_{ij}^\star \in \partial S_k^+(M), \quad \sum_{i=1}^n \sum_{j \in N_{ik}^-} \omega_{ij} X_{ij}^\star \in \partial S_k^-(M),$$

regardless of the unicity of the nearest-neighbors sets of each histogram r_i. The details of the computation of $S_k^-(M)$ and of the subgradient described above are given in Algorithm 1. The computations for $S_k^+(M)$ are analogous to those of $S_k^-(M)$ and we use the abbreviation $S_k^\pm(M)$ to consider either of these two cases in our algorithm outline.

4.2 Local Linearization of the Concave Part of C_k

We describe in Algorithm 2 a simple approach to obtain an approximate solution to the problem of minimizing C_k with a projected subgradient descent and a local linearization of the concave part of C_k. Algorithm 2 runs a subgradient descent on C_k using two nested loops: we linearize the concave part of C_k in an outer loop and minimize the resulting convex approximation in the inner loop.

More precisely, the first loop is parameterized with an iteration counter p and starts by computing both S_k^+ (the concave part of C_k) and a vector γ_+ in its subdifferential using the current candidate metric M_p. Using this value and the subgradient γ_+, the concave part S_k^+ of C_k can be locally approximated by its first order Taylor expansion,

$$C_k(M) \approx S_k^-(M) + S_k^+(M_p) + \gamma_+^T(M - M_p).$$

This approximation is convex, larger than C_k and can be minimized in an inner loop using a projected subgradient descent. When this convex function has been minimized up to

Algorithm 1 Computation of $z = S_k^{\pm}(M)$ and a subgradient γ, where $\pm$ is either $+$ or $-$.

Input: $M \in \mathcal{M}_1$.
for $(i,j) \in \mathcal{E}_{\pm}$ **do**
  Compute the optimum $z_{ij}^{\star}$ and an optimal solution $X_{ij}^{\star}$ for Problem (4) with cost vector $\mathbf{m}$ and constraint vector $[r_i; r_j]_*$.
end for
Set $G = 0, z = 0$.
for $i \in \{1, \cdots, n\}$ **do**
  Select the smallest k elements of $z_{ij}^{\star}, j \in \mathcal{E}_{i\pm}$ to define the set of neighbors $N_{ik}^{\pm}$.
  for $j \in N_{ik}^{\pm}$ **do**
    $G \leftarrow G + \omega_{ij} X_{ij}^{\star}$.
    $z \leftarrow z + \omega_{ij} z^{\star}$.
  end for
end for
Output z and $\gamma = \overline{\mathbf{g}} + \underline{\mathbf{g}}$.

sufficient precision, we obtain a point

$$M_{p+1} \in \operatorname*{argmin}_{M \in \mathcal{M}_1} S_k^{-}(M) + S_k^{+}(M_p) + \gamma_+^T (M - M_p).$$

We increment p and repeat the linearization step described above. The algorithm terminates when sufficient progress in the outer loop has been realized, at which point the matrix computed in the last iteration is returned as the output of the algorithm.

The overall quality of the solution obtained through this procedure is directly linked to the quality of the initial point M_0. The selection of M_0 requires thus some attention. We provide a few options to select M_0 in the next section.

4.3 Initial Points

Since C_k is not a convex criterion, particular care needs to be taken to initialize our descent algorithm. We propose in this section two approaches to choose the initial point M_0.

4.3.1 The Total Variation Distance as an Optimal Transport Distance

The total variation distance between two histograms, defined as half the l_1 norm of their difference, can provide an educated guess to define an initial point M_0 to optimize C_k. Indeed, as explained in Section 2.3, the total variation distance can be interpreted as the optimal transport distance parameterized with the uniform ground metric $M_{\mathbb{1}}$ which is a matrix equal to 1 on all its off-diagonal terms and 0 on the diagonal. Therefore, we consider $M_{\mathbb{1}}$ (divided by $d(d-1)$ to normalize it) in our experiments to initialize Algorithm 2. Since C_k is not convex, using $M_{\mathbb{1}}$ is attractive from a numerical point of view because $M_{\mathbb{1}}$ exhibits the highest entropy among all matrices in $\mathcal{M}_1$. This choice has, however, two drawbacks:

- Because all the costs enumerated in $M_{\mathbb{1}}$ are equal, one can show that for a pair of histograms (r, c) *any* transport matrix that assigns the maximum weight to its

Algorithm 2 Projected Subgradient Descent to minimize C_k

Input $M_0 \in \mathcal{M}_1$ (see Section 4.3), gradient step t_0.
$t \leftarrow 1$.
$p \leftarrow 0, \quad M_0^{\text{out}} \leftarrow M_0$.
while $p < p_{\max}$ or insufficient progress for z_p^{out} **do**
  Use Algorithm 1 to compute $z_+ \stackrel{\text{def}}{=} S_k^+(M_p^{\text{out}})$ and γ_+.
  $q \leftarrow 0, \quad M_0^{\text{in}} \leftarrow M_p^{\text{out}}$.
  while $q < q_{\max}$ or insufficient progress for z_q^{in} **do**
    Compute γ_- and z_- of S_k^- using Algorithm 1 with M_q^{in}, $(i,j) \in \mathcal{E}^-$.
    Set $z_q^{\text{in}} \leftarrow z_- + z_+ + \gamma_+^T(\underline{\mathbf{m}}_q^{\text{in}} - \underline{\mathbf{m}}_p^{\text{out}})$.
    Set $M_{q+1}^{\text{in}} \leftarrow P_{\mathcal{M}_1}\left(\underline{\mathbf{m}}_q^{\text{in}} - \frac{t_0}{\sqrt{q}}(\gamma_+ + \gamma_-)\right)$.
    $q \leftarrow q+1$.
    $t \leftarrow t+1$.
  end while
  $M_{p+1}^{\text{out}} \leftarrow M_q^{\text{in}}$.
  $p \leftarrow p+1$.
end while
Output M_p^{out}.

diagonal elements, namely any matrix X in the convex set

$$\{X \in U(r,c) |\, x_{ii} = \min(r_i, c_i)\}$$

is optimal. As a result, any matrix in that set is in the subdifferential of G_{rc} at $M_{\mathbb{1}}$. Solvers that build upon the network simplex will return an arbitrary vertex within that set, mostly depending on the pivot rule they use. The very first subgradient descent iteration is thus likely to be extremely uninformative, and this should be reflected by a poor initial behaviour which we do indeed observe in practice.

- Because such a starting point ignores the information provided by all histograms $\{r_i, 1 \leq i \leq n\}$ and weights $\{\omega_{ij}, (i,j) \in \mathcal{I}\}$, we expect it to be far from the actual optimum.

We propose an alternative approach in the next section: we approximate C_k by a linear function of M and set M_0 to be the minimizer of that approximation.

4.3.2 Linear Approximations to C_k and Independence Tables

We propose to form an initial point M_0 by replacing the optimization underlying the computation of each distance $G_{ij}(M)$ by a dot product,

$$G_{ij}(M) = \min_{X \in U(r_i, r_j)} \langle M, X \rangle \approx \langle M, \Xi_{ij} \rangle,$$

where Ξ_{ij} is a representative matrix of the polytope $U(r_i, r_j)$. This idea is illustrated in Figure 5. We discuss a natural choice to define Ξ_{ij} later in this section. Assuming we

have chosen such matrices, we replace now each term G_{ij} in the criterion presented in Equation (7) by the corresponding quantity $\langle M, \Xi_{ij} \rangle$ and obtain an approximation χ_k of C_k parameterized by a matrix Ξ_k,

$$\chi_k(M) \stackrel{\text{def}}{=} \langle M, \Xi_k \rangle, \text{ where } \Xi_k \stackrel{\text{def}}{=} \sum_{i=1}^{n} \sum_{j \in N_{ik}^- \cup N_{ik}^+} \omega_{ij} \, \Xi_{ij},$$

where the k nearest neighbors of each histogram r_i defined in N_{ik}^- and N_{ik}^+ are those selected by considering the total variation distance. To select a candidate matrix M that minimizes this criterion, we consider the following penalized problem,

$$\min_{M \in \mathcal{M}} \lambda \langle M, \Xi_k \rangle + \|M\|_2^2 = \min_{M \in \mathcal{M}} \|M + \frac{\lambda}{2} \Xi_k\|_2^2, \quad \lambda > 0, \tag{8}$$

which can be solved using the approach described by Brickell et al. (2008, Algorithm 3.1). Brickell et al. propose *triangle fixing* algorithms to obtain projections on the cone of distances under various norms, including the Euclidean distance. They study in particular the following problem,

$$\min_{M \in \mathcal{M}} \|M - H\|_2, \tag{9}$$

where H is a *symmetric nonnegative* matrix that is *zero on the diagonal*. It is however straightforward to check that these three conditions, although intuitive when considering the metric nearness problem (Brickell et al. 2008, §2), are not necessary for Algorithm (3.1) described by Brickell et al. (2008, §3) to work. This algorithm is not only valid for non-symmetric matrices H as pointed out by the authors themselves, but it is also applicable to matrices H with negative entries and non-zero diagonal entries. Problem (8) can thus be solved by replacing H by $-\frac{\lambda}{2}\Xi_k$ in Problem (9) regardless of the sign of the entries of Ξ.

Note that other approaches could be considered to minimize the dot product $\langle M, \Xi \rangle$ using alternative regularizers. Frangioni et al. (2005) propose for instance to handle linear programs in the intersection between the cone of metrics and the set of polyhedral constraints $\{M_{ik} + M_{kj} + M_{ij} \leq 2\}$ which defines what is known as the metric polytope.

The techniques presented above build upon a linear approximation of each function $G_{ij}(M)$ as $\langle M, \Xi_{ij} \rangle$ by selecting a particular matrix Ξ_{ij} such that $G_{ij}(M) \approx \langle M, \Xi_{ij} \rangle$. We propose to use a simple proxy for the optimal transport distance: the dot-product of M with a matrix that lies at the center of $U(r, c)$, as illustrated in Figure 5. We consider for such a center the *independence* table rc^T (Good 1963). The table rc^T, which is in $U(r, c)$ because $rc^T \mathbb{1}_d = r$ and $cr^T \mathbb{1}_d = c$, is also the maximal entropy table in $U(r, c)$, that is, the table which maximizes

$$h(X) = -\sum_{p,q=1}^{d} X_{pq} \log X_{pq}.$$

Using the independence table to approximate G_{ij}, that is using the approximation

$$\min_{X \in U(r_i, r_j)} \langle M, X \rangle \approx r_i^T M r_j,$$

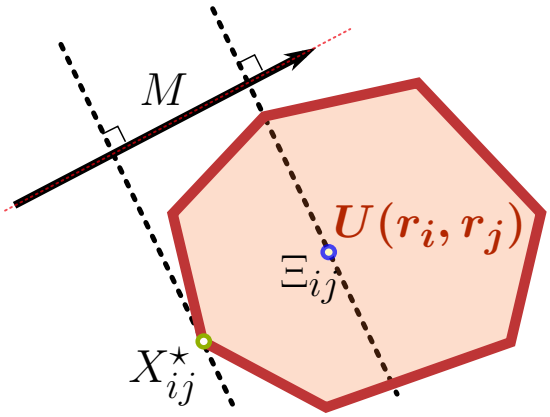


Figure 5: Schematic view of the approximation $\min_{X \in U(r_i,r_j)} \langle M, X \rangle \approx \langle M, \Xi_{ij} \rangle$ carried out when using a central transport table Ξ_{ij} instead of the optimal table $X^\star_{ij}$ to compare r_i and r_j.

provides us with a weighted center,

$$\Xi_k = \sum_{i=1}^{n} \sum_{j \in N_{ik}^- \cup N_{ik}^+} \omega_{ij} r_i r_j^T .$$

Note however that this approximation tends to overestimate substantially the distance between two similar histograms. Indeed, it is easy to check that $r^T M r$ is positive whenever r has positive entropy. In the case where all coordinates of r are equal to $1/d$, $r^T M r$ is $\|M\|_1/d^2$. To close this section, one may notice that several methods can be used to compute centers for polytopes such as $U(r, c)$, among which the Chebyshev center, the analytic center, or the center of the Löwner-John ellipsoid, all described by Boyd and Vandenberghe (2004, §8.4,§8.5). We have not considered these approaches because computing them involve, unlike the independence table proposed above, the resolution of large convex programs or LP's. Barvinok has, on the other hand, proposed recently a new center tailored specifically for transport polytopes, that he calls the typical table (2010). The typical table can be computed efficiently, both in theory and practice, as the result of a convex program of $2d$ variables (Barvinok 2010, p.523). Experimental results indicate that they perform very similarly to independent tables so we do not explore them further in this paper.

In summary, we propose in this section to approximate C_k by a linear function and compute its minimum in the intersection $\mathcal{M}_1$ of the l_1 unit sphere and the cone of metric matrices. This linear objective can be efficiently minimized using a set of tools proposed by Brickell et al. (2008) adapted to our problem. In order to propose such an approximation, we have used the *independence* tables as representative points of the polytopes $U(r_i, r_j)$. The successive steps of the computations that yield an initial point M_0 are given in Algorithm 3.

Algorithm 3 Initial Point M_0 to minimize C_k

Set $\Xi = 0$.
for $i \in \{1, \cdots, n\}$ **do**
  Compute the neighborhood sets N_{ik}^+ and N_{ik}^- of histogram r_i using an arbitrary distance, for example, the total variation distance.
  for $j \in N_{ik}^+ \cup N_{ik}^-$ **do**
    $\Xi \leftarrow \Xi + \omega_{ij} r_i r_j^T$.
  end for
end for
Set $M_0 \leftarrow \min_{M \in \mathcal{M}} \|M + \frac{\lambda}{2}\Xi\|_2$ (Brickell et al. 2008, Algorithm 3.1).
Output M_0. **optional**: regularize M_0 by setting $M_0 \leftarrow \lambda M_0 + (1-\lambda) M_{\mathbb{1}}$.

5. Related Work

We provide in this section an overview of other distances for histograms of features. We start by presenting simple distances on histograms and follow by presenting metric learning approaches.

5.1 Metrics on the Probability Simplex

Deza and Deza (2009, §14) provide an exhaustive list of metrics for probability measures, most of which apply to probability measures on $\mathbb{R}$ and $\mathbb{R}^d$. When narrowed down to distances for probabilities on unordered discrete sets—the dominant case in machine learning applications—Rubner et al. (2000, §2) propose to split such distances into two families: *bin-to-bin* distances and *cross-bin* distances. Let $r = (r_1, \ldots, r_d)^T$ and $c = (c_1, \ldots, c_d)^T$ be two histograms in the canonical simplex Σ_d.

Bin-to-bin distances only compare the d couples of bin-counts $(r_i, c_i)_{i=1..d}$ independently to form a distance between r and c. The Jensen-divergence, χ_2, Hellinger, total variation distances and more generally Csizar f-divergences (Amari and Nagaoka 2001, §3.2) all fall within this category. Notice that any of these divergences is known to work usually better for histograms than a straightforward application of the Euclidean distance as shown in our experiments or for instance by Chapelle et al. (1999, Table 4). This can be explained in theory using geometric (Amari and Nagaoka 2001, §3) or statistical arguments (Aitchison and Egozcue 2005).

Bin-to-bin distances are easy to compute and accurate enough to compare histograms when all d features are sufficiently distinct. When, on the contrary, some of these features are known to be similar, either because of statistical co-occurrence (e.g., the words `cat` and `kitty`) or through any other form of prior knowledge (e.g., pixel colors or amino-acid similarity) then a simple bin-to-bin comparison may not be accurate enough as argued by Rubner et al. (2000, §2.2). In particular, bin-to-bin distances are invariably large when they compare histograms with distinct supports, regardless of the fact that these two supports may in fact describe very similar features.

Cross-bin distances handle this issue by considering all d^2 possible pairs (r_i, c_j) of cross-bin counts to form a distance. The most simple cross-coordinate distance for general vectors

in $\mathbb{R}^d$ is arguably the Mahalanobis family of distances,

$$d^{\Omega}(x,y) = \sqrt{(x-y)^T \Omega (x-y)},$$

where Ω is a positive definite $d \times d$ matrix. The Mahalanobis distance between x and y can be interpreted as the Euclidean distance between Lx and Ly where L is a Cholesky factor of Ω or any square root of Ω. Learning such linear maps L or positive definite matrices Ω directly using labeled information has been the subject of a substantial amount of research in recent years. We briefly review this literature in the following section.

5.2 Mahalanobis Metric Learning

Xing et al. (2003), followed by Weinberger et al. (2006) and Davis et al. (2007) have proposed different algorithms to learn the parameters of a Mahalanobis distance. We refer to recent surveys by Kulis (2012) and Bellet et al. (2013) for more details on these approaches. These techniques define first a criterion and a feasible set of candidate matrices—either a positive semidefinite matrix Ω or a linear map L—to optimize the best parameter that fits best the data at hand. The criteria we propose in Section 3 are modeled along these ideas. Weinberger et al. (2006) were the first to consider criteria that only use nearest neighbors, which inspired in this work the proposal of C_k in Section 3.3.

We would like point out that Mahalanobis metric learning and ground metric learning have very little in common conceptually: Mahalanobis metric learning algorithms learn a $d \times d$ positive semidefinite matrix or a $m \times d$ linear operator L. Ground metric learning learns instead a $d \times d$ metric matrix M. The difference between Mahalanobis distances and optimal transport distances can be further highlighted by these simple identities:

$$d_{\mathrm{TV}}(r,c) = \frac{1}{2}\|r-c\|_1 = d_{M_1}(r,c), \quad d_2(r,c) = \|r-c\|_2 = d^I(r,c)$$

The relationship between the Euclidean distance and the family of Mahalanobis distances, in which the former is a trivial instance of the latter when Ω is set to the identity matrix, is analogous to that between the total variation distance and optimal transport distances, in which the former is also a trivial instance of the latter where all distances between features are uniformly set to 1. The two families of distances evolve in related albeit completely different sets of distances, just like the l_1 and l_2 norms describe different geometries. An illustration of this can be found in Figure 4 provided earlier in this paper, where the Euclidean and the total variation distances are compared with their parameterized counterparts. Both total variation and optimal transport distances have *piecewise linear* level sets, whereas the Euclidean and Mahalanobis distances have ellipsoidal level sets.

It is also worth mentioning that although Mahalanobis distances have been designed for general vectors in $\mathbb{R}^d$, and as a consequence can be applied to histograms, there is however, to our knowledge, no statistical theory which motivates their use on the probability simplex. This should be compared to the fact that there is a fairly large literature on optimal transport distances for probabilities, described by Villani (2003, §7) and references therein.

5.3 Metric Learning in the Probability Simplex

Lebanon (2006) has proposed to learn a bin-to-bin distance in the probability simplex using a parametric family of distances parameterized by a histogram $\lambda \in \Sigma_{d-1}$ defined as

$$d_\lambda(r,c) = \arccos\left(\sum_{i=1}^{d} \sqrt{\frac{r_i\lambda_i}{r^T\lambda}}\sqrt{\frac{c_i\lambda_i}{c^T\lambda}}\right).$$

This formula can be simplified by using the perturbation operator proposed by Aitchison (1986, p.46):

$$\forall r,\lambda \in \Sigma_{d-1}, \quad r \odot \lambda \stackrel{\text{def}}{=} \frac{1}{r^T\lambda}(r_1\lambda_1,\cdots,r_d\lambda_d)^T.$$

Aitchison argues that the perturbation operation can be naturally interpreted as an addition operation in the simplex. Using this notation, the family of distances $d_\lambda(r,c)$ proposed by Lebanon can be seen as the standard Fisher metric applied to perturbed histograms $r \odot \lambda$ and $c \odot \lambda$,

$$d_\lambda(r,c) = \arccos\langle\sqrt{r \odot \lambda}, \sqrt{c \odot \lambda}\,\rangle.$$

Using arguments related to the fact that a distance should vary according to the density of points described in a data set, Lebanon (2006) proposes to learn this perturbation λ in an unsupervised context, by only considering histograms but no other side-information.

More recently, Kedem et al. (2012) have proposed non-linear metric learning techniques, and focus more specifically on parameterized χ_2 distances defined as $d^P_{\chi_2}(r,c) = d_{\chi_2}(Pr,Pc)$ where P can be any stochastic matrix P with unit row sums. We also note that, a few months after the publication on the arxiv of an early version of our paper, Wang and Guibas (2012) have proposed an algorithm that is very similar to ours, with the notable difference that they do not take into account metric constraints for the ground metric.

6. Experiments

We provide in this section a few details on the practical implementation of Algorithms 1, 2 and 3. We follow by presenting empirical evidence that ground metric learning improves upon other state-of-the-art metric learning techniques when considered on normalized histograms of low dimensions, albeit at a substantial computational cost.

6.1 Implementation Notes

Algorithm 1 builds upon the computation of several optimal transport problems. We use the *CPLEX Matlab API* implementation of network flows to that effect. Using directly the API is faster than calling the *CPLEX matlab toolbox* or the *Mosek* solver. These benefits come from the fact that only the constraint vector in Problem (4) needs to be updated at each iteration of the first loop of Algorithm 1. We use the *metricNearness* toolbox released online by Suvrit Sra to carry out both the projections of each inner loop iteration of Algorithm 2 and the last step of Algorithm 3.

6.2 Distances Used in this Benchmark

We consider five distances in this benchmark. Three classic bin-to-bin distances, Mahalanobis distances with different learning schemes and the optimal transport distance coupled with ground metric learning. *Bin-to-bin distances* We consider the l_1, l_2 and Hellinger distances on histograms,

$$l_1(r,c) = \|r-c\|_1, \quad l_2(r,c) = \|r-c\|_2, \quad \mathcal{H}(r,c) = \|\sqrt{r} - \sqrt{c}\|_2,$$

where $\sqrt{r}$ is the vector whose coordinates are the squared roots of each coordinate of r.

6.2.1 Mahalanobis Distances

We use the publicly available implementations of LMNN (Weinberger and Saul 2009) and ITML (Davis et al. 2007) to learn Mahalanobis distances for each task. We run both algorithms with default settings, that is $k = 3$ for LMNN and $k = 4$ for ITML. We use these algorithms on the Hellinger representations $\{\sqrt{r_i}, i = 1, \ldots, n\}$ of all histograms originally in the training set using the element-wise square root. We have considered this representation because the Euclidean distance between the Hellinger representations of two histograms corresponds exactly to the Hellinger distance (Amari and Nagaoka 2001, p.57). Since the Mahalanobis distance builds upon the Euclidean distance, we argue that this representation is more adequate to learn Mahalanobis metrics in the probability simplex. This observation is confirmed in all of our experimental results, where Mahalanobis metric learning approaches perform consistently better with the Hellinger transformation (see for instance the results reported in Figure 7).

6.2.2 Optimal Transport Distances with Ground Metric Learning

We learn ground metrics using the following settings. The neighborhood parameter k is set to 3 to be directly comparable to the default parameter setting of ITML and LMNN. In each classification task, and for two images r_i and r_j, the corresponding weight ω_{ij} is set to $1/nk$ if both histograms come from the same class and to $-1/nk$ if they come from different classes. The subgradient stepsize t_0 of Algorithm 2 is set to $= 0.1$, guided by preliminary experiments and by the fact that, because of the normalization of the weights ω_{ij}, both the current iteration M_k in Algorithm 2 and subgradients γ_+ or γ_- all have the same 1-norms.

We carry out a minimum of 24 subgradient steps in each inner loop and set $q_{\max}$ to 80. Each inner loop is terminated when the objective does not progress more than 0.75% every 8 steps, or when q reaches $q_{\max}$. We carry out a maximum of 20 outer loop iterations. With these settings, the algorithm takes about 300 steps to converge (Figures 8 and 9), which, using a single Xeon 2.6Ghz core, 60 training points and $d = 128$ (the experimental setting considered below) takes about 900 seconds. The main computational bottleneck of the algorithm comes from the repeated computation of optimal transports. LMNN and ITML parameterized with default settings converge much faster, in about 2 and 30 seconds respectively.

6.3 Binary Classification

We study in this section the performance of ground metric learning when coupled with a nearest neighbor classifier on binary classification tasks generated with the Caltech-256 database.

6.3.1 Experimental Setting

We sample randomly 80 images for each of the 256 images classes[2] of the Caltech-256 database. Each image is represented as a normalized histogram of GIST features (Oliva and Torralba 2001, Douze et al. 2009), obtained using an implementation provided by the INRIA-LEAR team.[3] These features describe 8 edge directions at mid-resolution computed for each patch of a 4×4 grid on each image. Each feature histogram is of dimension $d = 8 \times 4 \times 4 = 128$ and subsequently normalized to sum to one.

We select randomly 1,000 distinct pairs of classes among the 256 classes available in the data set to form as many binary classification tasks. For each pair, we split the $80 + 80$ available points into $30+30$ points to train distance parameters and $50+50$ points to form a test set. This amounts to having $n = 60$ training points following the notations introduced in Section 3.1. We consider in the following κ nearest neighbors approaches. Note that the neighborhood size κ and the parameter k used in metric learning approaches need not be the same. In our experiments κ varies, whereas k is always kept fixed, as detailed in Section 6.2.

6.3.2 Results

The most important results of this experimental section are summarized in Figure 6, which displays, for all considered distances, their average recall accuracy on the test set and the average classification error using a κ-nearest neighbor classifier. These quantities are averaged over 1,000 binary classifications. In this figure, GML paired with the the optimal transport distance d_M is shown to provide, on average, the best performance with three different metrics: the leftmost plot considers retrieval performance for test points and shows that, for each point considered on its own, GML-EMD selects on average more training points from the same class as closest neighbors than any other distance. The performance gap between GML-EMD and competing distances increases significantly as the number of retrieved neighbors is itself increased. The middle plot displays the average error over all 1,000 tasks of a κ-nearest neighbor classification algorithm when considered with all distances for varying values of κ. The rightmost plot studies these errors in more detail for the case where the neighborhood parameter κ of nearest neighbors is 3. In this case too, GML combined with EMD fares significantly better than competing distances.

Figure 8 illustrates the empirical behavior of our descent algorithm. This plot displays 40 sample objective curves among the 1,000 computed to obtain the results above. The bumps that appear regularly on these curves correspond to the first update carried out after the linearization of the concave part of the objective. These results were obtained by setting the initial matrix to $M_{\mathbb{1}}$.

2. We do not consider the *clutter* class in our experiments.
3. Implementation can be found at `http://lear.inrialpes.fr/software`.

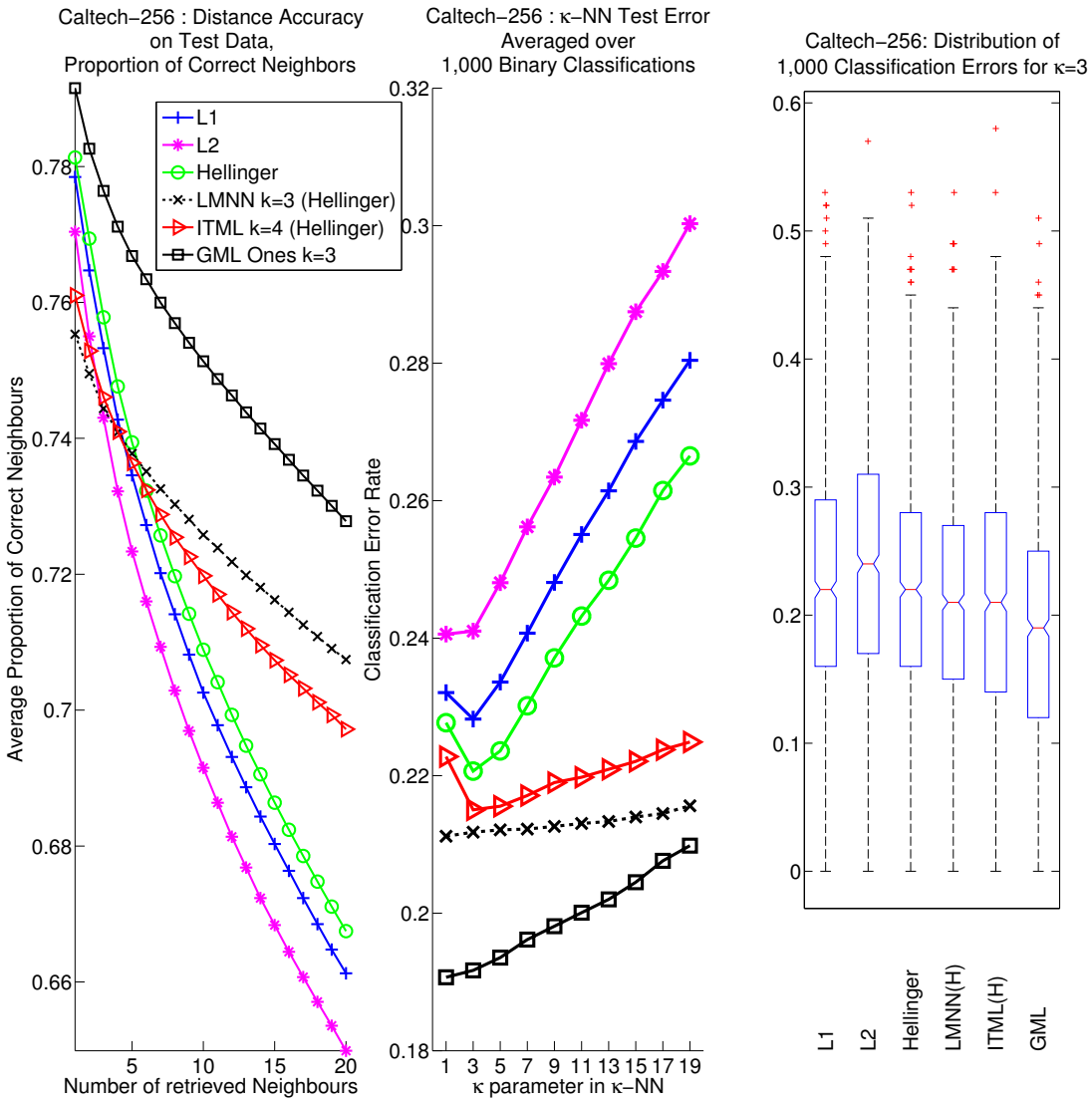


Figure 6: (left) Accuracy of each considered distance on the test set as measured by the average proportion, for each datapoint in the test set, of points coming from the same class within its κ nearest neighbors. These proportions were averaged over 1,000 binary classification problems randomly chosen among the $\binom{256}{2}$ possible. We use 40 test points from each class for each experiment, namely 80 test points. The ground metric in GML and Mahalanobis matrices in ITML and LMNN have been learned using a train set of $30 + 30$ points. (middle) κ-NN classification error using the same distances. These results show average κ-NN error over 1,000 classification tasks depending on the value of κ. A more detailed picture for the case $\kappa = 3$ is provided with boxplots of all $1,000$ errors (right).

Data Set	#Train	#Test	#Class	Feature	#Dim
20 News Group	600	19397	20	Topic Model (LDA)	100
Reuters	500	9926	10	Topic Model(LDA)	100
MIT Scene	800	800	8	SIFT	100
UIUC Scene	1500	1500	15	SIFT	100
OXFORD Flower	680	680	17	SIFT	100
CALTECH-101	3060	2995	102	SIFT	100

Table 1: Multiclass classification data sets and their parameters.

It is also worth mentioning as a side remark that the l_2 distance does not perform as well as the l_1 or Hellinger distances on these data sets, which validates our earlier statement that the Euclidean geometry is usually a poor choice to compare histograms directly. This intuition is further validated in Figure 7, where Mahalanobis learning algorithms are shown to perform significantly better when they use the Hellinger representation of histograms.

Finally, Figure 9 describes the evolution of the average *test* error for two initial ground metrics, $M_{\mathbb{1}}$ and that which builds upon independence tables (Algorithm 3). Two conclusions can be drawn from this plot: First, independence tables provide on average a better initialization of the algorithm if only the first iterations of the algorithm are taken into account. However, this advantage seems to vanish as the number of subgradient descent iterations increases. Second, our algorithm does not seem to suffer from overfitting on average, since the average error rate is a decreasing curve of the total number of iterations and does not seem to increase up to termination.

6.4 Multiclass Classification

We follow our experimental evaluation of ground metric learning by considering this time 6 multiclass classification data sets that consider text and image data.

6.4.1 Experimental Setting

The properties of the data sets and parameters used in our experiments are summarized in Table 1. The dimensions of the features have been kept low to ensure that the computation of optimal transports are tractable. We follow the recommended train/test splits for these data sets. If they are not provided, we split the data sets arbitrarily to form features using either LDA (Blei et al. 2003) or SIFT features (Lowe 1999). We then generate 5 random splits with the same balance to compute average accuracies over the entire data set.

6.4.2 Results

Figure 10 details the results for these 6 experiments, and show that GML coupled with EMD is at least equivalent or improves on the best techniques considered in our benchmark. These results also illustrate that the performance of Mahalanobis learning (LMNN in this case) is greatly improved by considering the Hellinger representation of histograms, and not their original representation as vectors of the simplex.

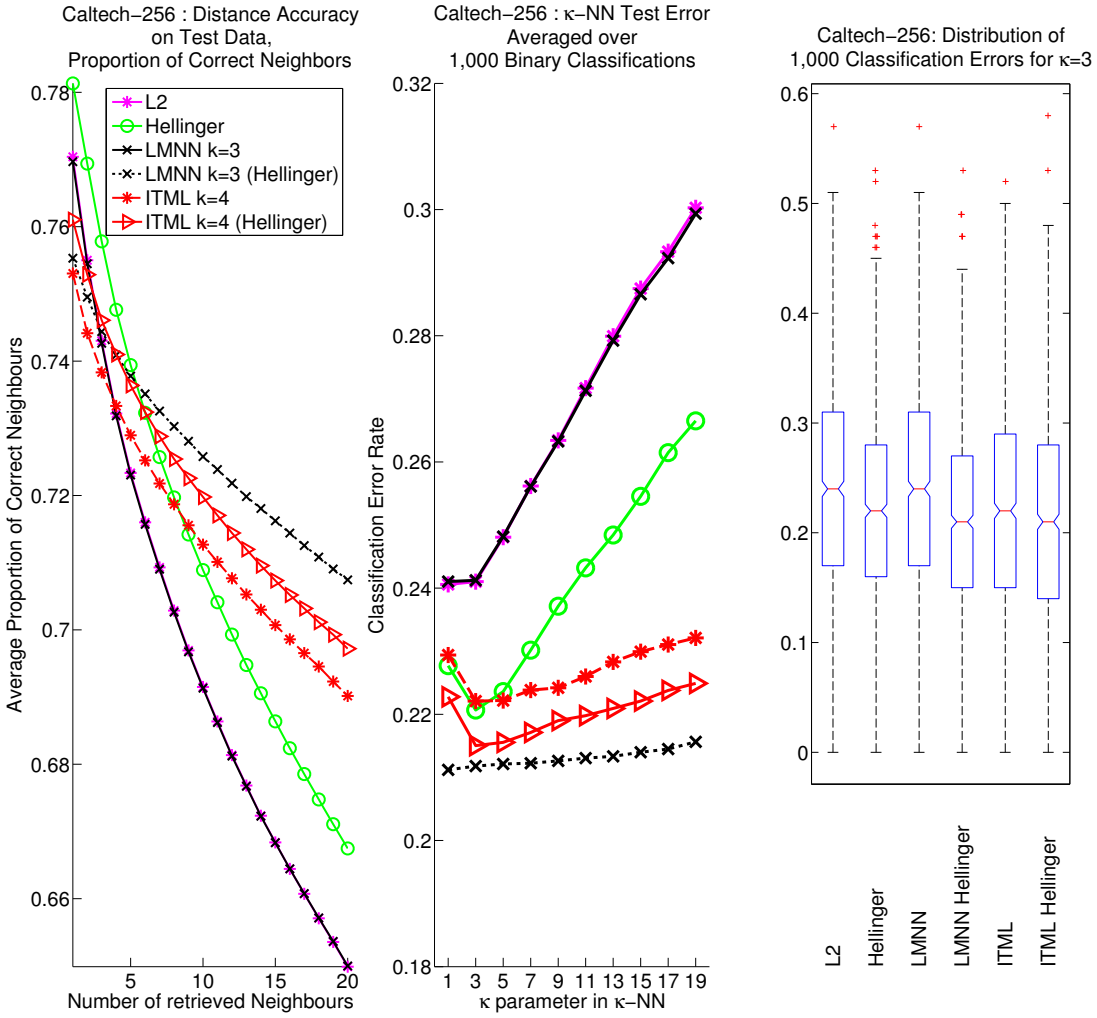


Figure 7: The experimental setting in this figure is identical to that of Figure 6, except that only two different versions of LMNN and ITML are compared with the Hellinger and Euclidean distances. This figure supports our claim in Section 6.2.1 that Mahalanobis learning methods work better using the Hellinger representation of histograms, $\{\sqrt{r_i},\, i = 1, \ldots, n\}$, rather than their straightforward representation in the simplex $\{r_i\}_{i=1,\ldots,n}$.

7. Conclusion and Future Work

We have proposed in this paper an approach to tune adaptively the unique parameter of optimal transport distances, the ground metric, given a training data set of histograms. This approach can be applied on any type of features, as long as a set of histograms along with side-information, typically labels, are provided for the algorithm to learn a good candidate for the ground metric. The algorithms proceeds with a projected subgradient descent to

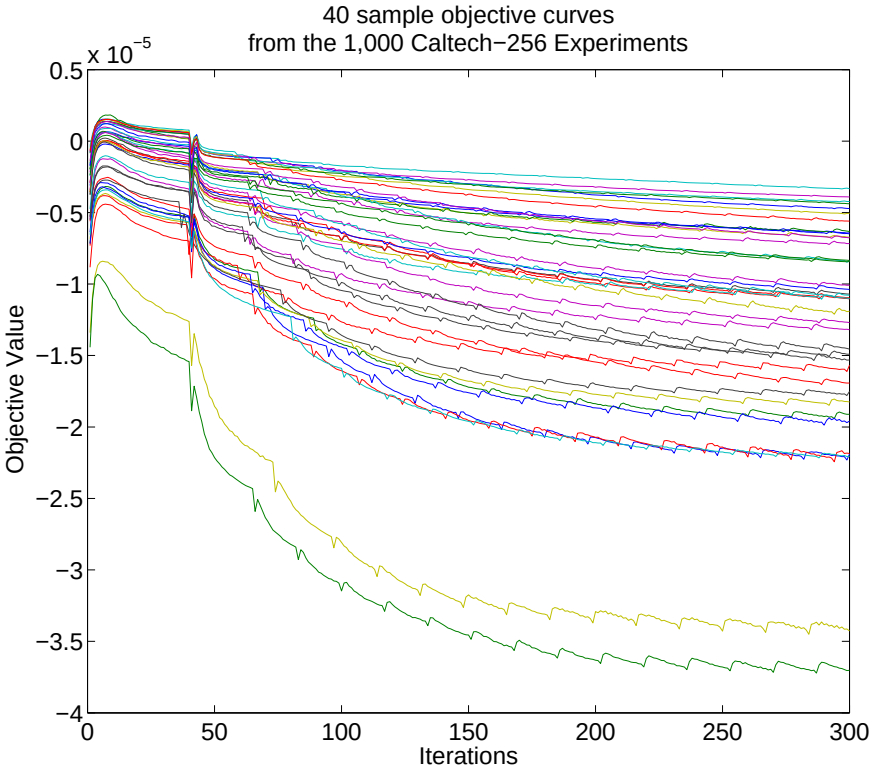


Figure 8: 40 sample objective curves randomly selected among the 1,000 binary classification tasks run on the Caltech-256 data set. The initial point used here is the matrix $M_{\mathbb{1}}$ of ones and zero diagonal. The very first bumps usually observed in the first iterations agree with our empirical findings on empirical test error displayed in Figure 9 which illustrate that the very first radients that are applied are usually not informative and result momentarily in an objective increase.

minimize approximately a criterion that is a difference of polyhedral convex functions. We propose two initial points to initialize this algorithm, and show that our approach provides, when compared to other competing distances, a superior average performance for a large set of image binary classification tasks using GIST features histograms, as well as different multiclass classification tasks. This improvement comes, however, with a heavy computational price tag.

Our benchmark experiments only contain low-dimensional descriptors. We chose such small dimensions because it is well known that optimal transport distances do not scale well for higher dimensions. That being said, the problem of speeding up the computation of optimal transport distances by considering restrictions on ground metrics has attracted significant attention. Ling and Okada (2007), Gudmundsson et al. (2007), Pele and Werman (2009), Ba et al. (2011) have all recently argued that this computation can be dramatically

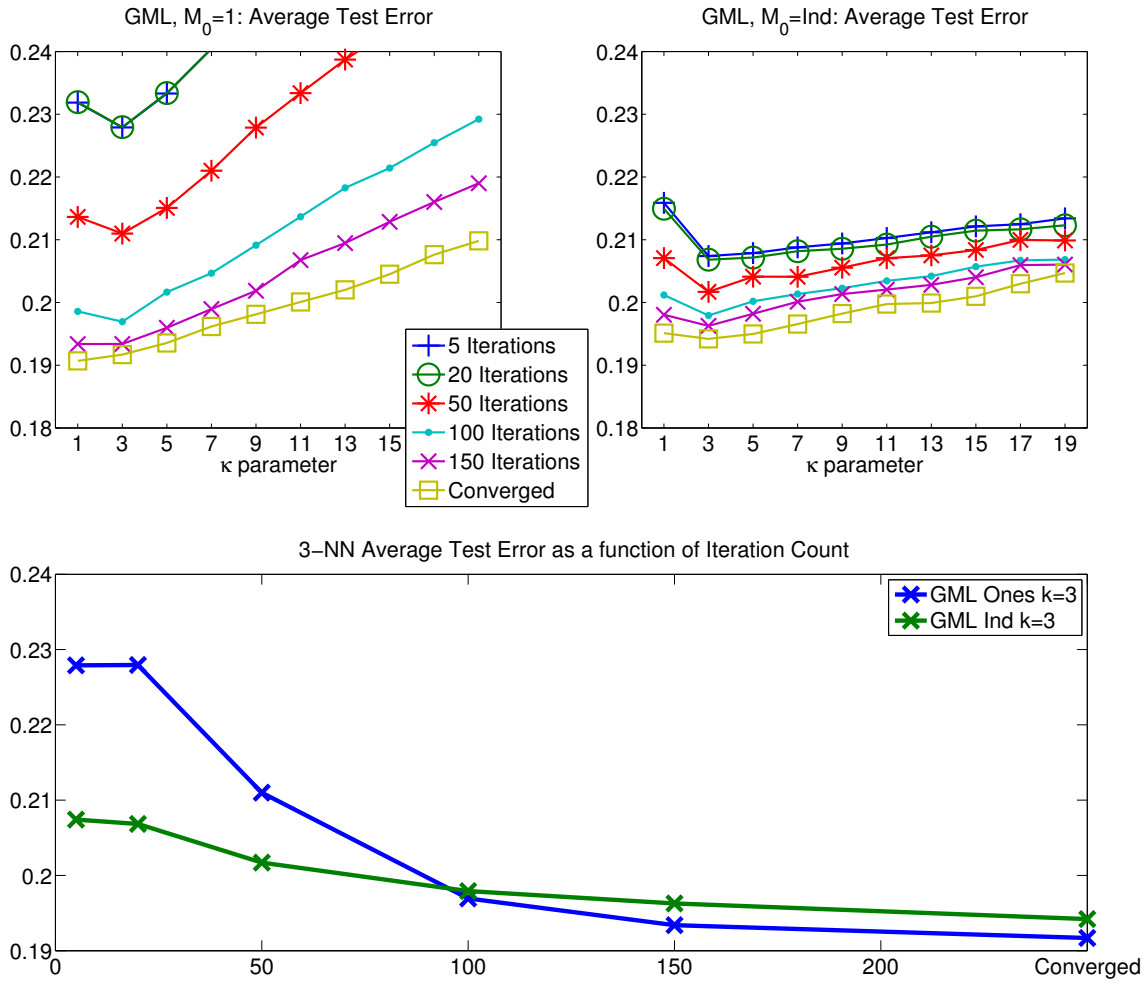


Figure 9: Average κ-nearest neighbor *test* error for GML using either the matrix of ones (top left) or the independent table (top right) described in Section 4.3. As can be seen for $\kappa = 3$ (bottom), initializing the algorithm with $M_{\mathbb{1}}$ performs worse than independence tables for a low iteration count. Yet this competitive advantage is reversed above a few iterations, as the algorithm converges. This figure also seems to indicate that, on average, the algorithm does not overfit the data since the average test error seems to decrease monotonically with the number of iterations, and becomes flat after 200 iterations. The experimental setting is identical to that of Figure 6.

sped up when the ground metric matrix has a certain structure. For instance, Pele and Werman (2009) have shown that the computational speed of earth mover's distances can be significantly accelerated when the ground metric is thresholded above a certain level. Ground metrics that follow such constraints are attractive because they result in transport

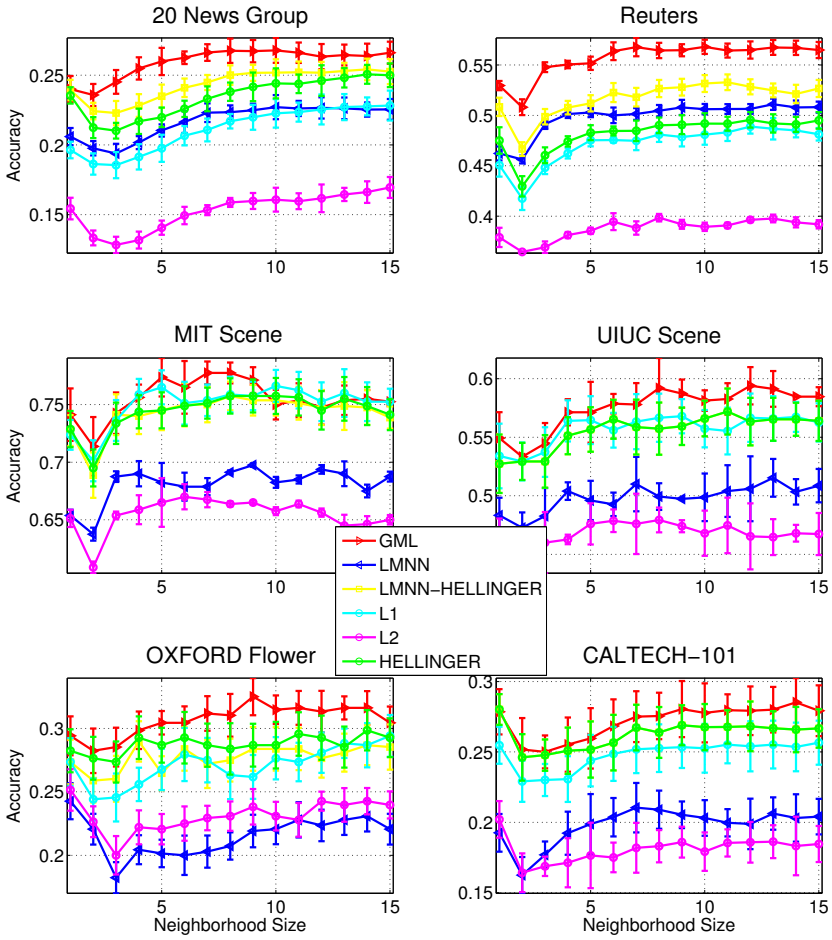


Figure 10: κ-nearest neighbor performance for different distances on multi-class problems. Performance is averaged over 5 repeats, whose variability is illustrated with error bars. Errors are reported over varying κ nearest neighbor parameters. Our benchmark considers three classical distances, l_1, l_2 and Hellinger, and their respective learned counterparts: GML paired with the transport distance initialized with the matrix $M_{\mathbb{1}}$, classic LMNN and LMNN on the Hellinger representation.

problems which are provably faster to compute. Our work in this paper suggests on the other hand that the content (and not the structure) of the ground metric can be learned to improve classification accuracy. We believe that the combination of these two viewpoints could result in optimal transport distances that are both adapted to the task at hand and fast to compute. A strategy to achieve both goals would be to enforce such structural constraints on candidate metrics M when looking for minimizers of criteria C_k. We also believe that the recent proposal of Sinkhorn distances (Cuturi 2013) may provide the necessary speed-ups to make our approach more scaleable regardless of the structure of the ground metric.

Acknowledgments

The authors would like to thank anonymous reviewers and the editor for stimulating comments. MC would like to thank Zaid Harchaoui and Alexandre d'Aspremont for fruitful discussions, as well as Tam Le for his help in preparing some of the experiments.

Appendix A.

Proof (Theorem 1) Symmetry and definiteness of the distance are easy to prove: since M has a null diagonal, $d_M(x,x) = 0$, with corresponding optimal transport matrix $X^\star = \mathrm{diag}(x)$; by the positivity of all off-diagonal elements of M, $d_M(x,y) > 0$ whenever $x \neq y$; by symmetry of M, d_M is itself a symmetric function in its two arguments. To prove the triangle inequality, Villani (2003, Theorem 7.3) uses the gluing lemma. We provide here a self-contained version of this proof which provides an explicit formulation for the gluing lemma in the discrete case. Let $x, y, z \in \Sigma_d$. Let P and Q be two optimal solutions of the transport problems between x and y, and y and z respectively. Let S be the $d \times d \times d$ tensor whose coefficients are defined as

$$s_{ijk} \stackrel{\text{def}}{=} \frac{p_{ij} q_{jk}}{y_j},$$

for all indices j such that $y_j > 0$. For indices j such that $y_j = 0$, the corresponding values s_{ijk} are set to 0. S is a probability measure on $\{1, \ldots, d\}^3$, as a direct consequence of the fact that the $d \times d$ matrix $S_{i\cdot k} \stackrel{\text{def}}{=} [\sum_j s_{ijk}]_{ik}$ is a transport matrix between x and z and thus sums to 1. Indeed,

$$\sum_i \sum_j s_{ijk} = \sum_j \sum_i \frac{p_{ij} q_{jk}}{y_j} = \sum_j \frac{q_{jk}}{y_j} \sum_i p_{ij} = \sum_j \frac{q_{jk}}{y_j} y_j = \sum_j q_{jk} = z_k \text{ (column sums)},$$
$$\sum_k \sum_j s_{ijk} = \sum_j \sum_k \frac{p_{ij} q_{jk}}{y_j} = \sum_j \frac{p_{ij}}{y_j} \sum_k q_{jk} = \sum_j \frac{p_{ij}}{y_j} y_j = \sum_j p_{ij} = x_i \text{ (row sums)}.$$

To obtain the triangle inequality, notice that $S_{i\cdot k}$ being a matrix of $U(x,z)$ we can write:

$$\begin{aligned}
d_M(x,z) &= \min_{X \in U(x,z)} \langle X, M \rangle \\
&\leq \langle S_{i\cdot k}, M \rangle = \sum_{ik} m_{ik} \sum_j \frac{p_{ij} q_{jk}}{y_j} \leq \sum_{ijk} (m_{ij} + m_{jk}) \frac{p_{ij} q_{jk}}{y_j} \\
&= \sum_{ijk} m_{ij} \frac{p_{ij} q_{jk}}{y_j} + m_{jk} \frac{p_{ij} q_{jk}}{y_j} \\
&= \sum_{ij} m_{ij} p_{ij} \sum_k \frac{q_{jk}}{y_j} + \sum_{jk} m_{jk} q_{jk} \sum_i \frac{p_{ij}}{y_j} \\
&= \sum_{ij} m_{ij} p_{ij} + \sum_{jk} m_{jk} q_{jk} = d_M(x,y) + d_M(y,z),
\end{aligned}$$

where we have used the triangle inequality for M at the end of the second line. ■

References

R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, 1993.

J. Aitchison. *The Statistical Analysis of Compositional Data*. Chapman & Hall, 1986.

J. Aitchison and J. Egozcue. Compositional data analysis: Where are we and where should we be heading? *Mathematical Geology*, 37(7):829–850, 2005.

S.-I. Amari and H. Nagaoka. *Methods of Information Geometry*. AMS vol. 191, 2001.

K. Ba, H. Nguyen, H. Nguyen, and R. Rubinfeld. Sublinear time algorithms for earth movers distance. *Theory of Computing Systems*, 48(2):428–442, 2011.

A. Barvinok. What does a random contingency table look like? *Combinatorics, Probability and Computing*, 19(04):517–539, 2010.

A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *arXiv:1306.6709*, 2013.

D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

D. Blei and J. Lafferty. Topic models. *Text Mining: Classification, Clustering, and Applications*, 10:71, 2009.

D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

J. Brickell, I. Dhillon, S. Sra, and J. Tropp. The metric nearness problem. *SIAM Journal of Matrix Analysis and Applications*, 30(1):375–396, 2008.

R. A. Brualdi. *Combinatorial Matrix Classes*, volume 108. Cambridge University Press, 2006.

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

M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems 26*, pages 2292–2300. 2013.

J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 209–216. ACM, 2007.

M. Deza and E. Deza. *Encyclopedia of Distances*. Springer Verlag, 2009.

P. Diaconis and B. Efron. Testing for independence in a two-way table: new interpretations of the chi-square statistic. *The Annals of Statistics*, 13(3):845–913, 1985.

M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid. Evaluation of GIST descriptors for web-scale image search. In *Proceedings of the ACM International Conference on Image and Video Retrieval*. Article 19, ACM, 2009.

L. Ford and Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

A. Frangioni, A. Lodi, and G. Rinaldi. New approaches for optimizing over the semimetric polytope. *Mathematical Programming*, 104(2):375–388, 2005.

I. Good. Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *The Annals of Mathematical Statistics*, pages 911–934, 1963.

J. Gudmundsson, O. Klein, C. Knauer, and M. Smid. Small manhattan networks and algorithmic applications for the earth movers distance. In *Proceedings of the 23rd European Workshop on Computational Geometry*, pages 174–177, 2007.

T. Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory, and Algorithms*. Kluwer Academic Publishers, 2002.

L. Kantorovich and G. Rubinshtein. On a space of totally additive functions. *Vestn Lening. Univ.*, 13:52–59, 1958.

H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the 20th International Conference on Machine Learning*, pages 321–328, 2003.

D. Kedem, S. Tyree, K. Weinberger, F. Sha, and G. Lanckriet. Non-linear metric learning. In *Advances in Neural Information Processing Systems 25*, pages 2582–2590, 2012.

B. Kulis. Metric learning: A survey. *Foundations & Trends in Machine Learning*, 5(4): 287–364, 2012.

S. Lauritzen. *Lectures on Contingency Tables*. Aalborg Univ. Press, 1982.

G. Lebanon. Metric learning for text documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):497–508, 2006.

C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: a string kernel for svm protein classific ation. In *Proceedings of the Pacific Symposium on Biology 2002*, pages 564–575, 2002.

E. Levina and P. Bickel. The earth mover's distance is the Mallows distance: some insights from statistics. In *Proceedings of the Eighth IEEE International Conference on Computer Vision*, volume 2, pages 251–256. IEEE, 2001.

H. Ling and K. Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 840–853, 2007.

D. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150 –1157 vol.2, 1999.

C. Mallows. A note on asymptotic joint normality. *The Annals of Mathematical Statistics*, pages 508–515, 1972.

A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

O. Pele and M. Werman. Fast and robust earth mover's distances. In *Proceedings of the International Conference on Computer Vision'09*, 2009.

S. Rachev. *Probability Metrics and the Stability of Stochastic Models*. Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley, 1991.

S. Rachev and L. Rüschendorf. *Mass Transportation Problems: Theory*, volume 1. Springer Verlag, 1998.

T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

Y. Rubner, L. Guibas, and C. Tomasi. The earth movers distance, multi-dimensional scaling, and color-based image retrieval. In *Proceedings of the ARPA Image Understanding Workshop*, pages 661–668, 1997.

Y. Rubner, C. Tomasi, and L. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40, 2000.

M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1): 11–32, 1991.

A. Vershik. Kantorovich metric: initial history and little-known applications. *Journal of Mathematical Sciences*, 133(4):1410–1417, 2006.

C. Villani. *Topics in Optimal Transportation*, volume 58. AMS Graduate Studies in Mathematics, 2003.

F. Wang and L. J. Guibas. Supervised earth movers distance learning and its computer vision applications. In *Computer Vision–ECCV 2012*, pages 442–455. Springer, 2012.

K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009.

K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems 18*, pages 1473–1480, 2006.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, 2003.

Journal of Machine Learning Research 15 (2014) 565-593 Submitted 11/12; Revised 10/13; Published 2/14

Link Prediction in Graphs with Autoregressive Features

Emile Richard EMILERIC@STANFORD.EDU
Department of Electrical Engineering
Stanford University - Packard 239
Stanford, CA 94304

Stéphane Gaïffas STEPHANE.GAIFFAS@CMAP.POLYTECHNIQUE.FR
CMAP - Ecole Polytechnique
Route de Saclay
91128 Palaiseau Cedex, France

Nicolas Vayatis NICOLAS.VAYATIS@CMLA.ENS-CACHAN.FR
CMLA - ENS Cachan
UMR CNRS No. 8536
61, avenue du Président Wilson
94 235 Cachan cedex, France

Editor: Tong Zhang

Abstract

In the paper, we consider the problem of link prediction in time-evolving graphs. We assume that certain graph features, such as the node degree, follow a vector autoregressive (VAR) model and we propose to use this information to improve the accuracy of prediction. Our strategy involves a joint optimization procedure over the space of adjacency matrices and VAR matrices. On the adjacency matrix it takes into account both sparsity and low rank properties and on the VAR it encodes the sparsity. The analysis involves oracle inequalities that illustrate the trade-offs in the choice of smoothing parameters when modeling the joint effect of sparsity and low rank. The estimate is computed efficiently using proximal methods, and evaluated through numerical experiments.

Keywords: graphs, link prediction, low-rank, sparsity, autoregression

1. Introduction

Forecasting systems behavior with multiple responses has been a challenging issue in many contexts of applications such as collaborative filtering, financial markets, or bioinformatics, where responses can be, respectively, movie ratings, stock prices, or activity of genes within a cell. Statistical modeling techniques have been widely investigated in the context of multivariate time series either in the multiple linear regression setup by Breiman and Friedman (1997) or with autoregressive models by Tsay (2005). More recently, kernel-based regularized methods have been developed for multitask learning by Evgeniou et al. (2005) and Andreas et al. (2007). These approaches share the use of the correlation structure among input variables to enrich the prediction on every single output. Often, the correlation structure is assumed to be given or it is estimated separately. A discrete encoding of correlations between variables can be modeled as a graph so that learning the dependence

structure amounts to performing graph inference through the discovery of uncovered edges on the graph. The latter problem is interesting *per se* and it is known as the problem of link prediction where it is assumed that only a part of the graph is actually observed, see the paper by Liben-Nowell and Kleinberg (2007) and Kolar and Xing (2011). This situation occurs in various applications such as recommender systems, social networks, or proteomics, and the appropriate tools can be found among matrix completion techniques, see for instance the papers by Srebro et al. (2005), Candès and Tao (2009) and Abernethy et al. (2009). In the realistic setup of a time-evolving graph, matrix completion was also used and adapted by Richard et al. (2010) to take into account the dynamics of the features of the graph. The estimation of a VAR model for node degrees (that are linear graph features) has been considered by Zhang et al. (2011) and successfully applied to customer valuation, and to measure network effect in user generated content market places. Note also that sparse autoregressive models are also considered by Davis et al. (2012) and Nardi and Rinaldo (2011).

In this paper, we study the prediction problem where the observation is a sequence of graphs represented through their adjacency matrices $(A_t)_{0\leq t\leq T}$ and the goal is to predict A_{T+1}. This prediction problem arises in recommender systems, where the purchases or preference declarations are registered over time. In this context, users and products can be modeled as the nodes of a bipartite graph, while purchases or clicks are modeled as edges. In functional genomics and systems biology, estimating regulatory networks in gene expression can be performed by modeling the data as graphs. In this setting, fitting predictive models is a natural way for estimating evolving networks in these contexts, see the paper by Shojaie et al. (2011). A large variety of methods for link prediction only consider prediction from a single instantaneous snapshot of the graph. This includes heuristics: measures of node neighbourhoods are considered by Liben-Nowell and Kleinberg (2007), Lü and Zhou (2011) and Sarkar et al. (2010), matrix factorization by Koren (2008), diffusion by see Myers and Leskovec (2010) and probabilistic methods by Taskar et al. (2003). More recently, some works have investigated the use of sequences of observations of the graph to improve the prediction, such as regression on features extracted from the graphs by Richard et al. (2010), matrix factorization by Koren (2010), continuous-time regression by Vu et al. (2011) or non-parametric models by Sarkar et al. (2012). An hybrid approach to dynamic link prediction is considered by Huang and Lin (2009), based on a mixture of the static approach by Liben-Nowell and Kleinberg (2007) and an individual ARIMA modeling of the links evolution.

The framework of the current paper is somehow related to compressed sensing introduced by Donoho (2006) and Candès and Wakin (2008). In fact, due to stationarity assumptions, the amount of available information is very small compared to the task of predicting the quadratically many potential edges of the graph. Therefore penalization terms that encourage both sparsity and low-rank of related matrices are used to recover the edges of the graph. In the static setup, these two effects have been previously combined by Richard et al. (2012b) for the estimation of sparse and low-rank matrices, the rationale being that graphs containing cliques have block-diagonal adjacency matrices that are simultaneously sparse and low-rank. Key elements in deriving theoretical results are tools from the theory of compressed sensing, developed by Candès and Tao (2005), Bickel et al. (2009), Koltchinskii et al. (2011) and in particular the Restricted Eigenvalue of Koltchinskii (2009a), Koltchinskii (2009b) and Bickel et al. (2009). Our main results are oracle inequalities under the general

assumption that the innovation process of the VAR is a martingale increment sequence with sub-gaussian tails. These oracle inequalities prove that our procedure achieves a trade-off in the calibration of smoothing parameters that balances the sparsity and the rank of the adjacency matrix. A preliminary version of this work can be found in a previous work by Richard et al. (2012a).

The rest of this paper is organized as follows. In Section 2, we describe the general setup of this study with the main assumptions. In Section 2.3, we formulate a regularized optimization problem which aims at jointly estimating the autoregression parameters and predicting the graph. In Section 3, we provide theoretical guarantees for the joint estimation-prediction by providing oracle inequalities. In Section 4 we provide an efficient algorithm for solving the optimization problem and show empirical results that illustrate our approach. The proofs are provided in Appendix.

2. Modeling Low-Rank Graphs Dynamics with Autoregressive Features

We first introduce the main notations used in the paper.
Matrix norms and entrywise matrix operations. Denote by A a matrix. In the sequel, the notations $\|A\|_F$, $\|A\|_p$, $\|A\|_\infty$, $\|A\|_*$ and $\|A\|_{\text{op}}$ stand, respectively, for the Frobenius norm of A, the entry-wise ℓ_p norm, the entry-wise ℓ_∞ norm, the trace-norm (or nuclear norm, given by the sum of the singular values) and operator norm (the largest singular value) of A. Given matrices A and B, we denote by $\langle A, B\rangle = \text{tr}(A^\top B)$ the Euclidean matrix product. A vector in $\mathbb{R}^d$ is always understood as a $d \times 1$ matrix. We denote by $\|A\|_0$ the number of non-zero elements of A. The product $A \circ B$ between two matrices with matching dimensions stands for the entry-wise product between A and B (also called Hadamard product). The matrix $|A|$ contains the absolute values of entries of A. The matrix $(M)_+$ is the entry-wise positive part of the matrix M, and $\text{sign}(M)$ is the sign matrix associated to M with the convention $\text{sign}(0) = 0$.
SVD and projections. If A is a $n \times n$ matrix with rank r, we write its Singular Value Decomposition (SVD) as $A = U\Sigma V^\top = \sum_{j=1}^r \sigma_j u_j v_j^\top$ where $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r)$ is a $r \times r$ diagonal matrix containing the non-zero singular values of A in decreasing order, and $U = [u_1, \ldots, u_r]$, $V = [v_1, \ldots, v_r]$ are $n \times r$ matrices with columns given by the left and right singular vectors of A. The projection matrix onto the space spanned by the columns (resp. rows) of A is given by $P_U = UU^\top$ (resp. $P_V = VV^\top$). The operator $\mathcal{P}_A : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ given by $\mathcal{P}_A(B) = P_U B + BP_V - P_U B P_V$ is the projector onto the linear space spanned by the matrices $u_k x^\top$ and $y v_k^\top$ for $1 \le j, k \le r$ and $x, y \in \mathbb{R}^n$. The projector onto the orthogonal space is given by $\mathcal{P}_A^\perp(B) = (I - P_U)B(I - P_V)$. We also use the notation $a \vee b = \max(a, b)$.

2.1 Working Assumptions

Our approach is based on a number of beliefs which we translate into mathematical assumptions:

- Low-rank of adjacency matrices A_t

  This reflects the presence of highly connected groups of nodes such as communities in social networks, or product categories and groups of loyal/fanatic users in a market place data, and is sometimes motivated by the small number of factors that explain

nodes interactions. We will not make an explicit assumption in the paper but the results we obtain will be meaningful in the specific case where rank is small compared to the dimension.

- Autoregressive linear features (VAR models)

  We assume that intrinsic features of the graph can explain most of the information contained in the graph, and that these features are evolving with time. Our approach considers the simplest assumption on the dynamics over time of these features and we assume a Vector Autoregressive Linear Regression model that is described in the next subsection.

- Sub-gaussian noise process

  A probabilistic framework is considered in order to describe performance under the form of oracle inequalities and we propose to specify the distribution of the discrepancy between the VAR model and the actual observations with a a sub-gaussian tail behavior. This assumption will be formulated below in Section 3.

The first two items correspond to modeling assumptions which partly capture observations made on real-life data. The third item is a technical assumption used in the proofs.

2.2 An Autoregressive Linear Model for Graph Features

Feature map. We consider a list of graph features encoded through a linear map of the adjacency matrix with $\omega : \mathbb{R}^{n\times n} \to \mathbb{R}^d$ defined by:

$$\omega(A) = \big[\langle \Omega_1, A\rangle, \cdots, \langle \Omega_d, A\rangle\big]^\top, \tag{1}$$

where $\{\Omega_i\}_{1\le i\le d}$ is a set of $n \times n$ matrices. These matrices could be either deterministic or random in our theoretical analysis, but we take them deterministic for the sake of simplicity. An example of linear features is the vector of node degrees, that is, the number of edges connected to each node. The degree can be computed from the adjacency matrix using the linear function $\omega : A \mapsto A\mathbf{1}$ or $\omega : A \mapsto A^\top\mathbf{1}$ respectively for the right and left nodes degrees, where $\mathbf{1}$ denotes the vector with all coordinates equal to 1 of the appropriate length. Other (linear) measures of popularity are considered in social and e-commerce networks, such as the sum of the weights of incident edges if there is some graduation in the strength of connection between nodes. Note that nonlinear features, such as the count of the number of cycles of length k ($k = 3, 4, \cdots$) through each node, may be relevant in real world applications. Such features involve, for instance, the diagonal elements of A^k. An extensive study of this very interesting case is beyond the scope of the present paper.
VAR model. We consider a linear model for the evolution of $\omega(A)$ over time.

Assumption 1 *The vector time series $\{\omega(A_t)\}_{t\ge 0}$ has autoregressive dynamics, given by a VAR (Vector Auto-Regressive) model:*

$$\omega(A_{t+1}) = W_0^\top \omega(A_t) + N_{t+1},$$

where $W_0 \in \mathbb{R}^{d\times d}$ is an unknown sparse matrix and $\{N_t\}_{t\ge 0}$ is a sequence of noise vectors in $\mathbb{R}^d$.

In the sequel, we shall use the following compact notations:

$$\mathbf{X}_{T-1} = \big[\omega(A_0), \ldots, \omega(A_{T-1})\big]^\top \quad \text{and} \quad \mathbf{X}_T = \big[\omega(A_1), \ldots, \omega(A_T)\big]^\top,$$

which are both $T \times d$ matrices, we can write this model in matrix form:

$$\mathbf{X}_T = \mathbf{X}_{T-1} W_0 + \mathbf{N}_T,$$

where $\mathbf{N}_T = [N_1, \ldots, N_T]^\top$.

2.3 Simultaneous Prediction and Estimation through Regularized Optimization

Optimization problem formulation. We now introduce the optimization problem which will account for both the prediction task (anticipate the appearance of new edges in the graph) and the modeling choices which are supposed to reflect phenomena observed on real data (smooth evolution of graph features). We consider that snapshots of the graph (and therefore also the corresponding features) are available at times $1, \ldots, T$ and we want to predict links which will appear at the next instant $T+1$. In order to fulfill this double objective, we combine two regularized problems in an additive fashion based on two terms:

1. First objective - data-fitting term for weight vector W with sparsity-enforcing penalty

$$J_1(W) = \frac{1}{T}\|\mathbf{X}_T - \mathbf{X}_{T-1}W\|_F^2 + \kappa\|W\|_1, \tag{2}$$

   where $\kappa > 0$ is a smoothing parameter.

2. Second objective - data-fitting term for the features of the adjacency matrix A with mixed penalty enforcing both sparsity and low-rank

$$J_2(A, W) = \frac{1}{d}\|\omega(A) - W^\top \omega(A_T)\|_2^2 + \tau\|A\|_* + \gamma\|A\|_1,$$

   where $\tau, \gamma > 0$ are smoothing parameters.

The resulting penalized criterion will be the main topic of the present paper. It is the sum of the two partial objectives J_1 and J_2, and is jointly convex with respect to A and W:

$$\mathcal{L}(A, W) \doteq \frac{1}{T}\|\mathbf{X}_T - \mathbf{X}_{T-1}W\|_F^2 + \kappa\|W\|_1 + \frac{1}{d}\|\omega(A) - W^\top\omega(A_T)\|_2^2 + \tau\|A\|_* + \gamma\|A\|_1. \tag{3}$$

Rationale. As shown by the introduction of the two functionals, our approach pursues a double goal. On the one hand, the data-fitting term on W in J_1 aims at an estimate on the past data of the weight factor in the autoregressive modeling setup according to Assumption 1 and under a sparsity constraint. On the other hand, the link prediction goes through the estimation of a matrix $A = A_{T+1}$ which should be sparse and low-rank simultaneously. Hence, the second functional J_2 involves a mixed penalty of the form $A \mapsto \tau\|A\|_* + \gamma\|A\|_1$, with τ, γ smoothing parameters. Such a combination of ℓ_1 and trace-norm was already studied by Gaïffas and Lecué (2011) for the matrix regression model, and by Richard et al. (2012b) for the prediction of an adjacency matrix. This mixed norm combines the benefits of each of the two norms and is well suited for estimating simultaneously sparse

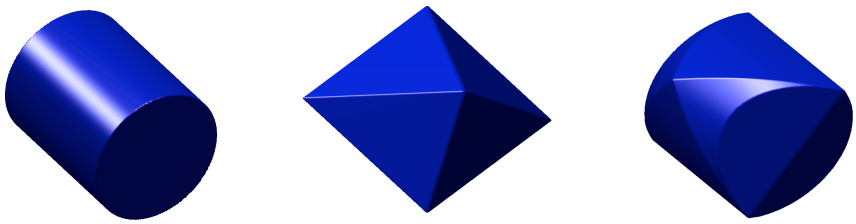


Figure 1: Unit balls for the trace norm (left), ℓ_1 (middle) and the mixed $X \mapsto \|X\|_* + \|X\|_1$ norm (right). The norms where computed on the set of 2×2 symmetric matrices that can be identified to $\mathbb{R}^3$.

and low-rank matrices. In Figure 1 we illustrated the unit balls for the three norms ℓ_1, trace-norm and the ℓ_1 + trace norm. The key observation is that the ball of the mixed norm has singularities at the points where each of the two other balls are singular, but the singularities get sharper at points where both norms are singular, namely on the matrices that are sparse and low-rank at the same time.

The set of sparse and low-rank matrices obtained by minimizing an objective including this mixed norm contains matrices that can be written in a block-diagonal or overlapping block-diagonal form, up to permutations of rows and columns. These matrices can be interpreted as adjacency matrices of networks containing highly connected groups of nodes and therefore are of particular interest for prediction and denoising applications in graph data and in covariance matrix estimation. Here we extend the approach developed by Richard et al. (2012b) for the time-dependent setting by considering data-fitting measures which ensure that the features of the next graph $\omega(A_{T+1})$ are close to $W^\top \omega(A_T)$.

Search space and general scheme of the estimation procedure. We shall consider the case where the optimization domain consists of the cartesian product of convex cones $\mathcal{A}$ and $\mathcal{W}$ such that $\mathcal{A} \subset \mathbb{R}^{n \times n}$ and $\mathcal{W} \subset \mathbb{R}^{d \times d}$. The joint estimation-prediction procedure is then defined by

$$(\hat{A}, \hat{W}) \in \operatorname*{arg\,min}_{(A,W) \in \mathcal{A} \times \mathcal{W}} \mathcal{L}(A, W). \tag{4}$$

It is natural to take $\mathcal{W} = \mathbb{R}^{d \times d}$ and $\mathcal{A} = (\mathbb{R}_+)^{n \times n}$ since there is no *a priori* on the values of the true VAR model matrix W_0, while the entries of the matrix A_{T+1} must be positive. Table 1 summarizes the methodology in a scheme where the symbols $\downarrow_\omega$ represent the feature extraction procedure through the map $\omega : \mathbb{R}^{n \times n} \to \mathbb{R}^d$. The prediction in the feature space is represented by $\to_W$, and is handled in practice by the least squares regression on W. Finally, the symbol $\uparrow$ that maps the predicted feature vector $\widehat{\omega(A_{T+1})}$ to $\widehat{A_{T+1}}$ represents the inverse problem that is solved through the regression penalized by the mixed penalization.

2.4 An Overview of Main Results

The central contribution of our work is to provide bounds on the prediction error under a Restricted Eigenvalue (RE) assumption on the feature map. The main result can be summarized as follows: the prediction error and the estimation error can be simultaneously bounded by the sum of three terms that involve homogeneously (a) the sparsity, (b) the

A_0	A_1	$\cdots$	A_T		$\widehat{A_{T+1}}$	Observed adjacency matrices $\in \mathbb{R}^{n\times n}$
$\downarrow\omega$	$\downarrow\omega$		$\downarrow\omega$		$\uparrow$	
$\omega(A_0)$	$\omega(A_1)$	$\cdots$	$\omega(A_T)$	$\underset{W}{\rightarrow}$	$\widehat{\omega(A_{T+1})}$	Features vectors $\in \mathbb{R}^d$

Table 1: General scheme of our method for prediction in dynamic graph sequences through a feature map ω.

rank of the true adjacency matrix A_{T+1}, and (c) the sparsity of the true VAR model matrix W_0.

Namely, we prove oracle inequalities for the mixed prediction-estimation error which is given, for any $A \in \mathbb{R}^{n\times n}$ and $W \in \mathbb{R}^{d\times d}$, by

$$\mathcal{E}(A,W)^2 \doteq \frac{1}{d}\|(W-W_0)^\top \omega(A_T) - \omega(A - A_{T+1})\|_2^2 + \frac{1}{T}\|\mathbf{X}_{T-1}(W-W_0)\|_F^2.$$

We point out that an upper-bound on $\mathcal{E}$ implies upper-bounds on each of its two components. It entails in particular an upper-bound on the feature estimation error $\|\mathbf{X}_{T-1}(\widehat{W}-W_0)\|_F$ that makes $\|(\widehat{W}-W_0)^\top\omega(A_T)\|_2$ smaller and consequently controls the prediction error over the graph edges through $\|\omega(\widehat{A}-A_{T+1})\|_2$.

We obtain upper bounds that are reminiscent of the bounds obtained for the Lasso by Bickel et al. (2009) for instance, and that are of the following order:

$$\frac{\log d}{T}\|W_0\|_0 + \frac{\log n}{d}\|A_{T+1}\|_0 + \frac{\log n}{d}\operatorname{rank} A_{T+1}.$$

This upper bound, formalized in Theorem 3, exhibits the dependence of the accuracy of estimation and prediction on the number of features d, the number of edges n and the number T of observed graphs in the sequence. It indicates, in particular, that an optimal choice for the number d of features is of order $T\log n$. The positive constants C_1, C_2, C_3 are proportional to the noise level σ. The interplay between the rank and the sparsity constraints on A_{T+1} are reflected in the observation that the values of C_2 and C_3 can be changed as long as their sum remains constant. The precise formulation of these results is given in the next section.

3. Oracle Inequalities

This section contains the main theoretical results ot the paper. Complete proofs and technical details are provided in the Appendix section at the end of the paper.

3.1 A General Oracle Inequality

We recall from subsection 2.2 that the noise sequence in the VAR model is denoted by $\{N_t\}_{t\geq 0}$. We now introduce the noise processes as

$$M = -\frac{1}{d}\sum_{j=1}^{d}(N_{T+1})_j\Omega_j \quad \text{and} \quad \Xi = \frac{1}{T}\sum_{t=1}^{T}\omega(A_{t-1})N_t^\top + \frac{1}{d}\omega(A_T)N_{T+1}^\top,$$

which are, respectively, $n \times n$ and $d \times d$ random matrices. The source of randomness comes from the noise sequence $\{N_t\}_{t \geq 0}$.

Now, if these noise processes are controlled, we can prove oracle inequalities for procedure (4). The first result is an oracle inequality of slow type, that holds in full generality.

Theorem 1 *Under Assumption 1, let $(\hat{A}, \hat{W})$ be given by (4) and suppose that*

$$\tau \geq 2\alpha \|M\|_{\mathrm{op}}, \quad \gamma \geq 2(1-\alpha)\|M\|_{\infty} \quad \text{and} \quad \kappa \geq 2\|\Xi\|_{\infty} \tag{5}$$

for some $\alpha \in (0,1)$. Then, we have

$$\mathcal{E}(\widehat{A}, \widehat{W})^2 \leq \inf_{(A,W) \in \mathcal{A} \times \mathcal{W}} \Big\{ \mathcal{E}(A,W)^2 + 2\tau \|A\|_* + 2\gamma \|A\|_1 + 2\kappa \|W\|_1 \Big\}.$$

3.2 Restricted Eigenvalue Condition and Fast Oracle Inequalities

For the proof of oracle inequalities, the *restricted eigenvalue* (RE) condition introduced by Bickel et al. (2009) and Koltchinskii (2009a,b) is of importance. As explained by van de Geer and Bühlmann (2009), this condition is acknowledged to be one of the weakest to derive fast rates for the Lasso. Matrix version of these assumptions are introduced by Koltchinskii et al. (2011). Below is a version of the RE assumption that fits in our context. First, we need to introduce the two restriction cones.

The first cone is related to the $\|W\|_1$ term used in procedure (4). If $W \in \mathbb{R}^{d \times d}$, we denote by $\Theta_W = \mathrm{sign}(W) \in \{0, \pm 1\}^{d \times d}$ the signed sparsity pattern of W and by $\Theta_W^{\perp} \in \{0,1\}^{d \times d}$ the complementary sparsity pattern. For a fixed matrix $W \in \mathbb{R}^{d \times d}$ and $c > 0$, we introduce the cone

$$\mathcal{C}_1(W, c) \doteq \Big\{ W' \in \mathcal{W} : \|\Theta_W^{\perp} \circ W'\|_1 \leq c \|\Theta_W \circ W'\|_1 \Big\}.$$

This cone contains the matrices W' that have their largest entries in the sparsity pattern of W.

The second cone is related to the mixture of the terms $\|A\|_*$ and $\|A\|_1$ in procedure (4). For a fixed $A \in \mathbb{R}^{n \times n}$ and $c, \beta > 0$, we introduce

$$\mathcal{C}_2(A, c, \beta) \doteq \Big\{ A' \in \mathcal{A} : \|\mathcal{P}_A^{\perp}(A')\|_* + \beta \|\Theta_A^{\perp} \circ A'\|_1 \leq c \Big(\|\mathcal{P}_A(A')\|_* + \beta \|\Theta_A \circ A'\|_1 \Big) \Big\}.$$

This cone consist of the matrices A' with large entries close to that of A and that are "almost aligned" with the row and column spaces of A. The parameter β quantifies the interplay between these two notions.

Assumption 2 (Restricted Eigenvalue (RE) condition) *For $W \in \mathcal{W}$ and $c > 0$, we have*

$$\mu_1(W, c) = \inf \Big\{ \mu > 0 : \|\Theta_W \circ W'\|_F \leq \frac{\mu}{\sqrt{T}} \|\mathbf{X}_{T-1} W'\|_F, \ \ \forall W' \in \mathcal{C}_1(W, c) \Big\} < +\infty \ .$$

For $A \in \mathcal{A}$ and $c, \beta > 0$, we introduce

$$\mu_2(A, W, c, \beta) = \inf \Big\{ \mu > 0 : \ \|\mathcal{P}_A(A')\|_F \vee \|\Theta_A \circ A'\|_F \leq \frac{\mu}{\sqrt{d}} \|W'^{\top} \omega(A_T) - \omega(A')\|_2 \\ \forall W' \in \mathcal{C}_1(W, c), \forall A' \in \mathcal{C}_2(A, c, \beta) \Big\} < +\infty \ .$$

Under this assumption, we can obtain refined oracle inequalities as shown in the next theorem.

Theorem 2 *Under Assumption 1 and Assumption 2, let* $(\hat{A}, \hat{W})$ *be given by* (4) *and suppose that*

$$\tau \geq 3\alpha\|M\|_{\mathrm{op}}, \quad \gamma \geq 3(1-\alpha)\|M\|_{\infty} \quad \text{and} \quad \kappa \geq 3\|\Xi\|_{\infty} \tag{6}$$

for some $\alpha \in (0,1)$*. Then, we have*

$$\mathcal{E}(\widehat{A}, \widehat{W})^2 \leq \inf_{(A,W)\in\mathcal{A}\times\mathcal{W}} \Big\{ \mathcal{E}(A,W)^2 + \frac{25}{18}\mu_2(A,W)^2\big(\tau^2 \operatorname{rank} A + \gamma^2\|A\|_0\big) + \frac{25}{36}\kappa^2\mu_1(W)^2\|W\|_0 \Big\} ,$$

where $\mu_1(W) = \mu_1(W,5)$ *and* $\mu_2(A,W) = \mu_2(A,W,5,\gamma/\tau)$ *(see Assumption 2).*

The proofs of Theorems 1 and 2 use tools introduced by Koltchinskii et al. (2011) and Bickel et al. (2009). Note that the residual term from this oracle inequality combines the sparsity of A and W via the terms $\operatorname{rank} A$, $\|A\|_0$ and $\|W\|_0$. It says that our mixed penalization procedure provides an optimal trade-off between fitting the data and complexity, measured by both sparsity and low-rank. To our knowledge, this is the first result of this nature to be found in literature.

3.3 Probabilistic Versions

We introduce the following natural hypothesis on the noise process.

Assumption 3 *We assume that* $\{N_t\}_{t\geq 0}$ *satisfies* $\mathbb{E}[N_t|\mathcal{F}_{t-1}] = 0$ *for any* $t \geq 1$ *and that there is* $\sigma > 0$ *such that for any* $\lambda \in \mathbb{R}$ *and* $j = 1, \ldots, d$ *and* $t \geq 0$*:*

$$\mathbb{E}[e^{\lambda (N_t)_j}|\mathcal{F}_{t-1}] \leq e^{\sigma^2\lambda^2/2}.$$

Moreover, we assume that for each $t \geq 0$*, the coordinates* $(N_t)_1, \ldots, (N_t)_d$ *are independent.*

The latter statement assumes that the noise is driven by time-series dynamics (a martingale increment), where the coordinates are independent (meaning that features are independently corrupted by noise), with a sub-gaussian tail and variance uniformly bounded by a constant σ^2. In particular, no independence assumption between the N_t is required here.

In the next result (Theorem 3), we obtain convergence rates for the procedure (4) by combining Theorem 2 with controls on the noise processes. We introduce the following quantities:

$$v^2_{\Omega,\mathrm{op}} = \Big\|\frac{1}{d}\sum_{j=1}^{d}\Omega_j^\top\Omega_j\Big\|_{\mathrm{op}} \vee \Big\|\frac{1}{d}\sum_{j=1}^{d}\Omega_j\Omega_j^\top\Big\|_{\mathrm{op}}, \quad v^2_{\Omega,\infty} = \Big\|\frac{1}{d}\sum_{j=1}^{d}\Omega_j\circ\Omega_j\Big\|_{\infty}, \tag{7}$$

$$\sigma^2_{\omega} = \max_{j=1,\ldots,d}\sigma^2_{\omega,j}, \quad \text{where} \quad \sigma^2_{\omega,j} = \Big(\frac{1}{T}\sum_{t=1}^{T}\omega_j(A_{t-1})^2 + \omega_j(A_T)^2\Big),$$

which are the (observable) variance terms that naturally appear in the upper bounds of the noise processes. We also introduce :

$$\ell_T = 2\max_{j=1,\ldots,d}\log\log\left(\sigma^2_{\omega,j} \vee \frac{1}{\sigma^2_{\omega,j}} \vee e\right), \tag{8}$$

which is a small (observable) technical term that comes out of our analysis of the noise process Ξ. This term is a small price to pay for the fact that no independence assumption is required on the noise sequence $\{N_t\}_{t\geq 0}$, but only a martingale increment structure with sub-gaussian tails.

We consider the following calibration of smoothing parameters as a function of noise process parameters:

$$\begin{aligned}
\tau &= 3\sqrt{2}\alpha\sigma v_{\Omega,\mathrm{op}}\sqrt{\frac{x+\log(2n)}{d}}\ , \\
\gamma &= 3(1-\alpha)\sigma v_{\Omega,\infty}\sqrt{\frac{2(x+2\log n)}{d}}\ , \\
\kappa &= 6\sigma\sigma_\omega\left\{\sqrt{\frac{2e(x+2\log d+\ell_T)}{T}}+\frac{\sqrt{2e(x+2\log d+\ell_T)}}{d}\right\}\ .
\end{aligned}$$

In the next Theorem 3 and Corollary 4, we fix the smoothing parameters to the latter values. These two results convey the main message of the paper as it was announced in Section 2.4.

Theorem 3 *Under Assumption 1, Assumption 2 and Assumption 3, consider the procedure $(\hat{A},\hat{W})$ given by (4) applied with the calibration of smoothing parameters shown above for some $\alpha\in(0,1)$ and a fixed confidence level $x>0$. Then, we have, with probability larger than $1-17e^{-x}$:*

$$\begin{aligned}
\mathcal{E}(\widehat{A},\widehat{W})^2 \leq \inf_{(A,W)\in\mathcal{A}\times\mathcal{W}}\Big\{&\mathcal{E}(A,W)^2 + C_1\|W\|_0(x+2\log d+\ell_T)\Big(\frac{1}{T}+\frac{1}{d^2}\Big) \\
&+ C_2\|A\|_0\frac{x+2\log n}{d} + C_3\,\mathrm{rank}\,A\frac{x+\log(2n)}{d}\Big\}
\end{aligned}$$

where

$$C_1 = 100e\mu_1(W)^2\sigma^2\sigma_\omega^2,\quad C_2 = 50\mu_2(A,W)^2(1-\alpha)^2\sigma^2 v_{\Omega,\infty}^2,\quad C_3 = 50\mu_2(A,W)^2\alpha^2\sigma^2 v_{\Omega,\mathrm{op}}^2,$$

and RE constants $\mu_1(W)$ and $\mu_2(A,W)$ are taken as in Theorem 2.

The proof of Theorem 3 follows directly from Theorem 2 together with noise control assumptions. In the next result, we propose more explicit upper bounds for both the individual estimation of W_0 and the prediction of A_{T+1}.

Corollary 4 *Under the same assumptions as in Theorem 3 and the same choice of smoothing parameters, for any $x>0$ the following inequalities hold with probability larger than $1-17e^{-x}$:*

- *Feature prediction error:*

$$\begin{aligned}
\frac{1}{T}\|\mathbf{X}_T(\hat{W}-W_0)\|_F^2 \leq{}& \frac{25}{36}\kappa^2\mu_1(W_0)^2\|W_0\|_0 \\
&+ \inf_{A\in\mathcal{A}}\Big\{\frac{1}{d}\|\omega(A)-\omega(A_{T+1})\|_2^2 + \frac{25}{18}\mu_2(A,W_0)^2\big(\tau^2\,\mathrm{rank}\,A+\gamma^2\|A\|_0\big)\Big\} \quad (9)
\end{aligned}$$

- *VAR parameter estimation error:*

$$\|\hat{W} - W_0\|_1 \leq 5\kappa\mu_1(W_0)^2\|W_0\|_0 \\ +6\sqrt{\|W_0\|_0}\mu_1(W_0)\inf_{A\in\mathcal{A}}\sqrt{\frac{1}{d}\|\omega(A)-\omega(A_{T+1})\|_2^2+\frac{25}{18}\mu_2(A,W_0)^2\big(\tau^2\operatorname{rank}A+\gamma^2\|A\|_0\big)} \tag{10}$$

- *Link prediction error:*

$$\|\hat{A} - A_{T+1}\|_* \leq 5\kappa\mu_1(W_0)^2\|W_0\|_0 + \mu_2(A_{T+1},W_0)(6\sqrt{\operatorname{rank}A_{T+1}}+5\frac{\gamma}{\tau}\sqrt{\|A_{T+1}\|_0}) \\ \times\inf_{A\in\mathcal{A}}\sqrt{\frac{1}{d}\|\omega(A)-\omega(A_{T+1})\|_2^2+\frac{25}{18}\mu_2(A,W_0)^2\big(\tau^2\operatorname{rank}A+\gamma^2\|A\|_0\big)}\ . \tag{11}$$

4. Algorithms and Data Modeling

In this section, we explore how the proposed strategy of regularized optimization for simultaneously estimating the feature dynamics and predicting the forthcoming links can be implemented in practice.

4.1 Incremental Proximal-Gradient Algorithm for Minimizing $\mathcal{L}$

The objective to be minimized in our problem can be written as:

$$\mathcal{L} = \ell + \mathcal{R}\ ,$$

where we have set the loss function ℓ:

$$\ell : (A, W) \mapsto \frac{1}{T}\|\mathbf{X}_T - \mathbf{X}_{T-1}W\|_F^2 + \frac{1}{d}\|\omega(A) - W^\top\omega(A_T)\|_2^2\ ,$$

and the regularizer $\mathcal{R}$:

$$\mathcal{R} : (A, W) \mapsto \kappa\ \|W\|_1 + \tau\|A\|_* + \gamma\|A\|_1\ .$$

We propose to develop an algorithm for solving this optimization problem based on proximal gradient methods. Proximal algorithms (Beck and Teboulle, 2009; Combettes and Pesquet, 2011) have been designed for solving convex optimization problems where functionals have the following structure : $\mathcal{L} = \ell + \mathcal{R}$, where ℓ is convex, differentiable with a Lipschitz gradient and $\mathcal{R}$ is convex and not differentiable. This is exactly our case. In the classical setup, it is assumed that $\mathcal{R}$ has an explicit (or fast to compute) proximal operator, defined by:

$$\operatorname{prox}_{\mathcal{R}}(X) = \arg\min_Y\left\{\frac{1}{2}\|X-Y\|_F^2+\mathcal{R}(Y)\right\}\ .$$

It has been proved by Beck and Teboulle (2009) that the sequence

$$X_{k+1} = \operatorname{prox}_{\theta\mathcal{R}}(X_k - \theta\nabla\ell(X_k))$$

converges after $O(1/\epsilon)$ steps to a ball of radius ϵ of the minimizer of $\mathcal{L}$. The step size θ is usually taken of the order of magnitude of the inverse of the Lipschitz constant L of $\nabla\ell$. An accelerated algorithm (FISTA) that reaches the optimal convergence rate $O(1/\sqrt{\epsilon})$ in the sense of Nesterov (2005) can be written using an auxiliary sequence, are described by Beck and Teboulle (2009) and Tseng (2008). The intuition behind the design of these algorithms relies on the linear expansion of ℓ around the point X_k and the quadratic term $\frac{L}{2}\|X - X_k\|_F^2$ that controls the closeness of the next step point X_{k+1} from X_k. Namely, we can write

$$\begin{aligned}\mathcal{L}(X) &\approx \ell(X_k) + \nabla\ell(X_k)^\top(X - X_k) + \mathcal{R}(X) + \frac{L}{2}\|X - X_k\|_F^2 \\ &= L\left\{\frac{1}{2}\left\|(X - X_k) + \frac{1}{L}\nabla\ell(X_k)\right\|_F^2 - \frac{1}{2L^2}\|\nabla\ell(X_k)\|_F^2 + \frac{1}{L}\ell(X_k) + \frac{1}{L}\mathcal{R}(X)\right\} \\ &= L\left\{\frac{1}{2}\left\|X - (X_k - \frac{1}{L}\nabla\ell(X_k))\right\|_F^2 + \frac{1}{L}\mathcal{R}(X)\right\} + \text{constant}.\end{aligned}$$

It follows that the point $X_{k+1} = \mathrm{prox}_{\frac{1}{L}\mathcal{R}}(X_k - \frac{1}{L}\nabla\ell(X_k))$ is a fair approximation of the minimizer of $\mathcal{L}$ around X_k. The detailed analysis and extensions can be found in the paper by Tseng (2008).

In our case, the presence of the sum of two simple regularizers (ℓ_1 and trace norm) applied to the same object A makes the situation slightly more complicated, since the proximal operator of this sum is non-explicit. We propose to use an incremental algorithm to address this complication. Indeed, the proximal operators of each term are available. First, it is known that the proximal operator of the trace norm is given by the spectral shrinkage operator: if $X = U\,\mathrm{diag}(\sigma_1, \cdots, \sigma_n)V^\top$ is the singular value decomposition of X, we have

$$\mathrm{prox}_{\tau\|\cdot\|_*}(X) = U\,\mathrm{diag}((\sigma_i - \tau)_+)V^\top.$$

For the ℓ_1-norm, the proximal operator is the entrywise soft-thresholding defined by

$$\mathrm{prox}_{\gamma\|\cdot\|_1}(X) = \mathrm{sgn}(X) \circ (|X| - \gamma)_+,$$

where we recall that $\circ$ denotes the entry-wise product. The algorithm converges under very mild conditions when the step size θ is smaller than $2/L$, where L is the operator norm of the joint quadratic loss.

The algorithm is described below (see Algorithm 1). It is inspired from the method proposed by Bertsekas (2011) Section 2 and conducts to the minimization our objective function. The order in which proximal mappings are performed is chosen in order to compute the SVD on a sparse matrix Z, for computational efficiency. If a sparse output is desired, an extra soft-thresholding step can be performed at the end. Note that this algorithm is preferable to the method previously introduced by Richard et al. (2010) as it directly minimizes $\mathcal{L}$ jointly in (A, W) rather than alternately minimizing in W and A.

4.2 A Generative Model for Graphs with Linearly Autoregressive Features

In order to prepare the setup for empirical evaluation of the algorithm, we now explain how synthetic data can be generated from the statistical model with linear autoregressive features. Let $V_0 \in \mathbb{R}^{n\times r}$ be a sparse matrix, $V_0^\dagger$ its pseudo-inverse such that $V_0^\dagger V_0 = V_0^\top V_0^{\top\dagger} = I_r$.

Algorithm 1 Incremental Proximal-Gradient to Minimize $\mathcal{L}$

Initialize A, Z_1, Z_2, W
repeat
  Compute $(G_A, G_W) = \nabla_{A,W}\ell(A, W)$.
  Compute $Z = \text{prox}_{\theta\gamma\|\cdot\|_1}(A - \theta G_A)$
  Compute $A = \text{prox}_{\theta\tau\|\cdot\|_*}(Z)$
  Set $W = \text{prox}_{\theta\kappa\|\cdot\|_1}(W - \theta G_W)$
until convergence
return (A, W) minimizing $\mathcal{L}$

Fix two sparse matrices $K_0 \in \mathbb{R}^{r\times r}$ and $U_0 \in \mathbb{R}^{n\times r}$. Now define the sequence of matrices $\{A_t\}_{t\geq 0}$ for $t = 1, 2, \cdots$ by

$$U_t = U_{t-1}K_0 + N_t$$

and

$$A_t = U_t V_0^\top$$

for a sequence of i.i.d sparse noise matrices $\{N_t\}_{t\geq 0}$, which means that for any pair of indices (i, j), we have $(N_t)_{i,j} = 0$ with a high probability. We consider the vectorization operator $A \mapsto \text{vec}(A)$ that stacks the columns of A into a single column, and define the linear feature map

$$\omega(A) \doteq \text{vec}(A\Psi),$$

where we set for short $\Psi = (V_0^\top)^\dagger$, so that $V_0^\top \Psi = I_r$. Let us notice that

1. The sequence $\{\omega(A_t)\}_t = \{\text{vec}(U_t)\}_t$ follows the linear autoregressive relation

   $$\omega(A_t) = (K_0^\top \otimes I_n)\omega(A_{t-1}) + \text{vec}(N_t),$$

   where $\text{vec}(N_t)$ is a zero-mean noise process and $\otimes$ is the Kronecker product.

2. For any time index t, the matrix A_t is close to $U_t V_0^\top$ that has rank at most r

3. The matrices A_t and U_t are both sparse by construction.

4. The dimension of the feature space is $d = nr \ll n^2$, so $W_0 = K_0^\top \otimes I_n \in \mathbb{R}^{nr\times nr}$. The feature map can be written in standard form, see Equation (1), after vectorization by using the design matrices

   $$\Omega_{(l-1)n+i} = e_i(\Psi^\top)_{l,\cdot}$$

   for $1 \leq l \leq r, 1 \leq i \leq n$, where the $n \times n$ design matrix $e_i(\Psi^\top)_{l,\cdot}$ contains a copy of the l-th column of Ψ at its i-th row and zeros elsewhere. The standard form of the feature map is then given by the vector

   $$\omega(A) = [\langle A, \Omega_{(l-1)n+i}\rangle : 1 \leq l \leq r, 1 \leq i \leq n]^\top.$$

   As a consequence, we can compute the variance terms $v_{\Omega,\infty}$ and $v_{\Omega,\text{op}}$ from Equation (7) as functions of Ψ. By using

   $$e_i(\Psi^\top)_{l,\cdot}\Psi_{\cdot,l}e_i^\top = \|\Psi_{\cdot,l}\|_2^2 e_i e_i^\top \quad \text{and} \quad \Psi_{\cdot,l}e_i^\top e_i(\Psi^\top)_{l,\cdot} = \Psi_{\cdot,l}(\Psi^\top)_{l,\cdot},$$

we get respectively by summation over indices i and l,

$$\sum_{l=1}^{r}\sum_{i=1}^{n} e_i \Psi_{l,\cdot}^\top \Psi_{l,\cdot} e_i^\top = \Big(\sum_{l=1}^{r} \|\Psi_{l,\cdot}^\top\|_2^2\Big)\Big(\sum_{i=1}^{n} e_i e_i^\top\Big) = \|\Psi\|_F^2 I_n$$

and Equation (7) gives us the values of the variance terms

$$v_{\Omega,\mathrm{op}} = \frac{1}{nr}\Big(\Big\|\sum_{l=1}^{r} \Psi_{\cdot,l}(\Psi_{\cdot,l})^\top\Big\|_{\mathrm{op}} \vee \|\Psi\|_F^2\Big) \quad \text{and} \quad v_{\Omega,\infty} = \frac{1}{n}\|\Psi\|_{\infty,2}^2,$$

where the $(\infty,2)$-norm is defined by the maximum ℓ_2-norm of the columns, $\|X\|_{\infty,2} \doteq \max_j \|X_{\cdot,j}\|_2$.

4.3 Beyond the First-Order Autoregressive Model

The theory developed in Sections 2 and 4 considers the VAR model of order $p = 1$ for the sake of simplicity. However, our approach is flexible, since we may use any other time-series modelling. To give a simple illustration of this fact, we consider below an extension to the second-order VAR model. Indeed, we don't want the VAR order p to be too large, since the number of parameters scales as $d^2 \times p$ (forgetting about sparsity assumptions). In our experiments (see Section 5 below), we consider and compare both first order and second order VAR models.

Let us define the $(T-1) \times d$ time-series matrices

$$\mathbf{X}_T = \big[\omega(A_2), \ldots, \omega(A_T)\big]^\top, \quad \mathbf{X}_{T-1} = \big[\omega(A_1), \ldots, \omega(A_{T-1})\big]^\top,$$
$$\mathbf{X}_{T-2} = \big[\omega(A_0), \ldots, \omega(A_{T-2})\big]^\top.$$

We consider the following order 2 extension of the features VAR model:

$$\mathbf{X}_T = \mathbf{X}_{T-1} W_1 + \mathbf{X}_{T-2} W_2 + \mathbf{N}_T,$$

where $\mathbf{N}_T = [N_2, \ldots, N_T]^\top$ denotes the centered noise vector, and W_1, W_2 are VAR model parameters. In this case the Lasso objective is

$$J_1(W_1, W_2) = \frac{1}{T}\Big\|\mathbf{X}_T - \begin{bmatrix}\mathbf{X}_{T-1} & \mathbf{X}_{T-2}\end{bmatrix}\begin{bmatrix}W_1 \\ W_2\end{bmatrix}\Big\|_F^2 + \kappa \left\|\begin{bmatrix}W_1 \\ W_2\end{bmatrix}\right\|_1$$

and the Lasso estimator is defined by

$$(\widehat{W_1}, \widehat{W_2}) = \operatorname*{arg\,min}_{W_1, W_2} J_1(W_1, W_2).$$

In the same spirit as in Section 2.3 we define the objective

$$\mathcal{L}(A, W_1, W_2) = J_1(W_1, W_2) + \frac{1}{d}\|\omega(A)^\top - \omega(A_T)^\top W_1 - \omega(A_{T-1})^\top W_2\|_2^2 + \tau\|A\|_* + \gamma\|A\|_1.$$

The gradients of the quadratic loss are given by

$$\frac{1}{2}\begin{bmatrix}\nabla_{W_1}\ell\\ \nabla_{W_2}\ell\end{bmatrix} = \frac{1}{T}\begin{bmatrix}\mathbf{X}_{T-1}^\top\\ \mathbf{X}_{T-2}^\top\end{bmatrix}\left\{\begin{bmatrix}\mathbf{X}_{T-1} & \mathbf{X}_{T-2}\end{bmatrix}\begin{bmatrix}W_1\\ W_2\end{bmatrix} - \mathbf{X}_T\right\} + \frac{1}{d}\begin{bmatrix}\omega(A_T)\\ \omega(A_{T-1})\end{bmatrix}\left\{\begin{bmatrix}\omega(A_T)^\top & \omega(A_{T-1})^\top\end{bmatrix}\begin{bmatrix}W_1\\ W_2\end{bmatrix} - \omega(A)^\top\right\},$$

and

$$\frac{1}{2}\nabla_A \ell^\top = \frac{1}{d}\sum_{j=1}^{d}\left\{\omega(A)_j - (\omega(A_T)^\top W_1 + \omega(A_{T-1})^\top W_2)_j\right\}\Omega_j,$$

where $\Omega_j \in \mathbb{R}^{n\times n}$ is the j-th design matrix. We implemented the second order autoregressive model with three different types of penalties. We used:

1. Ridge Regression using $\kappa\|W_1\|_F^2 + \kappa\|W_2\|_F^2$ as the penalty term
2. the Lasso estimator, that is, the minimizer of J_1
3. the estimator suggested in this work.

5. Empirical Evaluation

In Section 5.1 we assess our algorithms on synthetic data, generated as described in Section 4.2. In Section 5.2 we use our algorithm for the prediction of sales volume for web-marketing data

5.1 Experiments with Synthetic Data

Data generator. In our experiments, the noise matrices M_t are built by soft-thresholding *i.i.d.* noise $\mathcal{N}(0, \sigma^2)$. We took as input $T = 10$ successive graph snapshots on $n = 50$ nodes graphs of rank $r = 5$. We used $d = 10$ linear features, and finally the noise level was set to $\sigma = .5$. Since the matrix V_0 defining the linear map ω is unknown we consider the feature map $\omega(A) = \mathrm{vec}(AV)$ where $\widetilde{A_T} = U\Sigma V^\top$ is the SVD of $\widetilde{A_T}$.
Competitors. The competing methods for our problem, as considered in this paper, are:

- *Nearest Neighbors*, that scores pairs of nodes with the number of common friends between them, which is given by A^2 where A is the cumulative graph adjacency matrix $\widetilde{A_T} = \sum_{t=0}^{T} A_t$;
- *Static sparse and low-rank*, that is the link prediction algorithm suggested by Richard et al. (2012b), which is obtained by minimizing the objective $\|X - \widetilde{A_T}\|_F^2 + \tau\|X\|_* + \gamma\|X\|_1$. It is the closest static version of our method;
- *Autoregressive low-rank* and *Static low-rank*, that are regularized using only by the trace-norm (corresponding to $\gamma = 0$);
- *Katz* scores pairs of nodes i and j by the sum of number of paths of length l connecting i and j, weighted by an exponentially decreasing coefficient β^l: $\sum_{l=1}^{\infty}\beta^l\left(A^l\right)_{i,j}$;

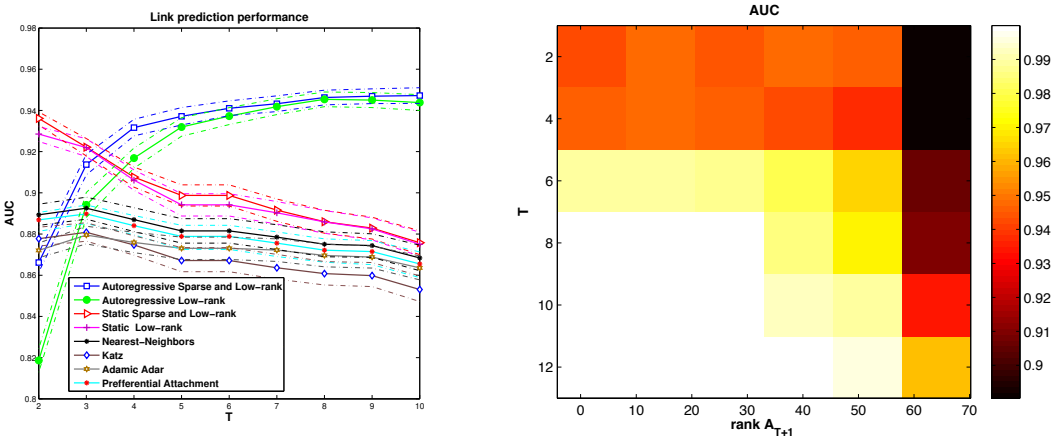


Figure 2: Left: performance of algorithms in terms of Area Under the ROC Curve, average and confidence intervals over 50 runs. Right: Phase transition diagram.

- *Adamic Adar* is the score $\sum_{\nu \in N(i) \cap N(j)} 1/\log(d_\nu)$, where d_ν is the degree of the node ν which is a common neighbor of i and j;
- *Preferential attachment* only takes popularity into account and scores an edge ij by the product of their degrees $d_i d_j$. See the papers by Liben-Nowell and Kleinberg (2007) and Lü and Zhou (2011) for details on Katz, Adamic-Adar and Preferential Attachment.

We also point out that other methods could possibly be adapted for the problem of link prediction as stated in the present paper. We mainly refer to the works by Liben-Nowell and Kleinberg (2007), Lü and Zhou (2011), Sarkar et al. (2012), Huang and Lin (2009), Nardi and Rinaldo (2011) and Davis et al. (2012). However, they were introduced either in a different setup, such as the one where multiple observations of a given edge occur, as described by Liben-Nowell and Kleinberg (2007) and Lü and Zhou (2011), or in the feature prediction problem of Nardi and Rinaldo (2011) and Davis et al. (2012), or they would involve tuning complex subroutines, such as the ones of Huang and Lin (2009), leading us far beyond the scope of the present work.

Performance assessment for validation and test. We compare our methods to standard baselines in link prediction by comparing predictions $\hat{A}$ to the adjacency matrix $A_{T+1} = A$, which is binary, at step $T+1$. Since the score matrix $\hat{A}$ outputs scalar values, we use a threshold parameter t to build a link predictor $\mathbf{I}\{\hat{A}_{i,j} > t\}$ on the edge (i,j). The quality of our estimation is then measured by considering all possible values of the threshold parameter t which leads to the ROC curve as the plot of the proportion of hits (pairs (i,j) such that $A_{ij} \cdot \mathbf{I}\{\hat{A}_{i,j} > t\} = 1$) versus the proportion of false detection (pairs (i,j) such that $A_{ij} \cdot \mathbf{I}\{\hat{A}_{i,j} > t\} = 0$). Our criterion is the AUC for this particular definition of the ROC curve. In this approach of assessment, the size of the coefficients of $\hat{A}$ accounts for the strength of the prediction. We report empirical results averaged over 50 runs with confidence intervals in Figure 2. The parameters τ and γ are chosen by a 10-fold cross validation for

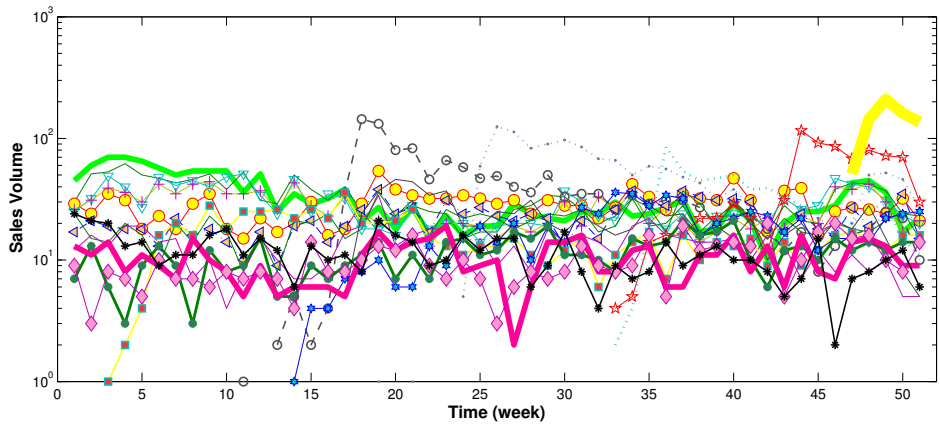


Figure 3: Sales volumes of 20 top-sold items weekly sales over the year.

each of the methods separately. The validation set is the upwards sliding time window when learning from the past. The right-hand side of Figure 2 is a phase transition diagram showing the impact of both rank and time on the accuracy of estimation of the adjacency matrix. The results are clearly better as we gain historical depth and the lower the rank of the adjacency matrix.

Comparison with the baselines. This experiment shows the benefits of using a temporal approach when one can handle the feature extraction task. The left-hand plot shows that if few snapshots are available ($T \leq 4$ in these experiments), then static approaches are to be preferred, whereas feature autoregressive approaches outperform them as soon as a *sufficient number* T of graph snapshots are available (see the Phase transition diagram from Figure 2). The decreasing performance of static algorithms can be explained by the fact that they use as input a mixture of different graphs observed at different time steps, whereas they require a single simple graph as input.

5.2 Experiments with Real Data: Predicting Sales Volumes

Motivations. Predicting the popularity of products is of major interest for marketers and product managers as it allows to anticipate or even create trends that diffuse in networks. A useful observation when dealing with sales volumes is that when modeling purchases by edges in the bipartite graph of users and products, the sales volumes can be seen as the degrees of the product nodes. We use two VAR models of order 1 and 2, as described in Section 4.3, in order to show the flexibility of our approach. We consider the linear feature map $\omega(A) = A^\top \mathbf{1}$ that computes the columns degree vector. The dimension of the feature space d equals the number of columns of the matrix in this case. If the input matrix A_t is the *users* $\times$ *products* matrix of sales at time period t then the degree of each product equals the sales volume of the product during that period, and it can be used as a fair popularity indicator for the product. It is in addition common to consider a regular evolution of such features, see the paper by Rogers (1962). Note that the suggested approach is valid for a similar activity indicator in any other network, such as users activity on a social network

	AR(1)			AR(2)		
Error	Ridge	Lasso	Our	Ridge	Lasso	Our
$T = 10$	0.9524	1.1344	0.9730	1.0037	1.0110	**0.9416**
$T = 15$	0.6394	0.5389	0.5336	0.6010	**0.5304**	0.5401
$T = 20$	0.3419	0.3111	0.4878	0.3310	**0.2972**	0.3086
$T = 25$	0.3777	0.6238	0.5689	0.3356	0.3286	**0.3279**

Table 2: Relative quadratic error of the prediction of sales volume for three regularized VAR models: one based on ridge regression penalty, one base don LASSO penalty, and one based on our strategy with both sparse and low-rank regularizers.

or protein activity on a protein-protein interaction network. A last remark is that the prior knowledge in the case of e-commerce data suggests that groups of highly related items exist, which makes the adjacency matrix low-rank in addition to be sparse. In fact the adjacency matrix of a fully clustered graph would be block-diagonal, and we expect the matrix of interest to be close to such a matrix.

Protocol and description of data sets. We performed our experiments on the sales volumes time series of the $n = 200$ top sold books over $T = 25$ consecutive weeks excluding the Christmas period in 2009 of 31972 users.[1] The weekly sales of each book corresponds to the degree of the corresponding node. The books catalogue contains several book categories, which motivates the low-rank matrix assumption. In Figure 3 we plot the time series of the top 20 items from the catalogue. From this observation, the stationary assumption seems plausible. More precisely, we observed that the time window allowing accurate predictions (a window in which the data is stationary) is, among the range of values we used in our experiments, equal to 20 weeks.

Comparison with other methods and performance metric. We compare estimators of the degrees based on Ridge and Lasso penalization using the objective J_1 only, see Equation (2), with our procedure based on joint minimization of (3). For choosing the tuning parameters κ, τ, γ we use the data collected from the same market a year before the test set to form the training and validation sets. For testing the quality of our predictor, we used the parameters performing the best predictions over the validation set. As data are abundant we can collect past data easily for this step. On the other hand, as seasonality effects may harm the method if cross-validation is performed on data taken from a different period of the year, this is the best way to proceed for splitting the data onto training validation and test sets. We evaluated the results in terms of relative quadratic error

$$\text{Relative quadratic error} = \frac{\|\omega(\widehat{A}) - \omega(A_{T+1})\|_2}{\|\omega(A_{T+1})\|_2}$$

over the prediction of the sales volumes. The results are reported in Table 2.

Comments on the results. From this experiment we conclude the following. The order of the VAR is an important factor. We provided theoretical results for the VAR of order 1,

1. The data was provided by the company 1000mercis.

but fitting a higher order VAR in practice may result in better performance. This is also a parameter that should ideally be chosen using the past data in a cross-validation process. Moreover, the size of the time window T should be chosen according to the data. A small value of T leads to poor result due to absence of enough signal. As opposite, a too large value of T harms the quality of prediction due to the nonstationary trends in too large windows of time. Note that in our synthetic data experiments only the first effect was observed: the performance is increasing as the time parameter T increases. This is due to the stationarity in synthetically generated data.

5.3 Discussion

Trading convexity for scalability. In the numerical experiments, for better scalability, one can replace the penalty on A by a sparsity inducing penalty on the factors of A. Namely if $A = UV^\top$ is a factorization of A, one can replace the term $\tau\|A\|_* + \gamma\|A\|_1$ by $\lambda\|U\|_1\|V\|_1$. This penalty leads to a non-convex problem, nevertheless it allows better scalability than the convex penalty both in terms of memory requirement and computational complexity, when evaluating the proximal. Another practical advantage of this change of variable is that we need to tune only one real parameter λ instead of two (γ and τ). The maximum rank of $A = UV^\top$ (number of columns of U and V) replaces the low-rank inducing effect of τ.
Generalization of the regression method. In this paper, we consider only an autoregressive process of order 1 and 2. For better prediction accuracy, one could consider more general models, such as vector ARMA models, and use model-selection techniques for the choice of the orders of the model. A general modelling based on state-space model could be developed as well.
Choice of the feature map ω. In this work, we have used the projection onto the vector space of the top-r singular vectors of the cumulative adjacency matrix as the linear map ω, and this choice has shown empirical superiority to other choices. The question of choosing the best measurement / representation to summarize graph information as in compress sensing seems to have both theoretical and application potential. In our work the map ω was applied to a single matrix A_t. One can consider a mapping taking as input several successive matrices A_t, A_{t+1}, A_{t+2}. This idea has been used by Zhang et al. (2011) in order to distinguish the effect of new and returning customers in a marketplace. Moreover, a deeper understanding of the connections of our problem with compressed sensing, for the construction and theoretical validation of the feature map, is an important point that needs several developments. An extension to nonlinear graph features such as the distribution of triangles or other nonlinear patterns of interest is also to be considered.

6. Conclusion

In this work, we studied the link prediction problem under structural hypotheses on the graph generation process (sparse low-rank adjacency and autoregressive features). Our work establishes a connection between the link prediction problem and compressed sensing through the use of common tools in the model and in the theoretical analysis. Empirical experiments show the benefit of adopting such a point of view. In fact, compared to the existing heuristics, this approach offers a principled search method in the hypothesis space through the regularization and convex optimization formulation. The flexibility of our ap-

proach and its connections with several active areas of research makes it very attractive and reveals several interesting directions of investigation for future work.

Appendix A. Proofs of the Main Results

From now on, we use the notation $\|(A,a)\|_F^2 = \|A\|_F^2 + \|a\|_2^2$ and $\langle (A,a),(B,b)\rangle = \langle A,B\rangle + \langle a,b\rangle$ for any $A,B \in \mathbb{R}^{T\times d}$ and $a,b \in \mathbb{R}^d$.

Let us introduce the linear mapping $\Phi : \mathbb{R}^{n\times n} \times \mathbb{R}^{d\times d} \to \mathbb{R}^{T\times d} \times \mathbb{R}^d$ given by

$$\Phi(A,W) = \Big(\frac{1}{\sqrt{T}}\mathbf{X}_{T-1}W, \frac{1}{\sqrt{d}}(\omega(A) - W^\top \omega(A_T))\Big).$$

Using this mapping, the objective (3) can be written in the following reduced way:

$$\mathcal{L}(A,W) = \Big\|\Big(\frac{1}{\sqrt{T}}\mathbf{X}_T, 0\Big) - \Phi(A,W)\Big\|_F^2 + \gamma\|A\|_1 + \tau\|A\|_* + \kappa\|W\|_1.$$

Recalling that the error writes, for any A and W:

$$\mathcal{E}(A,W)^2 = \frac{1}{d}\|(W-W_0)^\top\omega(A_T) - \omega(A - A_{T+1})\|_2^2 + \frac{1}{T}\|\mathbf{X}_{T-1}(W-W_0)\|_F^2,$$

we have

$$\mathcal{E}(A,W)^2 = \big\|\Phi(A - A_{T+1}, W - W_0)\|_F^2.$$

Let us introduce also the empirical risk

$$R_n(A,W) = \Big\|\Big(\frac{1}{\sqrt{T}}\mathbf{X}_T, 0\Big) - \Phi(A,W)\Big\|_F^2.$$

The proofs of Theorem 1 and 2 are based on tools developed by Koltchinskii et al. (2011) and Bickel et al. (2009). However, the context considered here is very different from the setting considered in these papers, so our proofs require a different scheme.

A.1 Proof of Theorem 1

First, note that

$$\begin{aligned} &R_n(\hat{A},\hat{W}) - R_n(A,W) \\ &\qquad = \|\Phi(\hat{A},\hat{W})\|_F^2 - \|\Phi(A,W)\|_F^2 - 2\langle(\frac{1}{\sqrt{T}}\mathbf{X}_T, 0), \Phi(\hat{A}-A, \hat{W}-W)\rangle. \end{aligned}$$

Since

$$\begin{aligned} &\|\Phi(\hat{A},\hat{W})\|_F^2 - \|\Phi(A,W)\|_F^2 \\ &\qquad = \mathcal{E}(\hat{A},\hat{W})^2 - \mathcal{E}(A,W)^2 + 2\langle\Phi(\hat{A}-A,\hat{W}-W), \Phi(A_{T+1},W_0)\rangle, \end{aligned}$$

we have

$$\begin{aligned} &R_n(\hat{A},\hat{W}) - R_n(A,W) \\ &\qquad = \mathcal{E}(\hat{A},\hat{W})^2 - \mathcal{E}(A,W)^2 + 2\langle\Phi(\hat{A}-A,\hat{W}-W), \Phi(A_{T+1},W_0) - (\frac{1}{\sqrt{T}}\mathbf{X}_T, 0)\rangle \\ &\qquad = \mathcal{E}(\hat{A},\hat{W})^2 - \mathcal{E}(A,W)^2 + 2\langle\Phi(\hat{A}-A,\hat{W}-W), (-\frac{1}{\sqrt{T}}\mathbf{N}_T, \frac{1}{\sqrt{d}}N_{T+1})\rangle. \end{aligned}$$

The next Lemma will come in handy several times in the proofs.

Lemma 5 *For any $A \in \mathbb{R}^{n\times n}$ and $W \in \mathbb{R}^{d\times d}$ we have*

$$\langle (\frac{1}{\sqrt{T}}\mathbf{N}_T, -\frac{1}{\sqrt{d}}N_{T+1}), \Phi(A,W)\rangle = \langle (M,\Xi),(A,W)\rangle = \langle W,\Xi\rangle + \langle A, M\rangle.$$

This Lemma follows from a direct computation, and the proof is thus omitted. This Lemma entails, together with (4), that

$$\begin{aligned}\mathcal{E}(\hat{A},\hat{W})^2 \leq \mathcal{E}(A,W)^2 &+ 2\langle \hat{W}-W,\Xi\rangle + 2\langle \hat{A}-A, M\rangle \\ &+ \tau(\|A\|_* - \|\widehat{A}\|_*) + \gamma(\|A\|_1 - \|\widehat{A}\|_1) + \kappa(\|W\|_1 - \|\widehat{W}\|_1).\end{aligned}$$

Now, using Hölder's inequality and the triangle inequality, and introducing $\alpha \in (0,1)$, we obtain

$$\begin{aligned}\mathcal{E}(\hat{A},\hat{W})^2 \leq \mathcal{E}(A,W)^2 &+ \Big(2\alpha\|M\|_{\mathrm{op}} - \tau\Big)\|\hat{A}\|_* + \Big(2\alpha\|M\|_{\mathrm{op}} + \tau\Big)\|A\|_* \\ &+ \Big(2(1-\alpha)\|M\|_\infty - \gamma\Big)\|\hat{A}\|_1 + \Big(2(1-\alpha)\|M\|_\infty + \gamma\Big)\|A\|_1 \\ &+ \Big(2\|\Xi\|_\infty - \kappa\Big)\|\hat{W}\|_1 + \Big(2\|\Xi\|_\infty + \kappa\Big)\|W\|_1,\end{aligned}$$

which concludes the proof of Theorem 1, using (5). □

A.2 Proof of Theorem 2

Let $A \in \mathbb{R}^{n\times n}$ and $W \in \mathbb{R}^{d\times d}$ be fixed, and let $A = U\,\mathrm{diag}(\sigma_1,\dots,\sigma_r)V^\top$ be the SVD of A. Recalling that $\circ$ is the entry-wise product, we have $A = \Theta_A \circ |A| + \Theta_A^\perp \circ A$, where $\Theta_A \in \{0,\pm 1\}^{n\times n}$ is the entry-wise sign matrix of A and $\Theta_A^\perp \in \{0,1\}^{n\times n}$ is the orthogonal sparsity pattern of A.

The definition (4) of $(\hat{A},\hat{W})$ is equivalent to the fact that one can find $\hat{G} \in \partial\mathcal{L}(\hat{A},\hat{W})$ (an element of the subgradient of $\mathcal{L}$ at $(\hat{A},\hat{W})$) that belongs to the normal cone of $\mathcal{A}\times\mathcal{W}$ at $(\hat{A},\hat{W})$. This means that for such a $\hat{G}$, and any $A \in \mathcal{A}$ and $W \in \mathcal{W}$, we have

$$\langle \hat{G}, (\hat{A}-A, \hat{W}-W)\rangle \leq 0. \tag{12}$$

Any subgradient of the function $g(A) = \tau\|A\|_* + \gamma\|A\|_1$ writes

$$Z = \tau Z_* + \gamma Z_1 = \tau\Big(UV^\top + \mathcal{P}_A^\perp(G_*)\Big) + \gamma\Big(\Theta_A + G_1 \circ \Theta_A^\perp\Big)$$

for some $\|G_*\|_{\mathrm{op}} \leq 1$ and $\|G_1\|_\infty \leq 1$ (see for instance the paper by Lewis (1995)). So, if $\hat{Z} \in \partial g(\hat{A})$, we have, by monotonicity of the sub-differential, that for any $Z \in \partial g(A)$

$$\langle \hat{Z}, \hat{A}-A\rangle = \langle \hat{Z}-Z, \hat{A}-A\rangle + \langle Z, \hat{A}-A\rangle \geq \langle Z, \hat{A}-A\rangle,$$

and, by duality, we can find Z such that

$$\langle Z, \widehat{A}-A\rangle = \tau\langle UV^\top, \widehat{A}-A\rangle + \tau\|\mathcal{P}_A^\perp(\widehat{A})\|_* + \gamma\langle \Theta_A, \widehat{A}-A\rangle + \gamma\|\Theta_A^\perp \circ \widehat{A}\|_1.$$

By using the same argument with the function $W \mapsto \|W\|_1$ and by computing the gradient of the empirical risk $(A, W) \mapsto R_n(A, W)$, Equation (12) entails that

$$\begin{aligned}
&2\langle \Phi(\widehat{A} - A_{T+1}, \widehat{W} - W_0), \Phi(\widehat{A} - A, \widehat{W} - W)\rangle \\
&\quad \leq 2\langle (\frac{1}{\sqrt{T}}\mathbf{N}_T, -\frac{1}{\sqrt{d}}N_{T+1}), \Phi(\widehat{A} - A, \widehat{W} - W)\rangle - \tau\langle UV^\top, \widehat{A} - A\rangle - \tau\|\mathcal{P}_A^\perp(\widehat{A})\|_* \\
&\qquad - \gamma\langle \Theta_A, \widehat{A} - A\rangle - \gamma\|\Theta_A^\perp \circ \widehat{A}\|_1 - \kappa\langle \Theta_W, \widehat{W} - W\rangle - \kappa\|\Theta_W^\perp \circ \widehat{W}\|_1.
\end{aligned} \tag{13}$$

Using Pythagora's theorem, we have

$$\begin{aligned}
&2\langle \Phi(\widehat{A} - A_{T+1}, \hat{W} - W_0), \Phi(\widehat{A} - A, \widehat{W} - W)\rangle \\
&\quad = \|\Phi(\widehat{A} - A_{T+1}, \hat{W} - W_0)\|_2^2 + \|\Phi(\widehat{A} - A, \widehat{W} - W)\|_2^2 - \|\Phi(A - A_{T+1}, W - W_0)\|_2^2.
\end{aligned} \tag{14}$$

It shows that if $\langle \Phi(\widehat{A} - A_{T+1}, W - W_0), \Phi(\widehat{A} - A, \widehat{W} - W)\rangle \leq 0$, then Theorem 2 trivially holds. Let us assume that

$$\langle \Phi(\widehat{A} - A_{T+1}, W - W_0), \Phi(\widehat{A} - A, \widehat{W} - W)\rangle > 0. \tag{15}$$

Using Hölder's inequality, we obtain

$$\begin{aligned}
|\langle UV^\top, \hat{A} - A\rangle| &= |\langle UV^\top, \mathcal{P}_A(\hat{A} - A)\rangle| \leq \|UV^\top\|_{\mathrm{op}}\|\mathcal{P}_A(\hat{A} - A)\|_* = \|\mathcal{P}_A(\hat{A} - A)\|_*, \\
|\langle \Theta_A, \hat{A} - A\rangle| &= |\langle \Theta_A, \Theta_A \circ (\hat{A} - A)\rangle| \leq \|\Theta_A\|_\infty\|\Theta_A \circ (\hat{A} - A)\|_1 = \|\Theta_A \circ (\hat{A} - A)\|_1,
\end{aligned}$$

and the same is done for $|\langle \Theta_W, \hat{W} - W\rangle| \leq \|\Theta_W \circ (\hat{W} - W)\|_1$. So, when (15) holds, we obtain by rearranging the terms of (13):

$$\begin{aligned}
&\tau\|\mathcal{P}_A^\perp(\widehat{A} - A)\|_* + \gamma\|\Theta_A^\perp \circ (\widehat{A} - A)\|_1 + \kappa\|\Theta_W^\perp \circ (\widehat{W} - W)\|_1 \\
&\qquad \leq \tau\|\mathcal{P}_A(\hat{A} - A)\|_* + \gamma\|\Theta_A \circ (\hat{A} - A)\|_1 + \kappa\|\Theta_W \circ (\hat{W} - W)\|_1 \\
&\qquad\quad + 2\langle (\frac{1}{\sqrt{T}}\mathbf{N}_T, -\frac{1}{\sqrt{d}}N_{T+1}), \Phi(\widehat{A} - A, \widehat{W} - W)\rangle.
\end{aligned} \tag{16}$$

Using Lemma 5, together with Hölder's inequality, we have for any $\alpha \in (0, 1)$:

$$\begin{aligned}
\langle (\frac{1}{\sqrt{T}}\mathbf{N}_T, -\frac{1}{\sqrt{d}}N_{T+1}), \Phi(\widehat{A} - A, \widehat{W} - W)\rangle &= \langle M, \hat{A} - A\rangle + \langle \Xi, \hat{W} - W\rangle \\
&\leq \alpha\|M\|_{\mathrm{op}}\|\mathcal{P}_A(\hat{A} - A)\|_* + \alpha\|M\|_{\mathrm{op}}\|\mathcal{P}_A^\perp(\hat{A} - A)\|_* \\
&\quad + (1 - \alpha)\|M\|_\infty\|\Theta_A \circ (\hat{A} - A)\|_1 + (1 - \alpha)\|M\|_\infty\|\Theta_A^\perp \circ (\hat{A} - A)\|_1 \\
&\quad + \|\Xi\|_\infty(\|\Theta_W \circ (\hat{W} - W)\|_1 + \|\Theta_W^\perp \circ (\hat{W} - W)\|_1)\ .
\end{aligned} \tag{17}$$

Now, using (16) together with (17), we obtain

$$\begin{aligned}
&\big(\tau - 2\alpha\|M\|_{\mathrm{op}}\big)\|\mathcal{P}_A^\perp(\hat{A} - A)\|_* + \big(\gamma - 2(1 - \alpha)\|M\|_\infty\big)\|\Theta_A^\perp \circ (\hat{A} - A)\|_1 \\
&\qquad + \big(\kappa - 2\|\Xi\|_\infty\big)\|\Theta_W^\perp \circ (\hat{W} - W)\|_1 \\
&\leq \big(\tau + 2\alpha\|M\|_{\mathrm{op}}\big)\|\mathcal{P}_A(\hat{A} - A)\|_* + \big(\gamma + 2(1 - \alpha)\|M\|_\infty\big)\|\Theta_A \circ (\hat{A} - A)\|_1 \\
&\qquad + \big(\kappa + 2\|\Xi\|_\infty\big)\|\Theta_W \circ (\hat{W} - W)\|_1
\end{aligned}$$

which proves, using (6), that

$$\tau\|\mathcal{P}_A^\perp(\hat{A}-A)\|_* + \gamma\|\Theta_A^\perp \circ (\hat{A}-A)\|_1 \leq 5\tau\|\mathcal{P}_A(\hat{A}-A)\|_* + 5\gamma\|\Theta_A \circ (\hat{A}-A)\|_1.$$

This proves that $\hat{A}-A \in \mathcal{C}_2(A,5,\gamma/\tau)$. In the same way, using (16) with $A=\hat{A}$ together with (17), we obtain that $\hat{W}-W \in \mathcal{C}_1(W,5)$.

Now, using together (13), (14) and (17) , and the fact that the Cauchy-Schwarz inequality entails

$$\|\mathcal{P}_A(\hat{A}-A)\|_* \leq \sqrt{\operatorname{rank} A}\|\mathcal{P}_A(\hat{A}-A)\|_F, \quad |\langle UV^\top, \hat{A}-A\rangle| \leq \sqrt{\operatorname{rank} A}\|\mathcal{P}_A(\hat{A}-A)\|_F,$$
$$\|\Theta_A \circ (\hat{A}-A)\|_1 \leq \sqrt{\|A\|_0}\|\Theta_A \circ (\hat{A}-A)\|_F, \quad |\langle \Theta_A, \hat{A}-A\rangle| \leq \sqrt{\|A\|_0}\|\Theta_A \circ (\hat{A}-A)\|_F \ .$$

and similarly for $\hat{W}-W$, we arrive at

$$\begin{aligned}
&\|\Phi(\widehat{A}-A_{T+1}, \hat{W}-W_0)\|_2^2 + \|\Phi(\widehat{A}-A, \widehat{W}-W)\|_2^2 - \|\Phi(A-A_{T+1}, W-W_0)\|_2^2 \\
&\quad\leq \big(2\alpha\|M\|_{\mathrm{op}}+\tau\big)\sqrt{\operatorname{rank} A}\|\mathcal{P}_A(\hat{A}-A)\|_F + \big(2\alpha\|M\|_{\mathrm{op}}-\tau\big)\|\mathcal{P}_A^\perp(\hat{A}-A)\|_* \\
&\qquad+ \big(2\alpha\|M\|_\infty+\gamma\big)\sqrt{\|A\|_0}\|\Theta_A \circ (\hat{A}-A)\|_F + \big(2\alpha\|M\|_\infty-\gamma\big)\|\Theta_A^\perp \circ (\hat{A}-A)\|_1 \\
&\qquad+ \big(2\alpha\|\Xi\|_\infty+\kappa\big)\sqrt{\|W\|_0}\|\Theta_W \circ (\hat{W}-W)\|_F + \big(2\alpha\|\Xi\|_\infty-\kappa\big)\|\Theta_W^\perp \circ (\hat{W}-W)\|_1,
\end{aligned}$$

which leads, using (6), to

$$\begin{aligned}
&\|\Phi(\widehat{A}-A_{T+1}, \hat{W}-W_0)\|_2^2 + \|\Phi(\widehat{A}-A, \widehat{W}-W)\|_2^2 - \|\Phi(A-A_{T+1}, W-W_0)\|_2^2 \\
&\quad\leq \frac{5\tau}{3}\sqrt{\operatorname{rank} A}\|\mathcal{P}_A(\hat{A}-A)\|_F + \frac{5\gamma}{3}\sqrt{\|A\|_0}\|\Theta_A \circ (\hat{A}-A)\|_F + \frac{5\kappa}{3}\sqrt{\|W\|_0}\|\Theta_W \circ (\hat{W}-W)\|_F.
\end{aligned}$$

Since $\hat{A}-A \in \mathcal{C}_2(A,5,\gamma/\tau)$ and $\hat{W}-W \in \mathcal{C}_1(W,5)$, we obtain using Assumption 2 and $ab \leq (a^2+b^2)/2$:

$$\begin{aligned}
&\|\Phi(\widehat{A}-A_{T+1}, \hat{W}-W_0)\|_2^2 + \|\Phi(\widehat{A}-A, \widehat{W}-W)\|_2^2 \\
&\leq \|\Phi(A-A_{T+1}, W-W_0)\|_2^2 + \frac{25}{18}\mu_2(A,W)^2\big(\operatorname{rank} A\tau^2 + \|A\|_0\gamma^2\big) \\
&\quad+ \frac{25}{36}\mu_1(W)^2\|W\|_0\kappa^2 + \|\Phi(\widehat{A}-A, \widehat{W}-W)\|_2^2,
\end{aligned}$$

which concludes the proof of Theorem 2. □

A.3 Proof of Corollary 4

For the proof of (9), we simply use the fact that $\frac{1}{T}\|\mathbf{X}_{T-1}(\hat{W}-W_0)\|_F^2 \leq \mathcal{E}(\hat{A},\hat{W})^2$ and use Theorem 3. Then we take $W=W_0$ in the infimum over A, W.

For (10), we use the fact that since $\hat{W}-W_0 \in \mathcal{C}_1(W_0,5)$, we have (see the Proof of Theorem 2),

$$\begin{aligned}
\|\hat{W}-W_0\|_1 &\leq 6\sqrt{\|W_0\|_0}\|\Theta_W \circ (\hat{W}-W_0)\|_F \\
&\leq 6\sqrt{\|W_0\|_0}\|\mathbf{X}_{T-1}(\hat{W}-W_0)\|_F/\sqrt{T} \\
&\leq 6\sqrt{\|W_0\|_0}\mathcal{E}(\hat{A},\hat{W}),
\end{aligned}$$

and then use again Theorem 3. The proof of (11) follows exactly the same scheme. □

A.4 Concentration Inequalities for the Noise Processes

The control of the noise terms M and Ξ is based on recent results on concentration inequalities for random matrices, developed by Tropp (2012). Moreover, the assumption on the dynamics of the features' noise vector $\{N_t\}_{t\geq 0}$ is quite general, since we only assumed that this process is a martingale increment. Therefore, our control of the noise Ξ rely in particular on martingale theory.

Proposition 6 *Under Assumption 3, the following inequalities hold for any $x > 0$. We have*

$$\Big\|\frac{1}{d}\sum_{j=1}^{d}(N_{T+1})_j\Omega_j\Big\|_{\mathrm{op}} \leq \sigma v_{\Omega,\mathrm{op}}\sqrt{\frac{2(x+\log(2n))}{d}} \tag{18}$$

with a probability larger than $1-e^{-x}$. We have

$$\Big\|\frac{1}{d}\sum_{j=1}^{d}(N_{T+1})_j\Omega_j\Big\|_{\infty} \leq \sigma v_{\Omega,\infty}\sqrt{\frac{2(x+2\log n)}{d}} \tag{19}$$

with a probability larger than $1-2e^{-x}$, and finally

$$\Big\|\frac{1}{T}\sum_{t=1}^{T}\omega(A_{t-1})N_t^\top + \frac{1}{d}\omega(A_T)N_{T+1}^\top\Big\|_{\infty} \leq \sigma\sigma_\omega\sqrt{2e(x+2\log d+\ell_T)}\Big(\frac{1}{\sqrt{T}}+\frac{1}{d}\Big) \tag{20}$$

with a probability larger than $1-15e^{-x}$, where we recall that ℓ_T is given by (8).

Proof For the proofs of Inequalities (18) and (19), we use the fact that $(N_{T+1})_1,\dots,(N_{T+1})_d$ are independent (scalar) sub-gaussian random variables.

From Assumption 3, we have for any $n\times n$ deterministic self-adjoint matrices X_j that $\mathbb{E}[\exp(\lambda(N_{T+1})_jX_j)] \preceq \exp(\sigma^2\lambda^2X_j^2/2)$, where $\preceq$ stands for the semidefinite order on self-adjoint matrices. Using Corollary 3.7 by Tropp (2012), this leads for any $x>0$ to

$$\mathbb{P}\Big[\lambda_{\max}\Big(\sum_{j=1}^{d}(N_{T+1})_jX_j\Big)\geq x\Big] \leq n\exp\Big(-\frac{x^2}{2v^2}\Big), \quad \text{where } v^2=\sigma^2\Big\|\sum_{j=1}^{d}X_j^2\Big\|_{\mathrm{op}}. \tag{21}$$

Then, following Tropp (2012), we consider the dilation operator $\Delta:\mathbb{R}^{n\times n}\to\mathbb{R}^{2n\times 2n}$ given by

$$\Delta(\Omega)=\begin{pmatrix}0 & \Omega\\ \Omega^* & 0\end{pmatrix}.$$

We have

$$\Big\|\sum_{j=1}^{d}(N_{T+1})_j\Omega_j\Big\|_{\mathrm{op}} = \lambda_{\max}\Big(\Delta\Big(\sum_{j=1}^{d}(N_{T+1})_j\Omega_j\Big)\Big) = \lambda_{\max}\Big(\sum_{j=1}^{d}(N_{T+1})_j\Delta(\Omega_j)\Big)$$

and an easy computation gives

$$\Big\|\sum_{j=1}^{d}\Delta(\Omega_j)^2\Big\|_{\mathrm{op}} = \Big\|\sum_{j=1}^{d}\Omega_j^\top\Omega_j\Big\|_{\mathrm{op}} \vee \Big\|\sum_{j=1}^{d}\Omega_j\Omega_j^\top\Big\|_{\mathrm{op}}.$$

So, using (21) with the self-adjoint $X_j = \Delta(\Omega_j)$ gives

$$\mathbb{P}\Big[\Big\|\sum_{j=1}^d (N_{T+1})_j \Omega_j\Big\|_{\rm op} \geq x\Big] \leq 2n\exp\Big(-\frac{x^2}{2v^2}\Big) \text{ where } v^2 = \sigma^2\Big\|\sum_{j=1}^d \Omega_j^\top \Omega_j\Big\|_{\rm op} \vee \Big\|\sum_{j=1}^d \Omega_j\Omega_j^\top\Big\|_{\rm op},$$

which leads easily to (18).

Inequality (19) comes from the following standard bound on the sum of independent sub-gaussian random variables:

$$\mathbb{P}\Big[\Big|\frac{1}{d}\sum_{j=1}^d (N_{T+1})_j(\Omega_j)_{k,l}\Big| \geq x\Big] \leq 2\exp\Big(-\frac{x^2}{2\sigma^2(\Omega_j)_{k,l}^2}\Big)$$

together with an union bound on $1 \leq k, l \leq n$.

Inequality (20) is based on a classical martingale exponential argument together with a peeling argument. We denote by $\omega_j(A_t)$ the coordinates of $\omega(A_t) \in \mathbb{R}^d$ and by $N_{t,k}$ those of N_t, so that

$$\Big(\sum_{t=1}^T \omega(A_{t-1})N_t^\top\Big)_{j,k} = \sum_{t=1}^T \omega_j(A_{t-1})N_{t,k}.$$

We fix j, k and denote for short $\varepsilon_t = N_{t,k}$ and $x_t = \omega_j(A_t)$. Since $\mathbb{E}[\exp(\lambda\varepsilon_t)|\mathcal{F}_{t-1}] \leq e^{\sigma^2\lambda^2/2}$ for any $\lambda \in \mathbb{R}$, we obtain by a recursive conditioning with respect to $\mathcal{F}_{T-1}$, $\mathcal{F}_{T-2}, \ldots, \mathcal{F}_0$, that

$$\mathbb{E}\Big[\exp\Big(\theta\sum_{t=1}^T \varepsilon_t x_{t-1} - \frac{\sigma^2\theta^2}{2}\sum_{t=1}^T x_{t-1}^2\Big)\Big] \leq 1.$$

Hence, using Markov's inequality, we obtain for any $v > 0$:

$$\mathbb{P}\Big[\sum_{t=1}^T \varepsilon_t x_{t-1} \geq x, \sum_{t=1}^T x_{t-1}^2 \leq v\Big] \leq \inf_{\theta>0}\exp(-\theta x + \sigma^2\theta^2 v/2) = \exp\Big(-\frac{x^2}{2\sigma^2 v}\Big),$$

that we rewrite in the following way:

$$\mathbb{P}\Big[\sum_{t=1}^T \varepsilon_t x_{t-1} \geq \sigma\sqrt{2vx}, \sum_{t=1}^T x_{t-1}^2 \leq v\Big] \leq e^{-x}.$$

Let us denote for short $V_T = \sum_{t=1}^T x_{t-1}^2$ and $S_T = \sum_{t=1}^T \varepsilon_t x_{t-1}$. We want to replace v by V_T from the previous deviation inequality, and to remove the event $\{V_T \leq v\}$. To do so, we use a peeling argument. We take $v = T$ and introduce $v_k = ve^k$ so that the event $\{V_T > v\}$ is decomposed into the union of the disjoint sets $\{v_k < V_T \leq v_{k+1}\}$. We introduce also $\ell_T = 2\log\log\Big(\frac{\sum_{t=1}^T x_{t-1}^2}{T} \vee \frac{T}{\sum_{t=1}^T x_{t-1}^2} \vee e\Big)$.

This leads to

$$\begin{aligned}\mathbb{P}\Big[S_T \geq \sigma\sqrt{2eV_T(x+\ell_T)}, V_T > v\Big] &= \sum_{k\geq 0} \mathbb{P}\big[S_T \geq \sigma\sqrt{2eV_T(x+\ell_T)}, v_k < V_T \leq v_{k+1}\Big] \\ &= \sum_{k\geq 0} \mathbb{P}\Big[S_T \geq \sigma\sqrt{2v_{k+1}(x+2\log\log(e^k\vee e))}, v_k < V_T \leq v_{k+1}\Big] \\ &\leq e^{-x}(1+\sum_{k\geq 1} k^{-2}) \leq 3.47e^{-x}.\end{aligned}$$

On $\{V_T \leq v\}$ the proof is the same: we decompose onto the disjoint sets $\{v_{k+1} < V_T \leq v_k\}$ where this time $v_k = ve^{-k}$, and we arrive at

$$\mathbb{P}\Big[S_T \geq \sigma\sqrt{2eV_T(x+\ell_T)}, V_T \leq v\Big] \leq 3.47e^{-x}.$$

This leads to

$$\mathbb{P}\Big[\sum_{t=1}^T \omega_j(A_{t-1})N_{t,k} \geq \sigma\Big(2e\sum_{t=1}^T \omega_j(A_{t-1})^2(x+\ell_{T,j})\Big)^{1/2}\Big] \leq 7e^{-x}$$

for any $1 \leq j,k \leq d$, where we introduced

$$\ell_{T,j} = 2\log\log\Big(\frac{\sum_{t=1}^T \omega_j(A_{t-1})^2}{T} \vee \frac{T}{\sum_{t=1}^T \omega_j(A_{t-1})^2} \vee e\Big).$$

The conclusion follows from an union bound on $1 \leq j,k \leq d$, and from the use of the same argument for the term $\omega(A_T)N_{T+1}^\top$. This concludes the proof of Proposition 6. ■

References

J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10:803–826, 2009.

A. Andreas, M. Pontil, Y. Ying, and C. A. Micchelli. A spectral regularization framework for multi-task structure learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 25–32, 2007.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal of Imaging Sciences*, 2(1):183–202, 2009.

D. P. Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: a survey. *Optimization for Machine Learning*, page 85, 2011.

P. J. Bickel, Y. Ritov, and A. B. Tsybakov. Simultaneous analysis of lasso and Dantzig selector. *Ann. Statist.*, 37(4):1705–1732, 2009.

L. Breiman and J. H. Friedman. Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society (JRSS): Series B (Statistical Methodology)*, 59:3–54, 1997.

E. J. Candès and T. Tao. Decoding by linear programming. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005.

E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *Information Theory, IEEE Transactions on*, 56(5), 2009.

E. J. Candès and M. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 12(51):21–30, 2008.

P. L. Combettes and J. C. Pesquet. Proximal splitting methods in signal processing. *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212, 2011.

R. A. Davis, P. Zang, and T. Zheng. Sparse vector autoregressive modeling. *arXiv preprint arXiv:1207.0520*, 2012.

D. L. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4): 1289–1306, 2006.

T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

S. Gaïffas and G. Lecué. Sharp oracle inequalities for high-dimensional matrix prediction. *Information Theory, IEEE Transactions on*, 57(10):6942 –6957, oct. 2011.

Z. Huang and D. K. J. Lin. The time-series link prediction problem with applications in communication surveillance. *INFORMS J. on Computing*, 21(2):286–303, 2009.

M. Kolar and E. P. Xing. On time varying undirected graphs. In *International Conference on Artificial Intelligence and Statistics*, pages 407–415, 2011.

V. Koltchinskii. Sparsity in penalized empirical risk minimization. *Ann. Inst. Henri Poincaré Probab. Stat.*, 45(1):7–57, 2009a.

V. Koltchinskii. The Dantzig selector and sparsity oracle inequalities. *Bernoulli*, 15(3): 799–828, 2009b.

V. Koltchinskii, K. Lounici, and A. B. Tsybakov. Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion. *Ann. Statist.*, 39(5):2302–2329, 2011.

Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434. ACM, 2008.

Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

A. S. Lewis. The convex analysis of unitarily invariant matrix functions. *J. Convex Anal.*, 2(1-2):173–183, 1995.

D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

S. A. Myers and J. Leskovec. On the convexity of latent social network inference. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

Y. Nardi and A. Rinaldo. Autoregressive process modeling via the lasso procedure. *Journal of Multivariate Analysis*, 102(3):528–549, 2011.

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

E. Richard, N. Baskiotis, T. Evgeniou, and N. Vayatis. Link discovery using graph feature tracking. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

E. Richard, S. Gaiffas, and N. Vayatis. Link prediction in graphs with autoregressive features. In *Advances in Neural Information Processing Systems (NIPS)*, 2012a.

E. Richard, P.-A. Savalle, and N. Vayatis. Estimation of simultaneously sparse and low-rank matrices. In *Proceedings of 29th Annual International Conference on Machine Learning*, 2012b.

E. M. Rogers. *Diffusion of Innovations*. London: The Free Press, 1962.

P. Sarkar, D. Chakrabarti, and A. W. Moore. Theoretical justification of popular link prediction heuristics. In *International Conference on Learning Theory (COLT)*, pages 295–307, 2010.

P. Sarkar, D. Chakrabarti, and M. I. Jordan. Nonparametric link prediction in dynamic networks. In *Proceedings of 29th Annual International Conference on Machine Learning*, 2012.

A. Shojaie, S. Basu, and G. Michailidis. Adaptive thresholding for reconstructing regulatory networks from time course gene expression data. *Statistics In Biosciences*, 2011.

N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*. 2005.

B. Taskar, M. F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

J. A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

R. S. Tsay. *Analysis of Financial Time Series*. Wiley-Interscience; 3rd edition, 2005.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *Preprint*, 2008.

S. A. van de Geer and P. Bühlmann. On the conditions used to prove oracle results for the Lasso. *Electron. J. Stat.*, 3:1360–1392, 2009.

D. Q. Vu, A. Asuncion, D. Hunter, and P. Smyth. Continuous-time regression models for longitudinal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

K. Zhang, T. Evgeniou, V. Padmanabhan, and E. Richard. Content contributor management and network effects in a ugc environment. *Marketing Science*, 2011.

Journal of Machine Learning Research 15 (2014) 595-627 Submitted 10/13; Revised 12/13; Published 2/14

Adaptivity of Averaged Stochastic Gradient Descent to Local Strong Convexity for Logistic Regression

Francis Bach FRANCIS.BACH@ENS.FR
INRIA - Sierra Project-team
Département d'Informatique de l'Ecole Normale Supérieure
Paris, France

Editor: Léon Bottou

Abstract

In this paper, we consider supervised learning problems such as logistic regression and study the stochastic gradient method with averaging, in the usual stochastic approximation setting where observations are used only once. We show that after N iterations, with a constant step-size proportional to $1/R^2\sqrt{N}$ where N is the number of observations and R is the maximum norm of the observations, the convergence rate is always of order $O(1/\sqrt{N})$, and improves to $O(R^2/\mu N)$ where μ is the lowest eigenvalue of the Hessian at the global optimum (when this eigenvalue is greater than $R^2/\sqrt{N}$). Since μ does not need to be known in advance, this shows that averaged stochastic gradient is adaptive to *unknown local* strong convexity of the objective function. Our proof relies on the generalized self-concordance properties of the logistic loss and thus extends to all generalized linear models with uniformly bounded features.

Keywords: stochastic approximation, logistic regression, self-concordance

1. Introduction

The minimization of an objective function which is only available through unbiased estimates of the function values or its gradients is a key methodological problem in many disciplines. Its analysis has been attacked mainly in three scientific communities: stochastic approximation (Fabian, 1968; Ruppert, 1988; Polyak and Juditsky, 1992; Kushner and Yin, 2003; Broadie et al., 2009), optimization (Nesterov and Vial, 2008; Nemirovski et al., 2009), and machine learning (Bottou and Le Cun, 2005; Shalev-Shwartz et al., 2007; Bottou and Bousquet, 2008; Shalev-Shwartz and Srebro, 2008; Shalev-Shwartz et al., 2009; Duchi and Singer, 2009; Xiao, 2010). The main algorithms which have emerged are stochastic gradient descent (a.k.a. Robbins-Monro algorithm), as well as a simple modification where iterates are averaged (a.k.a. Polyak-Ruppert averaging).

For convex optimization problems, the convergence rates of these algorithms depends primarily on the potential *strong convexity* of the objective function (Nemirovski and Yudin, 1983). For μ-strongly convex functions, after n iterations (i.e., n observations), the optimal rate of convergence of function values is $O(1/\mu n)$ while for convex functions the optimal rate is $O(1/\sqrt{n})$, both of them achieved by averaged stochastic gradient with step size respectively proportional to $1/\mu n$ or $1/\sqrt{n}$ (Nemirovski and Yudin, 1983; Agarwal et al.,

2012). For smooth functions, averaged stochastic gradient with step sizes proportional to $1/\sqrt{n}$ achieves them up to logarithmic terms (Bach and Moulines, 2011).

Convex optimization problems coming from supervised machine learning are typically of the form $f(\theta) = \mathbb{E}\big[\ell(y, \langle \theta, x \rangle)\big]$, where $\ell(y, \langle \theta, x \rangle)$ is the loss between the response $y \in \mathbb{R}$ and the prediction $\langle \theta, x \rangle \in \mathbb{R}$, where x is the input data in a Hilbert space $\mathcal{H}$ and linear predictions parameterized by $\theta \in \mathcal{H}$ are considered. They may or may not have strongly convex objective functions. This most often depends on (a) the correlations between covariates x, and (b) the strong convexity of the loss function ℓ. The logistic loss $\ell : u \mapsto \log(1 + e^{-u})$ is not strongly convex unless restricted to a compact set (indeed, restricted to $u \in [-U, U]$, we have $\ell''(u) = e^{-u}(1 + e^{-u})^{-2} \geqslant \frac{1}{4} e^{-U}$). Moreover, in the sequential observation model, the correlations are not known at training time. Therefore, many theoretical results based on strong convexity do not apply (adding a squared norm $\frac{\mu}{2}\|\theta\|^2$ is a possibility, however, in order to avoid adding too much bias, μ has to be small and typically much smaller than $1/\sqrt{n}$, which then makes all strongly-convex bounds vacuous). The goal of this paper is to show that with proper assumptions, namely self-concordance, one can readily obtain favorable theoretical guarantees for logistic regression, namely a rate of the form $O(R^2/\mu n)$ where μ is the lowest eigenvalue of the Hessian at the global optimum, *without any exponentially increasing constant factor* (e.g., with the notations above, without terms of the form e^U).

Another goal of this paper is to design an algorithm and provide an analysis that benefit from *hidden* local strong convexity without requiring to know the local strong convexity constant in advance. In smooth situations, the results of Bach and Moulines (2011) imply that the averaged stochastic gradient method with step sizes of the form $O(1/\sqrt{n})$ is adaptive to the strong convexity of the problem. However the dependence in μ in the strongly convex case is of the form $O(1/\mu^2 n)$, which is sub-optimal. Moreover, the final rate is rather complicated, notably because all possible step-sizes are considered. Finally, it does not apply here because even in low-correlation settings, the objective function of logistic regression cannot be globally strongly convex.

In this paper, we provide an analysis for stochastic gradient with averaging for generalized linear models such as logistic regression, with a step size proportional to $1/R^2\sqrt{n}$ where R is the radius of the data and n the number of observations, showing such adaptivity. In particular, we show that the algorithm can adapt to the *local* strong-convexity constant, that is, the lowest eigenvalue of the Hessian at the optimum. The analysis is done for a finite horizon N and a constant step size decreasing in N as $1/R^2\sqrt{N}$, since the analysis is then slightly easier, though (a) a decaying stepsize could be considered as well, and (b) it could be classically extended to varying step-sizes by a doubling trick (Hazan and Kale, 2001).

2. Stochastic Approximation for Generalized Linear Models

In this section, we present the assumptions our work relies on, as well as related work.

2.1 Assumptions

Throughout this paper, we make the following assumptions. We consider a function f defined on a Hilbert space $\mathcal{H}$, equipped with a norm $\|\cdot\|$. Throughout the paper, we identify the Hilbert space and its dual; thus, the gradients of f also belongs to $\mathcal{H}$ and we

use the same norm on these. Moreover, we consider an increasing family of σ-fields $(\mathcal{F}_n)_{n\geqslant 1}$ and we assume that we are given a deterministic $\theta_0 \in \mathcal{H}$, and a sequence of functions $f_n : \mathcal{H} \to \mathbb{R}$, for $n \geqslant 1$. We make the following assumptions, for a certain $R > 0$:

(A1) Convexity and differentiability of f: f is convex and three-times differentiable.

(A2) Generalized self-concordance of f (Bach, 2010): for all $\theta_1, \theta_2 \in \mathcal{H}$, the function $\varphi : t \mapsto f\big[\theta_1 + t(\theta_2 - \theta_1)\big]$ satisfies: $\forall t \in \mathbb{R}$, $|\varphi'''(t)| \leqslant R\|\theta_1 - \theta_2\|\varphi''(t)$.

(A3) Attained global minimum: f has a global minimum attained at $\theta_* \in \mathcal{H}$.

(A4) Lipschitz-continuity of f_n and f: all gradients of f and f_n are bounded by R, that is, for all $\theta \in \mathcal{H}$,

$$\|f'(\theta)\| \leqslant R \text{ and } \forall n \geqslant 1,\ \|f_n'(\theta)\| \leqslant R \text{ almost surely.}$$

(A5) Adapted measurability: $\forall n \geqslant 1$, f_n is $\mathcal{F}_n$-measurable.

(A6) Unbiased gradients: $\forall n \geqslant 1$, $\mathbb{E}(f_n'(\theta_{n-1})|\mathcal{F}_{n-1}) = f'(\theta_{n-1})$.

(A7) Stochastic gradient recursion: $\forall n \geqslant 1$, $\theta_n = \theta_{n-1} - \gamma_n f_n'(\theta_{n-1})$, where $(\gamma_n)_{n\geqslant 1}$ is a deterministic sequence.

In this paper, we will also consider the averaged iterate $\bar{\theta}_n = \frac{1}{n}\sum_{k=0}^{n-1}\theta_k$, which may be trivially computed on-line through the recursion $\bar{\theta}_n = \frac{1}{n}\theta_{n-1} + \frac{n-1}{n}\bar{\theta}_{n-1}$.

Among the seven assumptions above, the non-standard one is **(A2)**: the notion of self-concordance is an important tool in convex optimization and in particular for the study of Newton's method (Nesterov and Nemirovskii, 1994). It corresponds to having the third derivative bounded by the $\frac{3}{2}$-th power of the second derivative. For machine learning, Bach (2010) has generalized the notion of self-concordance by removing the $\frac{3}{2}$-th power, so that it is applicable to cost functions arising from probabilistic modeling, as shown below. The key consequence of our notion of self-concordance is a relationship shown in Lemma 9 (Section 5) between the norm of a gradient $\|f'(\theta)\|$ and the excess cost function $f(\theta) - f(\theta_*)$, which is the same than for strongly convex functions, but with the local strong convexity constant rather than the global one (which is equal to zero here).

Our set of assumptions corresponds to the following examples (with i.i.d. data, and $\mathcal{F}_n$ equal to the σ-field generated by $x_1, y_1, \dots, x_n, y_n$):

- **Logistic regression**: $f_n(\theta) = \log(1 + \exp(-y_n\langle x_n, \theta\rangle))$, with data x_n uniformly almost surely bounded by R and $y_n \in \{-1, 1\}$. The norm considered here is also the norm of the Hilbert space. Note that this includes other binary classification losses, such as $f_n(\theta) = -y_n\langle x_n, \theta\rangle + \sqrt{1 + \langle x_n, \theta\rangle^2}$.
- **Generalized linear models with uniformly bounded features**: $f_n(\theta) = -\langle\theta, \Phi(x_n, y_n)\rangle + \log\int h(y)\exp\big(\langle\theta, \Phi(x_n, y)\rangle\big)dy$, with $\Phi(x_n, y) \in \mathcal{H}$ almost surely bounded in norm by R, for all observations x_n and all potential responses y in a measurable space. This includes multinomial regression and conditional random fields (Lafferty et al., 2001).
- **Robust regression**: we may use $f_n(\theta) = \varphi(y_n - \langle x_n, \theta\rangle)$, with $\varphi(t) = \log\cosh t = \log\frac{e^t + e^{-t}}{2}$, with a similar boundedness assumption on x_n.

2.2 Running-time Complexity

The stochastic gradient descent recursion $\theta_n = \theta_{n-1} - \gamma_n f_n'(\theta_{n-1})$ operates in full generality in the potentially infinite-dimensional Hilbert space $\mathcal{H}$. There are two practical set-ups where this recursion can be implemented. When $\mathcal{H}$ is finite-dimensional with dimension d, then the complexity of a single iteration is $O(d)$, and thus $O(dn)$ after n iterations. When $\mathcal{H}$ is infinite-dimensional, the recursion can be readily implemented when (a) all functions f_n depend on one-dimensional projections $\langle x_n, \theta \rangle$, that is, are of the form $f_n(\theta) = \varphi_n\big(\langle x_n, \theta \rangle\big)$ for certain random functions φ_n (e.g., $\varphi_n(u) = \ell(y_n, u)$ in machine learning), and (b) all scalar products $K_{ij} = \langle x_i, x_j \rangle$ between x_i and x_j, for $i, j \geqslant 1$, can be computed. This may be done through the classical application of the "kernel trick" (Schölkopf and Smola, 2001; Shawe-Taylor and Cristianini, 2004): if $\theta_0 = 0$, we may represent θ_n as a linear combination of vectors $x_1, \dots, x_n$, that is, $\theta_n = \sum_{i=1}^n \alpha_i x_i$, and the recursion may be written in terms of the weights α_n, through

$$\alpha_n = -\gamma_n x_n \varphi_n'\bigg(\sum_{i=1}^{n-1} \alpha_i K_{ni} \bigg).$$

A key element to notice here is that without regularization, the weights α_i corresponding to previous observations remain constant. The overall complexity of the algorithm is $O(n^2)$ times the cost of evaluating a single kernel function. See Bordes et al. (2005) and Wang et al. (2012) for approaches aiming at reducing the computational load in this setting. Finally, note that in the kernel setting, the function $f(\theta)$ cannot be strongly convex because the covariance operator of x is typically a compact operator, with a sequence of eigenvalues tending to zero (some regularization is then needed).

3. Related Work

In this section, we review related work, first for non-strongly convex problems then for strongly convex problems.

3.1 Non-strongly-convex Functions

When only convexity of the objective function is assumed, several authors (Nesterov and Vial, 2008; Nemirovski et al., 2009; Shalev-Shwartz et al., 2009; Xiao, 2010) have shown that using a step-size proportional to $1/\sqrt{n}$, *together with some form of averaging*, leads to the minimax optimal rate of $O(1/\sqrt{n})$ (Nemirovski and Yudin, 1983; Agarwal et al., 2012). Without averaging, the known convergences rates are suboptimal, that is, averaging is key to obtaining the optimal rate (Bach and Moulines, 2011). Note that the smoothness of the loss does not change the rate, but may help to obtain better constants, with the potential use of acceleration (Lan, 2012). Recent work (Bach and Moulines, 2013) has considered algorithms which improve on the rate $O(1/\sqrt{n})$ for smooth self-concordant losses, such as the square and logistic losses. Their analysis relies on some of the results proved in this paper (in particular the high-order bounds in Section 4).

The compactness of the domain is often used within the algorithm (by using orthogonal projections) and within the analysis (in particular to optimize the step size and obtain high-probability bounds). In this paper, we do not make such compactness assumptions,

since in a machine learning context, the available bound would be loose and hurt practical performance. Note that the analysis of the related dual averaging methods (Nesterov, 2009; Xiao, 2010) has also been carried without compactness assumptions, and previous analyses would also go through in the same set-up for stochastic mirror descent (Nemirovski and Yudin, 1983), at least for bounds in expectation. In the present paper, we derive higher-order bounds and bounds in high-probability where the lack of compactness is harder to deal with.

Another difference between several analyses is the use of decaying step sizes of the form $\gamma_n \propto 1/\sqrt{n}$ vs. the use of a constant step size of the form $\gamma \propto 1/\sqrt{N}$ for a finite known horizon N of iterations. The use of a "doubling trick" as done by Hazan and Kale (2001) for strongly convex optimization, where a constant step size is used for iterations between 2^p and 2^{p+1}, with a constant that is proportional to $1/\sqrt{2^p}$, would allow to obtain an anytime algorithm from a finite horizon one. In order to simplify our analysis, we only consider a finite horizon N and a constant step-size that will be proportional to $1/\sqrt{N}$.

3.2 Strongly-convex Functions

When the function is μ-strongly convex, that is, $\theta \mapsto f(\theta) - \frac{\mu}{2}\|\theta\|^2$ is convex, there are essentially two approaches to obtaining the minimax-optimal rate of $O(1/\mu n)$ (Nemirovski and Yudin, 1983; Agarwal et al., 2012): (a) using a step size proportional to $1/\mu n$ with averaging for non-smooth problems (Nesterov and Vial, 2008; Nemirovski et al., 2009; Xiao, 2010; Shalev-Shwartz et al., 2009; Duchi and Singer, 2009; Lacoste-Julien et al., 2012) or a step size proportional to $1/(R^2 + n\mu)$ also with averaging, for smooth problems, where R^2 is the smoothness constant of the loss of a single observation (Le Roux et al., 2012); (b) for smooth problems, using longer step-sizes proportional to $1/n^\alpha$ for $\alpha \in (1/2, 1)$ *with* averaging (Polyak and Juditsky, 1992; Ruppert, 1988; Bach and Moulines, 2011).

Note that the often advocated step size, that is, of the form C/n where C is larger than $1/\mu$, leads, without averaging to a convergence rate of $O(1/\mu^2 n)$ (Fabian, 1968; Bach and Moulines, 2011), hence with a worse dependence on μ.

The solution (a) requires to have a good estimate of the strong-convexity constant μ, while the second solution (b) does not require to know such estimate and leads to a convergence rate achieving asymptotically the Cramer-Rao lower bound (Polyak and Juditsky, 1992). Thus, this last solution is adaptive to unknown (but positive) amount of strong convexity. However, unless we take the limiting setting $\alpha = 1/2$, it is not adaptive to lack of strong convexity. While the non-asymptotic analysis of Bach and Moulines (2011) already gives a convergence rate in that situation, the bound is rather complicated and also has a suboptimal dependence on μ. Another goal of this paper is to consider a less general result, but more compact and, as already mentioned, a better dependence on the strong convexity constant μ (moreover, as reviewed below, we consider the *local* strong convexity constant, which is much larger).

Finally, note that unless we restrict the support, the objective function for logistic regression cannot be globally strongly convex (since the Hessian tends to zero when $\|\theta\|$ tends to infinity). In this paper we show that stochastic gradient descent with averaging is adaptive to the *local* strong convexity constant, that is, the lowest eigenvalue of the Hessian

of f at the global optimum, without any exponential terms in RD (which would be present if a compact domain of diameter D was imposed and traditional analyses were performed).

3.3 Adaptivity to Unknown Constants

The desirable property of adaptivity to the difficulty of an optimization problem has also been studied in several settings. Gradient descent with constant step size is for example naturally adaptive to the strong convexity of the problem (see, e.g., Nesterov, 2004). In the stochastic context, Juditsky and Nesterov (2010) provide another strategy than averaging with longer step sizes, but for uniform convexity constants.

4. Non-Strongly Convex Analysis

In this section, we study the averaged stochastic gradient method in the non-strongly convex case, that is, without any (global or local) strong convexity assumptions. We first recall existing results in Section 4.1, that bound the expectation of the excess risk leading to a bound in $O(1/\sqrt{N})$. We then show using martingale moment inequalities how all higher-order moments may be bounded in Section 4.2, still with a rate of $O(1/\sqrt{N})$. However, in Section 4.3, we consider the convergence of the squared gradient, with now a rate of $O(1/N)$. This last result is key to obtaining the adaptivity to local strong convexity in Section 5.

4.1 Existing Results

In this section, we review existing results for Lipschitz-continuous non-strongly convex problems (Nesterov and Vial, 2008; Nemirovski et al., 2009; Shalev-Shwartz et al., 2009; Duchi and Singer, 2009; Xiao, 2010). Note that smoothness is not needed here. We consider a constant step size $\gamma_n = \gamma > 0$, for all $n \geqslant 1$, and we denote by $\bar{\theta}_n = \frac{1}{n}\sum_{k=0}^{n-1} \theta_k$ the averaged iterate.

We prove the following proposition, which provides a bound on the expectation of $f(\bar{\theta}_n) - f(\theta_*)$ that decays at rate $O(\gamma + 1/\gamma n)$, hence the usual choice $\gamma \propto 1/\sqrt{n}$:

Lemma 1 *Assume **(A1)** and **(A3-7)**. With constant step size equal to γ, for any $n \geqslant 0$, we have:*

$$\mathbb{E} f\left(\frac{1}{n}\sum_{k=1}^{n} \theta_{k-1}\right) - f(\theta_*) + \frac{1}{2\gamma n}\mathbb{E}\|\theta_n - \theta_*\|^2 \leqslant \frac{1}{2\gamma n}\|\theta_0 - \theta_*\|^2 + \frac{\gamma}{2}R^2.$$

Proof We have the following recursion, obtained from the Lipschitz-continuity of f_n:

$$\begin{aligned}\|\theta_n - \theta_*\|^2 &= \|\theta_{n-1} - \theta_*\|^2 - 2\gamma\langle \theta_{n-1} - \theta_*, f_n'(\theta_{n-1})\rangle + \gamma^2 \|f_n'(\theta_{n-1})\|^2 \\ &\leqslant \|\theta_{n-1} - \theta_*\|^2 - 2\gamma\langle \theta_{n-1} - \theta_*, f'(\theta_{n-1})\rangle + \gamma^2 R^2 + M_n,\end{aligned}$$

with

$$M_n = -2\gamma\langle \theta_{n-1} - \theta_*, f_n'(\theta_{n-1}) - f'(\theta_{n-1})\rangle.$$

We thus get, using the classical result from convexity $f(\theta_{n-1}) - f(\theta_*) \leqslant \langle \theta_{n-1} - \theta_*, f'(\theta_{n-1})\rangle$:

$$2\gamma\big[f(\theta_{n-1}) - f(\theta_*)\big] \leqslant \|\theta_{n-1} - \theta_*\|^2 - \|\theta_n - \theta_*\|^2 + \gamma^2 R^2 + M_n. \tag{1}$$

Summing over integers less than n, this implies:

$$\frac{1}{n}\sum_{k=0}^{n-1} f(\theta_k) - f(\theta_*) + \frac{1}{2\gamma n}\|\theta_n - \theta_*\|^2 \leqslant \frac{1}{2\gamma n}\|\theta_0 - \theta_*\|^2 + \frac{\gamma}{2}R^2 + \frac{1}{2\gamma n}\sum_{k=1}^{n} M_k.$$

We get the desired result by taking expectation in the last inequality, and using the expectation $\mathbb{E}M_k = \mathbb{E}(\mathbb{E}(M_k|\mathcal{F}_{k-1})) = 0$ and $f\big(\frac{1}{n}\sum_{k=0}^{n-1}\theta_k\big) \leqslant \frac{1}{n}\sum_{k=0}^{n-1} f(\theta_k)$. ■

The following corollary considers a specific choice of the step size (note that the bound is only true for the last iterate):

Corollary 2 *Assume **(A1)** and **(A3-7)**. With constant step size equal to $\gamma = \frac{1}{2R^2\sqrt{N}}$, we have:*

$$\begin{aligned}\forall n \in \{1,\dots,N\}, \quad \mathbb{E}\|\theta_n - \theta_*\|^2 &\leqslant \|\theta_0 - \theta_*\|^2 + \frac{1}{4R^2},\\ \mathbb{E}f\left(\frac{1}{N}\sum_{k=1}^{N}\theta_{k-1}\right) - f(\theta_*) &\leqslant \frac{R^2}{\sqrt{N}}\|\theta_0 - \theta_*\|^2 + \frac{1}{4\sqrt{N}}.\end{aligned}$$

Note that if $\|\theta_0 - \theta_*\|^2$ was known, then a better step-size would be $\gamma = \frac{\|\theta_0-\theta_*\|}{R\sqrt{N}}$, leading to a convergence rate proportional to $\frac{R\|\theta_0-\theta_*\|}{\sqrt{N}}$. However, this requires an estimate (or simply an upper-bound) of $\|\theta_0 - \theta_*\|^2$, which is typically not available.

We are going to improve this result in several ways:

- All moments of $\|\theta_n - \theta_*\|^2$ and $f(\bar{\theta}_n) - f(\theta_*)$ will be bounded, leading to a sub-exponential behavior. Note that we do not assume that the iterates are restricted to a predefined bounded set, which is the usual assumption made to derive tail bounds for stochastic approximation (Nesterov and Vial, 2008; Nemirovski et al., 2009; Kakade and Tewari, 2009).
- We are going to show that the squared norm of the gradient at $\bar{\theta}_n = \frac{1}{n}\sum_{k=1}^{n}\theta_{k-1}$ converges at rate $O(1/n)$, even in the non-strongly convex case. This will allow us to derive finer convergence rates in presence of local strong convexity in Section 5.
- The bounds above do not explicitly depend on the dimension of the problem, however, in practice, the quantity $R^2\|\theta_0-\theta_*\|^2$ typically *implicitly* scales linearly in the problem dimension.

4.2 Higher-Order and Tail Bound

In this section, we prove novel higher-order bounds (see the proof in Appendix C), both for any constant step-sizes and then for the specific choice $\gamma = \frac{1}{2R^2\sqrt{N}}$. This will immediately lead to tail bounds.

Proposition 3 *Assume **(A1)** and **(A3-7)**. With constant step size equal to γ, for any $n \geqslant 0$ and integer $p \geqslant 1$, we have:*

$$\mathbb{E}\Big(2\gamma n\big[f(\bar{\theta}_n) - f(\theta^*)\big] + \|\theta_n - \theta_*\|^2\Big)^p \leqslant \big(3\|\theta_0 - \theta_*\|^2 + 20np\gamma^2R^2\big)^p.$$

Corollary 4 *Assume **(A1)** and **(A3-7)**. With constant step size equal to $\gamma = \frac{1}{2R^2\sqrt{N}}$, for any integer $p \geqslant 1$, we have:*

$$\begin{aligned} \forall n \in \{1, \dots, N\}, \quad \mathbb{E}\|\theta_n - \theta_*\|^{2p} &\leqslant \Big[\frac{1}{R^2}\big(3R^2\|\theta_0 - \theta_*\|^2 + 5p\big)\Big]^p, \\ \mathbb{E}\big[f(\bar{\theta}_N) - f(\theta^*)\big]^p &\leqslant \Big[\frac{1}{\sqrt{N}}\big(3R^2\|\theta_0 - \theta_*\|^2 + 5p\big)\Big]^p. \end{aligned}$$

In Appendix C, we first provide two alternative proofs of the same result: (a) our original somewhat tedious proof based on taking powers of the inequality in Equation (1) and using martingale moment inequalities, (b) a shorter proof later derived by Bach and Moulines (2013), that uses Burkholder-Rosenthal-Pinelis inequality (Pinelis, 1994, Theorem 4.1). We also provide in Appendix C a direct proof of the large deviation bound that we now present.

Having a bound on all moments allows immediately to derive large deviation bounds in the same two cases (by applying Lemma 11 from Appendix A):

Proposition 5 *Assume **(A1)** and **(A3-7)**. With constant step size equal to γ, for any $n \geqslant 0$ and $t \geqslant 0$, we have:*

$$\begin{aligned} \mathbb{P}\Big(f(\bar{\theta}_n) - f(\theta_*) \geqslant 30\gamma R^2 t + \frac{3\|\theta_0 - \theta_*\|^2}{\gamma n}\Big) &\leqslant 2\exp(-t), \\ \mathbb{P}\Big(\|\theta_n - \theta_*\|^2 \geqslant 60 n\gamma^2 R^2 t + 6\|\theta_0 - \theta_*\|^2\Big) &\leqslant 2\exp(-t). \end{aligned}$$

Corollary 6 *Assume **(A1)** and **(A3-7)**. With constant step size equal to $\gamma = \frac{1}{2R^2\sqrt{N}}$, for any $t \geqslant 0$ we have:*

$$\begin{aligned} \mathbb{P}\Big(f(\bar{\theta}_N) - f(\theta_*) \geqslant \frac{15t}{\sqrt{N}} + \frac{6R^2\|\theta_0 - \theta_*\|^2}{\sqrt{N}}\Big) &\leqslant 2\exp(-t), \\ \mathbb{P}\Big(\|\theta_N - \theta_*\|^2 \geqslant 15R^{-2}t + 6\|\theta_0 - \theta_*\|^2\Big) &\leqslant 2\exp(-t). \end{aligned}$$

We can make the following observations:

- The results above are obtained by direct application of Proposition 3. In Appendix C, we also provide an alternative direct proof of a slightly weaker result, which was suggested and outlined by Alekh Agarwal (personal communication), and that uses Freedman's inequality for martingales (Freedman, 1975, Theorem 1.6).
- The results above bounding the norm between the last iterate and a global optimum extend to the averaged iterate.
- The iterates θ_n and $\bar{\theta}_n$ do not necessarily converge to θ_* (note that θ_* may not be unique in general anyway).
- Given that $(\mathbb{E}[f(\bar{\theta}_n) - f(\theta_*)]^p)^{1/p}$ is affine in p, we obtain a subexponential behavior, that is, tail bounds similar to an exponential distribution. The same decay was obtained by Nesterov and Vial (2008) and Nemirovski et al. (2009), but with an extra orthogonal projection step that is equivalent in our setting to know a bound on $\|\theta_*\|$, which is in practice not available.

– The constants in the bounds of of Proposition 3 (and thus other results as well) could clearly be improved. In particular, we have, for $p = 1, 2, 3$ (see proof in Appendix E):

$$\begin{aligned}
\mathbb{E}\Big(2\gamma n\big[f(\bar{\theta}_n) - f(\theta^*)\big] + \|\theta_n - \theta_*\|^2\Big) &\leqslant \|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2, \\
\mathbb{E}\Big(2\gamma n\big[f(\bar{\theta}_n) - f(\theta^*)\big] + \|\theta_n - \theta_*\|^2\Big)^2 &\leqslant \big(\|\theta_0 - \theta_*\|^2 + 9n\gamma^2 R^2\big)^2, \\
\mathbb{E}\Big(2\gamma n\big[f(\bar{\theta}_n) - f(\theta^*)\big] + \|\theta_n - \theta_*\|^2\Big)^3 &\leqslant \big(\|\theta_0 - \theta_*\|^2 + 20n\gamma^2 R^2\big)^3.
\end{aligned}$$

4.3 Convergence of Gradients

In this section, we prove higher-order bounds on the convergence of the gradient, with an improved rate $O(1/n)$ for $\|f'(\bar{\theta}_n)\|^2$. In this section, we will need the self-concordance property in Assumption **(A2)**.

Proposition 7 *Assume **(A1-7)**. With constant step size equal to γ, for any $n \geqslant 0$ and integer p, we have:*

$$\left(\mathbb{E}\left\|f'\left(\frac{1}{n}\sum_{k=1}^{n}\theta_{k-1}\right)\right\|^{2p}\right)^{1/2p} \leqslant \frac{R}{\sqrt{n}}\left[8\sqrt{p} + \frac{4p}{\sqrt{n}} + 40R^2\gamma p\sqrt{n} + \frac{3}{\gamma\sqrt{n}}\|\theta_0 - \theta_*\|^2 + \frac{3}{\gamma R\sqrt{n}}\|\theta_0 - \theta_*\|\right].$$

Corollary 8 *Assume **(A1-7)**. With constant step size equal to $\gamma = \frac{1}{2R^2\sqrt{N}}$, for any integer p, we have:*

$$\left(\mathbb{E}\left\|f'\left(\frac{1}{N}\sum_{k=1}^{N}\theta_{k-1}\right)\right\|^{2p}\right)^{1/2p} \leqslant \frac{R}{\sqrt{N}}\left[8\sqrt{p} + \frac{4p}{\sqrt{n}} + 20p + 6R^2\|\theta_0 - \theta_*\|^2 + 6R\|\theta_0 - \theta_*\|\right].$$

We can make the following observations:

– The squared norm of the gradient $\|f'(\bar{\theta}_N)\|^2$ converges at rate $O(1/N)$.

– Given that $(\mathbb{E}\|f'(\bar{\theta}_N)\|^{2p})^{1/2p}$ is affine in p, we obtain a subexponential behavior for $\|f'(\bar{\theta}_N)\|$, that is, tail bounds similar to an exponential distribution.

– The proof of Proposition 7 makes use of the self-concordance assumption (that allows to upperbound deviations of gradients by deviations of function values) together with the proof technique of Polyak and Juditsky (1992).

5. Self-Concordance Analysis for Strongly-Convex Problems

In the previous section, we have shown that $\|f'(\bar{\theta}_N)\|^2$ is of order $O(1/N)$. If the function f was strongly convex with constant $\mu > 0$, this would immediately lead to the bound $f(\bar{\theta}_N) - f(\theta_*) \leqslant \frac{1}{2\mu}\|f'(\bar{\theta}_N)\|^2$, of order $O(1/\mu N)$. However, because of the Lipschitz-continuity of f on the full Hilbert space $\mathcal{H}$, it cannot be strongly convex. In this section, we show how the self-concordance assumption may be used to obtain the exact same behavior, but with μ replaced by the *local* strong convexity constant, which is more likely to be strictly positive.

The required property is summarized in the following proposition about (generalized) self-concordant function (see proof in Appendix B.1):

Lemma 9 *Let f be a convex three-times differentiable function from $\mathcal{H}$ to $\mathbb{R}$, such that for all $\theta_1, \theta_2 \in \mathcal{H}$, the function $\varphi : t \mapsto f\big[\theta_1 + t(\theta_2 - \theta_1)\big]$ satisfies: $\forall t \in \mathbb{R}$, $|\varphi'''(t)| \leqslant R\|\theta_1 - \theta_2\|\varphi''(t)$. Let θ_* be a global minimizer of f and μ the lowest eigenvalue of $f''(\theta_*)$, which is assumed strictly positive.*

$$\text{If } \frac{\|f'(\theta)\|R}{\mu} \leqslant \frac{3}{4} \text{ , then } \|\theta - \theta_*\|^2 \leqslant 4\frac{\|f'(\theta)\|^2}{\mu^2} \text{ and } f(\theta) - f(\theta_*) \leqslant 2\frac{\|f'(\theta)\|^2}{\mu}.$$

We may now use this proposition for the averaged stochastic gradient. For simplicity, we only consider the step-size $\gamma = \frac{1}{2R^2\sqrt{N}}$, and the last iterate (see proof in Appendix F):

Proposition 10 *Assume **(A1-7)**. Assume $\gamma = \frac{1}{2R^2\sqrt{N}}$. Let $\mu > 0$ be the lowest eigenvalue of the Hessian of f at the unique global optimum θ_*. Then:*

$$\begin{aligned}
\mathbb{E}f(\bar{\theta}_N) - f(\theta_*) &\leqslant \frac{R^2}{N\mu}\Big(5R\|\theta_0 - \theta_*\| + 15\Big)^4, \\
\mathbb{E}\big\|\bar{\theta}_N - \theta_*\big\|^2 &\leqslant \frac{R^2}{N\mu^2}\Big(6R\|\theta_0 - \theta_*\| + 21\Big)^4.
\end{aligned}$$

We can make the following observations:

- The proof relies on Lemma 9 and requires a control of the probability that $\frac{\|f'(\bar{\theta}_N)\|R}{\mu} \leqslant \frac{3}{4}$, which is obtained from Proposition 7.
- We conjecture a bound of the form $\left[\frac{R^2}{N\mu}(\square R\|\theta_0 - \theta_*\| + \triangle\sqrt{p})^4\right]^p$ for the p-th order moment of $f(\bar{\theta}_N) - f(\theta_*)$, for some scalar constants $\square$ and $\triangle$.
- The new bound now has the term $R\|\theta_0 - \theta_*\|$ with a fourth power (compared to the bound in Lemma 1, which has a second power), which typically grows with the dimension of the underlying space (or the slowness of the decay of eigenvalues of the covariance operator when $\mathcal{H}$ is infinite-dimensional). It would be interesting to study whether this dependence can be reduced.
- The key elements in the previous proposition are that (a) the constant μ is the *local* convexity constant, and (b) the step-size does not depend on that constant μ, hence the claimed adaptivity.
- The bounds are only better than the non-strongly-convex bounds from Lemma 1, when the Hessian lowest eigenvalue is large enough, that is, $\mu R^2\sqrt{N}$ larger than a fixed constant.
- In the context of logistic regression, even when the covariance matrix of the inputs is invertible, then the only available lower bound on μ is equal to the lowest eigenvalue of the covariance matrix times $\exp(-R\|\theta_*\|)$, which is exponentially small. However, the previous bound is overly pessimistic since it is based on an upper bound on the largest possible value of $\langle x, \theta_*\rangle$. In practice, the actual value of μ is much larger and only a small constant smaller than the lowest eigenvalue of the covariance matrix. In order to assess if this result can be improved, it is interesting to look at the asymptotic result from Polyak and Juditsky (1992) for logistic regression, which leads to a limit rate of $1/n$ times $\operatorname{tr} f''(\theta_*)^{-1}\big(\mathbb{E}f_n'(\theta_*)f_n'(\theta_*)^\top\big)$; note that this rate holds both for the

stochastic approximation algorithm and for the global optimum of the training cost, using standard asymptotic statistics results (Van der Vaart, 1998). When the model is well-specified, that is, the log-odds ratio of the conditional distribution of the label given the input is linear, then $\mathbb{E}f_n'(\theta_*)f_n'(\theta_*)^\top = \mathbb{E}f_n''(\theta_*) = f''(\theta_*)$, and the asymptotic rate is exactly d/n, where d is the dimension of $\mathcal{H}$ (which has to be finite-dimensional for the covariance matrix to be invertible). It would be interesting to see if making the extra assumption of well-specification, we can also get an improved *non-asymptotic* result. When the model is mis-specified however, the quantity $\mathbb{E}f_n'(\theta_*)f_n'(\theta_*)^\top$ may be large even when $f''(\theta_*)$ is small, and the asymptotic regime does not readily lead to an improved bound.

6. Conclusion

In this paper, we have provided a novel analysis of averaged stochastic gradient for logistic regression and related problems. The key aspects of our result are (a) the adaptivity to local strong convexity provided by averaging and (b) the use of self-concordance to obtain a simple bound that does not involve a term which is explicitly exponential in $R\|\theta_0 - \theta_*\|$, which could be obtained by constraining the domain of the iterates.

Our results could be extended in several ways: (a) with a finite and known horizon N, we considered a constant step-size proportional to $1/R^2\sqrt{N}$; it thus seems natural to study the decaying step size $\gamma_n = O(1/R^2\sqrt{n})$, which should, up to logarithmic terms, lead to similar results—and thus likely provide a solution to a a recently posed open problem for online logistic regression (McMahan and Streeter, 2012); (b) an alternative would be to consider a doubling trick where the step-sizes are piecewise constant; also, (c) it may be possible to consider other assumptions, such as exp-concavity (Hazan and Kale, 2001) or uniform convexity (Juditsky and Nesterov, 2010), to derive similar or improved results. Finally, by departing from a plain averaged stochastic gradient recursion, Bach and Moulines (2013) have considered an online Newton algorithm with the same running-time complexity, which leads to a rate of $O(1/n)$ without strong convexity assumptions for logistic regression (though with additional assumptions regarding the distributions of the inputs). It would be interesting to understand if simple assumptions such as the ones made in the present paper are possible while preserving the improved convergence rate.

Acknowledgments

The author was partially supported by the European Research Council (SIERRA Project), and thanks Simon Lacoste-Julien, Eric Moulines and Mark Schmidt for helpful discussions. Morever, Alekh Agarwal suggested and provided a detailed outline of the proof technique based on Freedman's inequality; this was greatly appreciated.

Appendix A. Probability Lemmas

In this appendix, we prove simple lemmas relating bounds on moments to tail bounds, with the traditional use of Markov's inequality. See more general results by Boucheron et al. (2013).

Lemma 11 *Let X be a non-negative random variable such that for some positive constants A and B, and all $p \in \{1, \dots, n\}$,*

$$\mathbb{E}X^p \leqslant (A + Bp)^p.$$

Then, if $t \leqslant \frac{n}{2}$,

$$\mathbb{P}(X \geqslant 3Bt + 2A) \leqslant 2\exp(-t).$$

Proof We have, by Markov's inequality, for any $p \in \{1, \dots, n\}$:

$$\mathbb{P}(X \geqslant 2Bp + 2A) \leqslant \frac{\mathbb{E}X^p}{(2Bp + 2A)^p} \leqslant \frac{(A + Bp)^p}{(2A + 2Bp)^p} = \exp(-\log(2)p).$$

For $u \in [1, n]$, we consider $p = \lfloor u \rfloor$, so that

$$\mathbb{P}(X \geqslant 2Bu + 2A) \leqslant \mathbb{P}(X \geqslant 2Bp + 2A) \leqslant \exp(-\log(2)p) \leqslant 2\exp(-\log(2)u).$$

We take $t = \log(2)u$ and use $2/\log 2 \leqslant 3$. This is thus valid if $t \leqslant \frac{n}{2}$. ■

Lemma 12 *Let X be a non-negative random variable such that for some positive constants A, B and C, and for all $p \in \{1, \dots, n\}$,*

$$\mathbb{E}X^p \leqslant (A\sqrt{p} + Bp + C)^{2p}.$$

Then, if $t \leqslant n$,

$$\mathbb{P}(X \geqslant (2A\sqrt{t} + 2Bt + 2C)^2) \leqslant 4\exp(-t).$$

Proof We have, by Markov's inequality, for any $p \in \{1, \dots, n\}$:

$$\begin{aligned}\mathbb{P}(X \geqslant (2A\sqrt{p} + 2Bp + 2C)^2) &\leqslant \frac{\mathbb{E}X^p}{(2A\sqrt{p} + 2Bp + 2C)^{2p}} \\ &\leqslant \frac{(A\sqrt{p} + Bp + C)^{2p}}{(2A\sqrt{p} + 2Bp + 2C)^{2p}} \leqslant \exp(-\log(4)p).\end{aligned}$$

For $u \in [1, n]$, we consider $p = \lfloor u \rfloor$, so that

$$\begin{aligned}\mathbb{P}(X \geqslant (2A\sqrt{u} + 2Bu + 2C)^2) &\leqslant \mathbb{P}(X \geqslant (2A\sqrt{u} + 2Bu + 2C)^2) \\ &\leqslant \exp(-\log(2)p) \leqslant 4\exp(-\log(4)u).\end{aligned}$$

We take $t = \log(4)u$ and use $\log 4 \geqslant 1$. This is thus valid if $t \leqslant n$. ■

Appendix B. Self-Concordance Properties

In this appendix, we show two lemmas regarding our generalized notion of self-concordance, as well as Lemma 9. For more details, see Bach (2010) and references therein.

The following lemma provide an upper-bound on a one-dimensional self-concordant function at a given point which is based on the gradient at this point and the value and the Hessian at the global minimum. This is key to going in Section 5 from a convergence of gradients to a convergence of function values.

Lemma 13 *Let $\varphi : [0,1] \to \mathbb{R}$ a strictly convex three-times differentiable function such that for some $S > 0$, $\forall t \in [0,1]$, $|\varphi'''(t)| \leqslant S\varphi''(t)$. Assume $\varphi'(0) = 0$, $\varphi''(0) > 0$. Then:*

$$\frac{\varphi'(1)}{\varphi''(0)} S \geqslant 1 - e^{-S} \text{ and } \varphi(1) \leqslant \varphi(0) + \frac{\varphi'(1)^2}{\varphi''(0)}(1+S).$$

Moreover, if $\alpha = \frac{\varphi'(1)S}{\varphi''(0)} < 1$, then $\varphi(1) \leqslant \varphi(0) + \frac{\varphi'(1)^2}{\varphi''(0)} \frac{1}{\alpha} \log \frac{1}{1-\alpha}$. If in addition $\alpha \leqslant \frac{3}{4}$, then $\varphi(1) \leqslant \varphi(0) + 2\frac{\varphi'(1)^2}{\varphi''(0)}$ and $\varphi''(0) \leqslant 2\varphi'(1)$.

Proof By self-concordance, we obtain that the derivative of $u \mapsto \log \varphi''(u)$ is lower-bounded by $-S$. By integrating between 0 and $t \in [0,1]$, we get

$$\log \varphi''(t) - \log \varphi''(0) \geqslant -St \text{ , that is, } \varphi''(t) \geqslant \varphi''(0) e^{-St}, \tag{2}$$

and by integrating between 0 and 1, we obtain (note that we have assumed $\varphi'(0) = 0$):

$$\varphi'(1) \geqslant \varphi''(0) \frac{1 - e^{-S}}{S}. \tag{3}$$

We then get (with a first inequality from convexity of φ, and the last inequality from $e^S \geqslant 1 + S$):

$$\varphi(1) - \varphi(0) \leqslant \varphi'(1) \leqslant \varphi'(1) \frac{\varphi'(1)}{\varphi''(0)} \frac{S}{1 - e^{-S}} = \frac{\varphi'(1)^2}{\varphi''(0)} \left(S + \frac{S}{e^S - 1} \right) \leqslant \frac{\varphi'(1)^2}{\varphi''(0)}(1+S).$$

Equation (3) implies that $\alpha \geqslant 1 - e^{-S}$, which implies, if $\alpha < 1$, $S \leqslant \log \frac{1}{1-\alpha}$. This implies that

$$\varphi(1) - \varphi(0) \leqslant \varphi'(1) \frac{\varphi'(1)}{\varphi''(0)} \frac{S}{1 - e^{-S}} \leqslant \frac{\varphi'(1)^2}{\varphi''(0)} \frac{1}{\alpha} \log \frac{1}{1-\alpha},$$

using the monotonicity of $S \mapsto \frac{S}{1-e^{-S}}$. Finally the last bounds are a consequence of $\frac{S}{\alpha} \leqslant \frac{1}{\alpha} \log \frac{1}{1-\alpha} \leqslant 2$, which is valid for $\alpha \leqslant \frac{3}{4}$.

Note that in Equation (2), we do consider a lower-bound on the Hessian with an exponential factor e^{-St}. The key feature of using self-concordance properties is to get around this exponential factor in the final bound. ■

The following lemma upper-bounds the remainder in the first-order Taylor expansion of the gradient by the remainder in the first-order Taylor expansion of the function. This is important when function values behave well (i.e., converge to the minimal value) while the iterates may not.

Lemma 14 *Let f be a convex three-times differentiable function from $\mathcal{H}$ to $\mathbb{R}$, such that for all $\theta_1, \theta_2 \in \mathcal{H}$, the function $\varphi : t \mapsto f\big[\theta_1 + t(\theta_2 - \theta_1)\big]$ satisfies: $\forall t \in \mathbb{R}$, $|\varphi'''(t)| \leqslant R\|\theta_1 - \theta_2\|\varphi''(t)$. For any $\theta_1, \theta_2 \in H$, we have:*

$$\big\|f'(\theta_1) - f'(\theta_2) - f''(\theta_2)(\theta_2 - \theta_1)\big\| \leqslant R\big[f(\theta_1) - f(\theta_2) - \langle f'(\theta_2), \theta_2 - \theta_1\rangle\big].$$

Proof For a given $z \in \mathcal{H}$ of unit norm, let $\varphi(t) = \big\langle z, f'\big(\theta_2 + t(\theta_1 - \theta_2)\big) - f'(\theta_2) - tf''(\theta_2)(\theta_2 - \theta_1)\big\rangle$ and $\psi(t) = R\big[f(\theta_2 + t(\theta_1 - \theta_2)) - f(\theta_2) - t\langle f'(\theta_2), \theta_2 - \theta_1\rangle\big]$. We have $\varphi(0) = \psi(0) = 0$. Moreover, we have the following derivatives:

$$\begin{aligned}
\varphi'(t) &= \big\langle z, f''\big(\theta_2 + t(\theta_1 - \theta_2)\big) - f''(\theta_2), \theta_1 - \theta_2\big\rangle \\
\varphi''(t) &= f'''\big(\theta_2 + t(\theta_1 - \theta_2)\big)[z, \theta_1 - \theta_2, \theta_1 - \theta_2] \\
&\leqslant R\|z\|_2 f''\big(\theta_2 + t(\theta_1 - \theta_2)\big)[\theta_1 - \theta_2, \theta_1 - \theta_2], \text{ using the Appendix A of Bach (2010),} \\
&= R\big\langle \theta_2 - \theta_1, f''\big(\theta_2 + t(\theta_1 - \theta_2)\big)(\theta_1 - \theta_2)\big\rangle \\
\psi'(t) &= R\big\langle f'\big(\theta_2 + t(\theta_1 - \theta_2)\big) - f'(\theta_2), \theta_1 - \theta_2\big\rangle \\
\psi''(t) &= R\big\langle \theta_2 - \theta_1, f''\big(\theta_2 + t(\theta_1 - \theta_2)\big)(\theta_1 - \theta_2)\big\rangle,
\end{aligned}$$

where $f'''(\theta)$ is the third order tensor of third derivatives. This leads to $\varphi'(0) = \psi'(0) = 0$ and $\varphi''(t) \leqslant \psi''(t)$. We thus have $\varphi(1) \leqslant \psi(1)$ by integrating twice, which leads to the desired result by maximizing with respect to z. ■

B.1 Proof of Lemma 9

We follow the standard proof techniques in self-concordant analysis and define an appropriate function of a single real variable and apply simple lemmas like the ones above.

Define $\varphi : t \mapsto f\big[\theta_* + t(\theta - \theta_*)\big] - f(\theta_*)$. We have

$$\begin{aligned}
\varphi'(t) &= \big\langle f'\big[\theta_* + t(\theta - \theta_*)\big], \theta - \theta_*\big\rangle \\
\varphi''(t) &= \big\langle \theta - \theta_*, f''\big[\theta_* + t(\theta - \theta_*)\big](\theta - \theta_*)\big\rangle \\
\varphi'''(t) &= f'''\big[\theta_* + t(\theta - \theta_*)\big][\theta - \theta_*, \theta - \theta_*, \theta - \theta_*].
\end{aligned}$$

We thus have: $\varphi(0) = \varphi'(0) = 0$, $0 \leqslant \varphi'(1) = \langle f'(\theta), \theta - \theta_*\rangle \leqslant \|f'(\theta)\|\|\theta - \theta_*\|$, $\varphi''(0) = \langle \theta - \theta_*, f''(\theta_*)(\theta - \theta_*)\rangle \geqslant \mu\|\theta - \theta_*\|^2$, and $\varphi(t) \geqslant 0$ for all $t \in [0, 1]$. Moreover, $\varphi'''(t) \leqslant R\|\theta - \theta_*\|\varphi''(t)$ for all $t \in [0, 1]$, that is, Lemma 13 applies with $S = R\|\theta - \theta_*\|$. This leads to the desired result, with $\alpha = \frac{\varphi'(1)S}{\varphi''(0)} \leqslant \frac{\|f'(\theta)\|R}{\mu}$. Note that we also have (using the second inequality in Lemma 13), for all $\theta \in \mathcal{H}$ (and without any assumption on θ):

$$f(\theta) - f(\theta_*) \leqslant \big(1 + R\|\theta - \theta_*\|\big)\frac{\|f'(\theta)\|^2}{\mu}.$$

Appendix C. Proof of Proposition 3

We provide two alternative proofs of the same result: (a) our original somewhat tedious proof in Appendices C.3 and C.4, based on taking powers of the inequality in Equation (1)

and using martingale moment inequalities, (b) a shorter proof in Appendix C.5, later derived by Bach and Moulines (2013), that uses Burkholder-Rosenthal-Pinelis inequality (Pinelis, 1994, Theorem 4.1). Another proof technique was suggested and outlined by Alekh Agarwal (personal communication), that uses Freedman's inequality for martingales (Freedman, 1975, Theorem 1.6); it allows to directly get a tail bound like in Proposition 5. This proof will be presented in Appendix C.6.

Note that the two shorter proofs currently lead to slightly worse constants (or to extra logarithmic factors), that may be improved with more refined derivations.

All proofs start from a similar martingale set-up that we describe in Appendix C.1 and use an almost-sure bound when p gets large (Appendix C.2).

C.1 Bounding Martingales

From the proof of Lemma 1, we have the recursion:

$$2\gamma\big[f(\theta_{n-1}) - f(\theta_*)\big] + \|\theta_n - \theta_*\|^2 \quad \leqslant \quad \|\theta_{n-1} - \theta_*\|^2 + \gamma^2 R^2 + M_n,$$

with

$$M_n = -2\gamma\langle \theta_{n-1} - \theta_*, f_n'(\theta_{n-1}) - f'(\theta_{n-1})\rangle.$$

This leads to, by summing from 1 to n, and using the convexity of f:

$$2\gamma n f\left(\frac{1}{n}\sum_{k=1}^n \theta_{k-1}\right) - 2\gamma n f(\theta^*) + \|\theta_n - \theta_*\|^2 \leqslant A_n,$$

with

$$A_n = \|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2 + \sum_{k=1}^n M_k \geqslant 0.$$

Note that A_n may also be defined recursively as $A_0 = \|\theta_0 - \theta_*\|^2$ and

$$A_n = A_{n-1} + \gamma^2 R^2 + M_n. \tag{4}$$

The random variables (M_n) and (A_n) satisfy the following properties that will proved useful throughout the proof:

(a) Martingale increment: for all $k \geqslant 1$, $\mathbb{E}(M_k|\mathcal{F}_{k-1}) = 0$. This implies that $S_n = \sum_{k=1}^n M_k$ is a martingale.

(b) Boundedness: $|M_k| \leqslant 4\gamma R\|\theta_{k-1} - \theta_*\| \leqslant 4\gamma R A_{k-1}^{1/2}$ almost surely.

C.2 Almost Sure Bound

In this section, we derive an almost sure bound that will be valid for small n. From the stochastic gradient recursion $\theta_n = \theta_{n-1} - \gamma f_n'(\theta_{n-1})$, we get, using Assumption **(A4)** and the triangle inequality:

$$\|\theta_n - \theta_*\| \leqslant \|\theta_{n-1} - \theta_*\| + \gamma\|f_n'(\theta_{n-1})\| \leqslant \|\theta_{n-1} - \theta_*\| + \gamma R \ \text{ almost surely.}$$

This leads to $\|\theta_n - \theta_*\| \leqslant \|\theta_0 - \theta_*\| + n\gamma R$ for all $n \geqslant 0$. This in turn implies that

$$\begin{aligned}
A_n &\leqslant \|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2 + 4\gamma R \sum_{k=1}^{n} \|\theta_{k-1} - \theta_*\| \text{ using } |M_k| \leqslant 4\gamma R \|\theta_{k-1} - \theta_*\|, \\
&\leqslant \|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2 + 4\gamma R \sum_{k=1}^{n} \big[\|\theta_0 - \theta_*\| + (k-1)\gamma R\big] \text{ using the inequality above,} \\
&\leqslant \|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2 + 4\gamma n R \|\theta_0 - \theta_*\| + 2\gamma^2 R^2 n^2 \\
&\qquad\qquad \text{by summing over the first } n-1 \text{ integers,} \\
&\leqslant \|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2 + 2\gamma^2 n^2 R^2 + 2\|\theta_0 - \theta_*\|^2 + 2\gamma^2 R^2 n^2 \text{ using } ab \leqslant \frac{a^2}{2} + \frac{b^2}{2}, \\
&\leqslant 3\|\theta_0 - \theta_*\|^2 + 5n^2\gamma^2 R^2 \text{ almost surely.} \qquad (5)
\end{aligned}$$

This implies that the bound is shown for all $p \geqslant \frac{n}{4}$.

C.3 Derivation of p-th Order Recursion

The first proof works as follows: (a) derive a recursion between the p-th moments and the lower-order moments (this section) and (c) prove the result by induction on p (Appendix C.4). Note that we have to treat separately small values on n in the recursion, for which we use the almost sure bound from Appendix C.2.

Starting from Equation (4), using the binomial expansion formula, we get:

$$\begin{aligned}
A_n^p &\leqslant \Big(A_{n-1} + \gamma^2 R^2 + M_n\Big)^p = \sum_{k=0}^{p} \binom{p}{k} \big(A_{n-1} + \gamma^2 R^2\big)^{p-k} M_n^k \\
&\leqslant \big(A_{n-1} + \gamma^2 R^2\big)^p + p\big(A_{n-1} + \gamma^2 R^2\big)^{p-1} M_n + \sum_{k=2}^{p} \binom{p}{k} \big(A_{n-1} + \gamma^2 R^2\big)^{p-k} \big(4\gamma R A_{n-1}^{1/2}\big)^k.
\end{aligned}$$

This leads to, using $E(M_n|\mathcal{F}_{n-1}) = 0$, upper bounding $\gamma^2 R^2$ by $4\gamma^2 R^2$, and using the binomial expansion formula several times:

$$\begin{aligned}
\mathbb{E}\big[A_n^p \big| \mathcal{F}_{n-1}\big] &\leqslant \big(A_{n-1} + 4\gamma^2 R^2\big)^p + \sum_{k=2}^{p} \binom{p}{k} \big(A_{n-1} + 4\gamma^2 R^2\big)^{p-k} \big(4\gamma R A_{n-1}^{1/2}\big)^k \\
&= \big(A_{n-1} + 4\gamma^2 R^2 + 4\gamma R A_{n-1}^{1/2}\big)^p - 4\gamma R p \big(A_{n-1} + 4\gamma^2 R^2\big)^{p-1} A_{n-1}^{1/2} \\
&\qquad\qquad \text{by isolating the term } k = 1 \text{ in the binomial formula,} \\
&= \big(A_{n-1}^{1/2} + 2\gamma R\big)^{2p} - 4\gamma R p \big(A_{n-1} + 4\gamma^2 R^2\big)^{p-1} A_{n-1}^{1/2} \\
&= \sum_{k=0}^{2p} \binom{2p}{k} A_{n-1}^{k/2} (2\gamma R)^{2p-k} - 4\gamma R p A_{n-1}^{1/2} \sum_{k=0}^{p-1} \binom{p-1}{k} A_{n-1}^k (2\gamma R)^{2(p-1-k)} \\
&= \sum_{k=0}^{2p} A_{n-1}^{k/2} (2\gamma R)^{2p-k} C_k,
\end{aligned}$$

with the constants C_k defined as:

$$\begin{aligned} C_{2q} &= \binom{2p}{2q} \text{ for } q \in \{0, \dots, p\}, \\ C_{2q+1} &= \binom{2p}{2q+1} - 2p\binom{p-1}{q} \text{ for } q \in \{0, \dots, p-1\}. \end{aligned}$$

In particular, $C_0 = 1$, $C_{2p} = 1$, $C_1 = 0$ and $C_{2p-1} = \binom{2p}{2p-1} - 2p\binom{p-1}{p-1} = 0$.

Our goal is now to bounding the values of C_k to obtain Equation (8) below. This will be done by bounding the odd-indexed element by the even-indexed elements.

We have, for $q \in \{1, \dots, p-2\}$,

$$\begin{aligned} C_{2q+1}\frac{2q+1}{2p-2q-1} &\leqslant \binom{2p}{2q+1}\frac{2q+1}{2p-2q-1} \\ &= \frac{(2p)!}{(2q+1)!(2p-2q-1)!}\frac{2q+1}{2p-2q-1} \\ &= \frac{(2p)!}{(2q)!(2p-2q)!}\frac{2p-2q}{2p-2q-1} = \binom{2p}{2q}\frac{2p-2q}{2p-2q-1}. \end{aligned} \tag{6}$$

For the end of the interval above in q, that is, $q = p-2$, we obtain $C_{2q+1}\frac{2q+1}{2p-2q-1} \leqslant C_{2q}\frac{4}{3}$, while for $q \leqslant p-3$, we obtain $C_{2q+1}\frac{2q+1}{2p-2q-1} \leqslant C_{2q}\frac{6}{5}$.

Moreover, for $q \in \{1, \dots, p-2\}$,

$$\begin{aligned} C_{2q+1}\frac{2p-2q-1}{2q+1} &\leqslant \binom{2p}{2q+1}\frac{2p-2q-1}{2q+1} \\ &= \frac{(2p)!}{(2q+1)!(2p-2q-1)!}\frac{2p-2q-1}{2q+1} \\ &= \frac{(2p)!}{(2q+2)!(2p-2q-2)!}\frac{2q+2}{2q+1} = \binom{2p}{2q+2}\frac{2q+2}{2q+1}. \end{aligned} \tag{7}$$

For the end of the interval above in q, that is, $q = 1$, we obtain $C_{2q+1}\frac{2p-2q-1}{2q+1} \leqslant C_{2q+2}\frac{4}{3}$, while for $q \geqslant 2$, we obtain $C_{2q+1}\frac{2p-2q-1}{2q+1} \leqslant C_{2q+2}\frac{6}{5}$.

We have moreover, by using the bound $2\gamma R A_{n-1}^{1/2} \leqslant \frac{\alpha}{2}(2\gamma R)^2 + \frac{1}{2\alpha}A_{n-1}$ for $\alpha = \frac{2q+1}{2p-2q-1}$:

$$\begin{aligned} & C_{2q+1}A_{n-1}^{q+1/2}(2\gamma R)^{2p-2q-1} \\ = \;& C_{2q+1}A_{n-1}^{q}(2\gamma R)^{2p-2q-2}A_{n-1}^{1/2}(2\gamma R) \\ \leqslant \;& C_{2q+1}A_{n-1}^{q}(2\gamma R)^{2p-2q-2}\frac{1}{2}\left[\frac{2q+1}{2p-2q-1}(2\gamma R)^2 + \frac{2p-2q-1}{2q+1}A_{n-1}\right] \\ = \;& \frac{1}{2}C_{2q+1}\frac{2p-2q-1}{2q+1}A_{n-1}^{q+1}(2\gamma R)^{2p-2q-2} + \frac{1}{2}C_{2q+1}\frac{2q+1}{2p-2q-1}A_{n-1}^{q}(2\gamma R)^{2p-2q}. \end{aligned}$$

By combining the previous inequality with Equation (6) and Equation (7), we get that the terms indexed by $2q+1$ are bounded by the terms indexed by $2q+2$ and $2q$. All terms with $q \in \{2, \dots, p-3\}$ are expanded with constants $\frac{3}{5}$, while for $q = 1$ and $q = p-2$, this is

$\frac{2}{3}$. Overall each even term receives a contribution which is less than $\max\{\frac{6}{5}, \frac{3}{5}+\frac{2}{3}, \frac{2}{3}\} = \frac{19}{15}$. This leads to

$$\sum_{q=1}^{p-2} C_{2q+1} A_{n-1}^{q+1/2} (2\gamma R)^{2p-2q-1} \leqslant \frac{19}{15} \sum_{q=0}^{p-1} C_{2q} A_{n-1}^{q} (2\gamma R)^{2p-2q},$$

leading to the recursion that will allow us to derive our result:

$$\mathbb{E}\big[A_n^p \big| \mathcal{F}_{n-1}\big] \leqslant A_{n-1}^p + \frac{34}{15} \sum_{q=0}^{p-1} \binom{2p}{2q} A_{n-1}^q (2\gamma R)^{2p-2q}. \tag{8}$$

C.4 Proof by Induction

We now proceed by induction on p. If we assume that $\mathbb{E}A_k^q \leqslant \big(3\|\theta_0 - \theta_*\|^2 + kq\gamma^2 R^2 B\big)^q$ for all $q < p$, and a certain B (which we will choose to be equal to 20). We first note that if $n \leqslant 4p$, then from Equation (5), we have

$$\begin{aligned} \mathbb{E}A_n^p &\leqslant \big(3\|\theta_0 - \theta_*\|^2 + 5n^2\gamma^2 R^2\ \big)^p \\ &\leqslant \big(3\|\theta_0 - \theta_*\|^2 + 20np\gamma^2 R^2\ \big)^p. \end{aligned}$$

Thus, we only need to consider $n \geqslant 4p$. We then get from Equation (8):

$$\begin{aligned} \mathbb{E}\|\theta_n - \theta_*\|^{2p} &\leqslant \|\theta_0 - \theta_*\|^{2p} + \frac{34}{15} \sum_{k=0}^{n-1} \sum_{q=0}^{p-1} \binom{2p}{2q} \mathbb{E}A_k^q (2\gamma R)^{2p-2q} \\ &\leqslant \|\theta_0 - \theta_*\|^{2p} + \frac{34}{15} \sum_{k=0}^{n-1} \sum_{q=0}^{p-1} \binom{2p}{2q} \big(3\|\theta_0 - \theta_*\|^2 + kq\gamma^2 R^2 B\big)^q (2\gamma R)^{2p-2q}, \end{aligned}$$

using the induction hypothesis. We may now sum with respect to k:

$$\begin{aligned} &\mathbb{E}\|\theta_n - \theta_*\|^{2p} \\ \leqslant\ & \|\theta_0 - \theta_*\|^{2p} + \frac{34}{15} \sum_{q=0}^{p-1} \binom{2p}{2q} (2\gamma R)^{2p-2q} \sum_{k=0}^{n-1} \big(3\|\theta_0 - \theta_*\|^2 + kq\gamma^2 R^2 B\big)^q \\ \leqslant\ & \|\theta_0 - \theta_*\|^{2p} + \frac{34}{15} \sum_{q=0}^{p-1} \binom{2p}{2q} (2\gamma R)^{2p-2q} \sum_{j=0}^{q} 3^j \|\theta_0 - \theta_*\|^{2j} \binom{q}{j} \big(q\gamma^2 R^2 B\big)^{q-j} \frac{n^{q-j+1}}{q-j+1} \\ & \text{using } \sum_{k=0}^{n-1} k^\alpha \leqslant \frac{n^{\alpha+1}}{\alpha+1} \text{ for any } \alpha > 0, \\ =\ & \|\theta_0 - \theta_*\|^{2p} + \frac{34}{15} \sum_{j=0}^{p-1} 3^j \|\theta_0 - \theta_*\|^{2j} (4\gamma^2 R^2 n)^{p-j} \sum_{q=j}^{p-1} \binom{2p}{2q} \binom{q}{j} (\tfrac{qB}{4})^{q-j} \frac{n^{q-p+1}}{q-j+1}, \end{aligned}$$

by changing the order of summations. We now aim to show that it is less than

$$\Big(3\|\theta_0 - \theta_*\|^2 + kp\gamma^2 R^2 B\Big)^p = 3^p \|\theta_0 - \theta_*\|^{2p} + \sum_{j=0}^{p-1} 3^j \|\theta_0 - \theta_*\|^{2j} (\gamma^2 R^2 n)^{p-j} (Bp)^{p-j} \binom{p}{j}.$$

By comparing all terms in $\|\theta_0 - \theta_*\|^{2j}$, this is true as soon as for all $j \in \{0, \dots, p-1\}$,

$$\frac{34}{15} \sum_{q=j}^{p-1} \binom{2p}{2q} \binom{q}{j} (qB/4)^{q-j} \frac{1}{q-j+1} \frac{1}{n^{p-q-1}} \leqslant (Bp/4)^{p-j} \binom{p}{j}$$

$$\Leftrightarrow \frac{34}{15} \sum_{k=0}^{p-1-j} \binom{2p}{2k+2} \binom{p-1-k}{j} ((p-1-k)B/4)^{p-1-k-j} \frac{1}{p-k-j} \frac{1}{n^k} \leqslant (Bp/4)^{p-j} \binom{p}{j},$$

obtained by using the change of variable $k = p - 1 - q$. This is implied by, using $n \geqslant 4p$:

$$\frac{136}{15} \sum_{k=0}^{p-1-j} B^{-1-k} p^{-k-p+j} \binom{2p}{2k+2} \frac{\binom{p-1-k}{j}}{\binom{p}{j}} (p-1-k)^{p-1-k-j} \frac{1}{p-k-j} \leqslant 1.$$

By expanding the binomial coefficients and simplifying by $p - k - j$, this is equivalent to

$$\frac{136}{15} \sum_{k=0}^{p-1-j} B^{-1-k} p^{-k-p+j} \binom{2p}{2k+2} \frac{(p-1-k)\cdots(p-k-j+1)}{p\cdots(p-j+1)} (p-1-k)^{p-1-k-j} \leqslant 1.$$

We may now write

$$\begin{aligned} \frac{(p-1-k)\cdots(p-k-j+1)}{p\cdots(p-j+1)} &= \frac{(p-1-k)!}{(p-k-j)!} \frac{(p-j)!}{p!} = \frac{(p-1-k)!}{p!} \frac{(p-j)!}{(p-k-j)!} \\ &= \frac{(p-j)\cdots(p-k-j+1)}{p\cdots(p-k)}, \end{aligned}$$

so that we only need to show that

$$\frac{136}{15} \sum_{k=0}^{p-1-j} B^{-1-k} p^{-k-p+j} \binom{2p}{2k+2} \frac{(p-j)\cdots(p-k-j+1)}{p\cdots(p-k)} (p-1-k)^{p-1-k-j} \leqslant 1.$$

We have, by bounding all terms then than p by p:

$$\begin{aligned}
& \frac{136}{15}\sum_{k=0}^{p-1-j} A^{-1-k}p^{-k-p+j}\binom{2p}{2k+2}\frac{(p-j)\cdots(p-k-j+1)}{p\cdots(p-k)}(p-1-k)^{p-1-k-j} \\
\leqslant\ & \frac{136}{15}\sum_{k=0}^{p-1-j} A^{-1-k}p^{-k-p+j}\binom{2p}{2k+2}\frac{p^k}{p\cdots(p-k)}p^{p-1-k-j} \\
=\ & \frac{136}{15}\sum_{k=0}^{p-1-j} A^{-1-k}p^{-k-1}\binom{2p}{2k+2}\frac{1}{p\cdots(p-k)} \\
=\ & \frac{136}{15}\sum_{k=0}^{p-1-j} A^{-1-k}\frac{p^{-k-1}}{(2k+2)!}\frac{2p(2p-1)\cdots(2p-2k-1)}{p\cdots(p-k)} \\
=\ & \frac{136}{15}\sum_{k=0}^{p-1-j} A^{-1-k}\frac{p^{-2-1}2^{2k+2}}{(2k+2)!}\frac{p(p-1/2)\cdots(p-k-1/2)}{p\cdots(p-k)} \\
\leqslant\ & \frac{136}{15}\sum_{k=0}^{p-1-j} A^{-1-k}\frac{2^{2k+2}}{(2k+2)!} \\
& \text{by associating all } 2k+2 \text{ terms in ratios which are all less than 1,} \\
\leqslant\ & \frac{136}{15}\sum_{k=0}^{+\infty}\frac{(2/\sqrt{A})^{2k+2}}{(2k+2)!} = \frac{136}{15}\big[\cosh(2/\sqrt{A})-1\big] < 1 \text{ if } A \leqslant 20.
\end{aligned}$$

We thus get the desired result $\mathbb{E}A_n^p \leqslant \big(3\|\theta_0-\theta_*\|^2 + 20np\gamma^2R^2\big)^p$, and the proposition is proved by induction.

C.5 Alternative Proof Using Burkholder-Rosenthal-Pinelis Inequality

In this section, we present (a slightly modified version of) the proof from Bach and Moulines (2013) which is based on Burkholder-Rosenthal-Pinelis inequality (Pinelis, 1994, Theorem 4.1), which we now recall.

C.5.1 BRP Inequality

Throughout the proof, we use the notation for $X \in \mathcal{H}$ a random vector, and p any real number greater than 1, $\|X\|_p = \big(\mathbb{E}\|X\|^p\big)^{1/p}$. We first recall the Burkholder-Rosenthal-Pinelis (BRP) inequality (Pinelis, 1994, Theorem 4.1). Let $p \in \mathbb{R}$, $p \geqslant 2$ and $(\mathcal{F}_n)_{n\geqslant 0}$ be a sequence of increasing σ-fields, and $(X_n)_{n\geqslant 1}$ an adapted sequence of elements of $\mathcal{H}$, such that $\mathbb{E}\big[X_n|\mathcal{F}_{n-1}\big] = 0$, and $\|X_n\|_p$ is finite. Then,

$$\begin{aligned}
\left\| \sup_{k\in\{1,\dots,n\}} \left\| \sum_{j=1}^k X_j \right\| \right\|_p &\leqslant \sqrt{p}\left\| \sum_{k=1}^n \mathbb{E}\big[\|X_k\|^2|\mathcal{F}_{k-1}\big]\right\|_{p/2}^{1/2} + p\left\| \sup_{k\in\{1,\dots,n\}} \|X_k\| \right\|_p \qquad (9)\\
&\leqslant \sqrt{p}\left\| \sum_{k=1}^n \mathbb{E}\big[\|X_k\|^2|\mathcal{F}_{k-1}\big]\right\|_{p/2}^{1/2} + p\left\| \sup_{k\in\{1,\dots,n\}} \|X_k\|^2 \right\|_{p/2}^{1/2}.
\end{aligned}$$

C.5.2 Proof of Proposition 3 (With Slightly Worse Constants)

We use BRP's inequality in Equation (9) to get, for $p \in [2, n/4]$:

$$\begin{aligned}
\Big\| \sup_{k\in\{0,\dots,n\}} A_k \Big\|_p &\leqslant \|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2 + \sqrt{p}\, \Big\| 16\gamma^2 R^2 \sum_{k=1}^n \|\theta_{k-1} - \theta_*\|^2 \Big\|_{p/2}^{1/2} \\
&\qquad + p\, \Big\| \sup_{k\in\{1,\dots,n\}} 4\gamma R \|\theta_{k-1} - \theta_*\| \Big\|_p \\
&\leqslant \|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2 + \sqrt{p}\, 4\gamma R\sqrt{n} \Big\| \sup_{k\in\{0,\dots,n-1\}} A_k \Big\|_{p/2}^{1/2} \\
&\qquad + p\, 4\gamma R \Big\| \sup_{k\in\{0,\dots,n-1\}} A_k^{1/2} \Big\|_p \\
&\leqslant \|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2 + 4\gamma R \Big\| \sup_{k\in\{0,\dots,n-1\}} A_k \Big\|_{p/2}^{1/2} \big(\sqrt{pn} + p\big).
\end{aligned}$$

Thus if $B = \Big\| \sup_{k\in\{0,\dots,n\}} A_k \Big\|_p$, we have (using $p \leqslant n/4$, which implies $\sqrt{pn} + p \leqslant \frac{3}{2}\sqrt{pn}$):

$$B \leqslant \|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2 + 6\gamma R B^{1/2} \sqrt{pn}.$$

By solving this quadratic inequality, we get:

$$\big(B^{1/2} - 3\gamma R\sqrt{pn}\big)^2 \leqslant \|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2 + 9\gamma^2 R^2 pn,$$

which implies

$$\begin{aligned}
B^{1/2} &\leqslant 3\gamma R\sqrt{pn} + \sqrt{\|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2 + 9\gamma^2 R^2 pn} \\
B &\leqslant 2 \times 9\gamma^2 R^2 pn + 2 \times \big(\|\theta_0 - \theta_*\|^2 + n\gamma^2 R^2 + 9\gamma^2 R^2 pn\big) \\
&\leqslant 40\gamma^2 R^2 pn + 2\|\theta_0 - \theta_*\|^2.
\end{aligned}$$

The previous statement is valid for $p \geqslant 2$ and trivial for $p = 1$. From Appendix C.2, we only need to have the result for $p \leqslant \frac{n}{4}$. Thus the bound is slightly worse (but could be clearly improved with more care, for example, by using induction on n).

C.6 Alternative Proof Using Freedman's Inequality

In the previous section, we have used p-th order moment martingale inequalities that relate the norm of a martingale to the norm of its predictable quadratic variation process. Similar results may be obtained for tail bounds through Freedman's inequality (Freedman, 1975, Theorem 1.6). This proof technique was suggested and outlined by Alekh Agarwal (personal communication).

C.6.1 Freedman's Inequality and Extensions

Let (X_n) be a real-valued martingale increment adapted to the increasing sequence of σ-fields $(\mathcal{F}_n)$, that is, such that $\mathbb{E}(X_n|\mathcal{F}_n) = 0$, that is almost surely bounded, that is, $|X_n| \leqslant R$

almost surely. Let $\Sigma_n = \sum_{k=1}^n \mathbb{E}(X_k^2|\mathcal{F}_{k-1})$ the predictable quadratic variation process. Then for any constants t and σ^2,

$$\mathbb{P}\Big(\max_{k\in\{1,\dots,n\}} \sum_{i=1}^k X_i \geqslant t, \Sigma_n \leqslant \sigma^2 \Big) \leqslant 2\exp\Big(\frac{-t^2}{2(\sigma^2 + Rt/3)} \Big).$$

When (X_n) are independent random variables, this recovers Bernstein's inequality. From this bound, one may derive the following bound (Kakade and Tewari, 2009); with probability $1 - 4(\log n)\delta$, we have:

$$\max_{k\in\{1,\dots,n\}} \sum_{i=1}^k X_i \leqslant \max\Big\{ 2\sqrt{\Sigma_n}, 3R\sqrt{\log\tfrac{1}{\delta}} \Big\} \sqrt{\log\tfrac{1}{\delta}} \leqslant 2\sqrt{\Sigma_n}\sqrt{\log\tfrac{1}{\delta}} + 3R\log\tfrac{1}{\delta}. \qquad (10)$$

Note that the result of Kakade and Tewari (2009) considers only $\sum_{i=1}^n X_i$ rather than $\max_{k\in\{1,\dots,n\}} \sum_{i=1}^k X_i$, but that the extension of their proof is straightforward.

C.6.2 Proof of Proposition 5 (With Slightly Worse Constants and Scalings)

We can now apply the inequality in Equation (10) to (M_n). We have $|M_n| \leqslant 4\gamma R\|\theta_{n-1} - \theta_*\| \leqslant 4\gamma R\big(\|\theta_0 - \theta_*\| + n\gamma R\big)$ almost surely. Moreover, $\mathbb{E}(M_n^2|\mathcal{F}_{n-1}) \leqslant 16\gamma^2R^2\|\theta_{n-1} - \theta_*\|^2 \leqslant 16\gamma^2R^2A_{n-1}$.

This leads to with probability greater than $1 - 4(\log n)\delta$,

$$\begin{aligned}
\max_{k\in\{1,\dots,n\}} A_k &\leqslant \|\theta_0 - \theta_*\|^2 + n\gamma^2R^2 + 8\gamma R\sqrt{\sum_{k=1}^{n-1} A_k}\sqrt{\log\tfrac{1}{\delta}} + 12\gamma R\big(\|\theta_0 - \theta_*\| + n\gamma R\big)\log\tfrac{1}{\delta} \\
&\leqslant \|\theta_0 - \theta_*\|^2 + n\gamma^2R^2 + 8\gamma R\sqrt{n}\max_{k\in\{1,\dots,n\}} \sqrt{A_k}\sqrt{\log\tfrac{1}{\delta}} \\
&\qquad + 12\gamma R\big(\|\theta_0 - \theta_*\| + n\gamma R\big)\log\tfrac{1}{\delta}.
\end{aligned}$$

We may now solve the quadratic inequality in $\max_{k\in\{1,\dots,n\}} \sqrt{A_k}$. This leads to

$$\begin{aligned}
&\Big(\max_{k\in\{1,\dots,n\}} \sqrt{A_k} - 4\gamma R\sqrt{n}\sqrt{\log\tfrac{1}{\delta}} \Big)^2 \\
\leqslant\ & \|\theta_0 - \theta_*\|^2 + n\gamma^2R^2 + 12\gamma R\big(\|\theta_0 - \theta_*\| + n\gamma R\big)\log\tfrac{1}{\delta} + 16\gamma^2R^2n\log\tfrac{1}{\delta} \\
=\ & \|\theta_0 - \theta_*\|^2 + n\gamma^2R^2 + \big(12\gamma R\|\theta_0 - \theta_*\| + 28n\gamma^2R^2\big)\log\tfrac{1}{\delta}.
\end{aligned}$$

Then

$$\begin{aligned}
&\max_{k\in\{1,\dots,n\}} \sqrt{A_k} \\
\leqslant\ & 4\gamma R\sqrt{n}\sqrt{\log\frac{1}{\delta}} + \|\theta_0 - \theta_*\| + \sqrt{n}\gamma R + \sqrt{12\gamma R\|\theta_0 - \theta_*\| + 28n\gamma^2R^2}\ \sqrt{\log\frac{1}{\delta}}
\end{aligned}$$

and

$$\begin{aligned}
&\max_{k\in\{1,\dots,n\}} A_k \\
\leqslant\ & 64\gamma^2R^2n\log\frac{1}{\delta} + 4\|\theta_0-\theta_*\|^2 + 4n\gamma^2R^2 + 4\big(12\gamma R\|\theta_0-\theta_*\| + 28n\gamma^2R^2\big)\,\log\frac{1}{\delta} \\
\leqslant\ & 4\|\theta_0-\theta_*\|^2 + 4n\gamma^2R^2 + \Big(64\gamma^2R^2n + 48\gamma R\|\theta_0-\theta_*\| + 112n\gamma^2R^2\Big)\log\frac{1}{\delta} \\
\leqslant\ & 4\|\theta_0-\theta_*\|^2 + 4n\gamma^2R^2 + \Big(176\gamma^2R^2n + 48\gamma R\|\theta_0-\theta_*\|\Big)\log\frac{1}{\delta}.
\end{aligned}$$

We thus recover a tail bound which is very similar to the one obtained in Proposition 5, with the following differences: the additional term $48\gamma R\|\theta_0-\theta_*\|$ is unimportant because $\gamma = O(N^{-1/2})$; however, because the extension of Freedman's inequality is satisfied with probability $1-4(\log n)\delta$, this proof technique loses a logarithmic factor.

Appendix D. Proof of Proposition 7

The proof is organized in two parts: we first show a bound on the averaged gradient $\frac{1}{n}\sum_{k=1}^n f'(\theta_{k-1})$, then relate it to the gradient at the averaged iterate, that is, $f'\Big(\frac{1}{n}\sum_{k=1}^n\theta_{k-1}\Big)$, using self-concordance.

D.1 Bound on $\frac{1}{n}\sum_{k=1}^n f'(\theta_{k-1})$

We have, following Polyak and Juditsky (1992) and Bach and Moulines (2011):

$$f_n'(\theta_{n-1}) = \frac{1}{\gamma}(\theta_{n-1}-\theta_n),$$

which implies, by summing over all integers between 1 and n:

$$\frac{1}{n}\sum_{k=1}^n f'(\theta_{k-1}) \quad = \quad \frac{1}{n}\sum_{k=1}^n\big[f'(\theta_{k-1}) - f_k'(\theta_{k-1})\big] + \frac{1}{\gamma n}(\theta_0-\theta_*) + \frac{1}{\gamma n}(\theta_*-\theta_n).$$

We denote $X_k = \frac{1}{n}\big[f'(\theta_{k-1}) - f_k'(\theta_{k-1})\big] \in \mathcal{H}$. We have: $\|X_k\| \leqslant \frac{2R}{n}$ almost surely and $\mathbb{E}(X_k|\mathcal{F}_{k-1}) = 0$, with $\big(\sum_{k=1}^n\mathbb{E}(\|X_k\|^2|\mathcal{F}_{k-1})\big)^{1/2} \leqslant \frac{2R}{\sqrt{n}}$. We may thus apply the Burkholder-Rosenthal-Pinelis inequality (Pinelis, 1994, Theorem 4.1), and get:

$$\left[\mathbb{E}\left\|\frac{1}{n}\sum_{k=1}^n\big[f'(\theta_{k-1}) - f_k'(\theta_{k-1})\big]\right\|^{2p}\right]^{1/2p} \quad \leqslant \quad 2p\frac{2R}{n} + \sqrt{2p}\frac{2R}{n^{1/2}}.$$

This leads to, using Proposition 3 and Minkowski's inequality:

$$\begin{aligned}
&\left[\mathbb{E}\left\|\frac{1}{n}\sum_{k=1}^n f'(\theta_{k-1})\right\|^{2p}\right]^{1/2p} \\
\leqslant\ & \left[\mathbb{E}\left\|\frac{1}{n}\sum_{k=1}^n \big[f'(\theta_{k-1}) - f_k'(\theta_{k-1})\big]\right\|^{2p}\right]^{1/2p} + \frac{1}{\gamma n}\|\theta_0-\theta_*\| + \frac{1}{\gamma n}\big[\mathbb{E}\|\theta_*-\theta_n\|^{2p}\big]^{1/2p} \\
\leqslant\ & 2p\frac{2R}{n} + \sqrt{2p}\frac{2R}{n^{1/2}} + \frac{1}{\gamma n}\|\theta_0-\theta_*\| + \big[\frac{1}{\gamma n}\sqrt{3\|\theta_0-\theta_*\|^2 + 20np\gamma^2R^2}\big] \\
\leqslant\ & 2p\frac{2R}{n} + \sqrt{2p}\frac{2R}{n^{1/2}} + \frac{1}{\gamma n}\|\theta_0-\theta_*\| + \big[\frac{\sqrt{3}}{\gamma n}\|\theta_0-\theta_*\| + \frac{1}{\gamma n}\sqrt{20np}\gamma R\big] \\
\leqslant\ & \frac{4pR}{n} + \sqrt{2p}\frac{2R}{n^{1/2}} + \frac{2}{\gamma n}\|\theta_0-\theta_*\| + \frac{1}{\gamma n}\sqrt{20np}\gamma R \\
\leqslant\ & \frac{4pR}{n} + \sqrt{p}\frac{R}{\sqrt{n}}\big[2\sqrt{2}+\sqrt{20}\big] + \frac{1+\sqrt{3}}{\gamma n}\|\theta_0-\theta_*\| \\
\leqslant\ & \frac{4pR}{n} + 8\sqrt{p}\frac{R}{\sqrt{n}} + \frac{3}{\gamma n}\|\theta_0-\theta_*\|. \qquad (11)
\end{aligned}$$

D.2 Using Self-Concordance

Using the self-concordance property of Lemma 14 several times, we obtain:

$$\begin{aligned}
&\left\|\frac{1}{n}\sum_{k=1}^n f'(\theta_{k-1}) - f'\left(\frac{1}{n}\sum_{k=1}^n \theta_{k-1}\right)\right\| \\
=\ & \left\|\frac{1}{n}\sum_{k=1}^n \big[f'(\theta_{k-1}) - f'(\theta_*) - f''(\theta_*)(\theta_{k-1}-\theta_*)\big]\right. \\
&\qquad \left. - f'\left(\frac{1}{n}\sum_{k=1}^n \theta_{k-1}\right) + f'(\theta_*) + f''(\theta_*)\left(\frac{1}{n}\sum_{k=1}^n \theta_{k-1} - \theta_*\right)\right\| \\
\leqslant\ & \frac{R}{n}\sum_{k=1}^n \big[f(\theta_{k-1}) - f(\theta_*) - \langle f'(\theta_*), \theta_{k-1}-\theta_*\rangle\big] \\
&\qquad + R\left[f\left(\frac{1}{n}\sum_{k=1}^n \theta_{k-1}\right) - f(\theta_*) + \left\langle f'(\theta_*), \frac{1}{n}\sum_{k=1}^n \theta_{k-1} - \theta_*\right\rangle\right] \\
\leqslant\ & 2R\left(\frac{1}{n}\sum_{k=1}^n f(\theta_{k-1}) - f(\theta_*)\right) \text{ using the convexity of } f.
\end{aligned}$$

This leads to, using Proposition 3:

$$\begin{aligned}
&\left(\mathbb{E}\left\|\frac{1}{n}\sum_{k=1}^n f'(\theta_{k-1}) - f'\left(\frac{1}{n}\sum_{k=1}^n \theta_{k-1}\right)\right\|^{2p}\right)^{1/2p} \\
\leqslant\ & 2R\left(\mathbb{E}\left[\frac{1}{n}\sum_{k=1}^n f(\theta_{k-1}) - f(\theta_*)\right]^{2p}\right)^{1/2p} \leqslant \frac{2R}{2\gamma n}\left(3\|\theta_0-\theta_*\|^2 + 40np\gamma^2R^2\right). \qquad (12)
\end{aligned}$$

Summing Equation (11) and Equation (12) leads to the desired result.

Appendix E. Results for Small p

In Proposition 3, we may replace the bound $3\|\theta_0 - \theta_*\|^2 + 20np\gamma^2R^2$ with a bound with smaller constants for $p = 1, 2, 3$ (to be used in proofs of results in Section 5). This is done using the same proof principle but finer derivations, as follows. We denote $\gamma^2R^2 = b$ and $\|\theta - \theta_*\|^2 = a$, and consider the following inequalities which we have considered in the proof of Proposition 3:

$$\begin{aligned} A_n^p &\leqslant (A_{n-1} + b + M_n)^p \\ M_n &\leqslant 4b^{1/2}A_{n-1}^{1/2} \text{ and } \mathbb{E}(M_n|\mathcal{F}_{n-1}) = 0, \\ A_0 &= a. \end{aligned}$$

We simply take expansions of the p-th power above, and sum for all first integers. We have:

$$\begin{aligned} \mathbb{E}A_n &\leqslant \mathbb{E}A_{n-1} + b \leqslant a + nb, \\ \mathbb{E}A_n^2 &\leqslant \mathbb{E}(A_{n-1}^2 + b^2 + 2bA_{n-1} + M_n^2) \leqslant \mathbb{E}A_{n-1}^2 + 2\mathbb{E}A_{n-1}b + b^2 + 16b\mathbb{E}A_{n-1} \\ &\leqslant a^2 + 18b\Big[\sum_{k=0}^{n-1} a + kb\Big] + b^2n \leqslant a^2 + 18b[na + \frac{n^2}{2}b] + b^2n \\ &\quad \text{using the result about } \mathbb{E}A_{n-1}, \\ &= a^2 + 18bna + b^2(n + 9n^2) \\ &\leqslant (a + 9nb)^2. \end{aligned}$$

We may now pursue for the third order moments:

$$\begin{aligned} \mathbb{E}A_n^3 &\leqslant \mathbb{E}(A_{n-1} + b)^3 + 3\mathbb{E}(A_{n-1} + b)^2M_n^2 + 3\mathbb{E}(A_{n-1} + b)^3M_n + \mathbb{E}M_{n-1}^3 \\ &\leqslant \mathbb{E}(A_{n-1} + b)^3 + 3\mathbb{E}(A_{n-1} + b)^216bA_{n-1} + 0 + 64b^{3/2}\mathbb{E}A_{n-1}^{3/2} \\ &\leqslant (\mathbb{E}A_{n-1}^3 + 3\mathbb{E}A_{n-1}^2b + 3\mathbb{E}A_{n-1}b^2 + b^3) + 3\mathbb{E}(A_{n-1} + b)16bA_{n-1} + 64b^{3/2}\mathbb{E}A_{n-1}^{3/2} \\ &= (\mathbb{E}A_{n-1}^3 + 3\mathbb{E}A_{n-1}^2b + 3\mathbb{E}A_{n-1}b^2 + b^3) + 3\mathbb{E}(A_{n-1} + b)16bA_{n-1} \\ &\qquad + 32b\mathbb{E}A_{n-1}[2b^{1/2}A_{n-1}^{1/2}]. \end{aligned}$$

By expanding, we get

$$\begin{aligned} \mathbb{E}A_n^3 &\leqslant (\mathbb{E}A_{n-1}^3 + 3\mathbb{E}A_{n-1}^2b + 3\mathbb{E}A_{n-1}b^2 + b^3) + 3\mathbb{E}(A_{n-1} + b)16bA_{n-1} \\ &\qquad + 32\mathbb{E}bA_{n-1}[\frac{A_{n-1}}{4} + 4b] \\ &= EA_{n-1}^3 + \mathbb{E}A_{n-1}^2b[3 + 48 + 8] + \mathbb{E}A_{n-1}b^2[3 + 48 + 128] + b^3 \\ &= \mathbb{E}A_{n-1}^3 + 59\mathbb{E}A_{n-1}^2b + 179\mathbb{E}A_{n-1}b^2 + b^3 \\ &\leqslant a^3 + 59b\Big[\sum_{k=1}^{n-1} a^2 + 18bka + b^2(k + 9k^2)\Big] + 179b^2\Big[\sum_{k=1}^{n-1} a + kb\Big] + nb^3 \\ &\leqslant a^3 + 59b[na^2 + 9bn^2a + b^2(n^2/2 + 3n^3)] + 179b^2[na + bn^2/2] + nb^3 \\ &= a^3 + 59nba^2 + b^2a[59 \cdot 9n^2 + 179n] + b^3[59/2 \cdot n^2 + 3 \cdot 59n^3 + 179/2 \cdot n^2 + n] \\ &= a^3 + 59nba^2 + b^2a[531n^2 + 179n] + b^3[119n^2 + 177n^3 + n] \\ &\leqslant (a + 20nb)^3. \end{aligned}$$

We then obtain:

$$\begin{aligned}\mathbb{E}\Big[2\gamma n\big[f(\bar\theta_n)-f(\theta^*)\big]+\|\theta_n-\theta_*\|^2\Big]^2 &\leqslant \big(\|\theta_0-\theta_*\|^2+9n\gamma^2R^2\big)^2\\ \mathbb{E}\Big[2\gamma n\big[f(\bar\theta_n)-f(\theta^*)\big]+\|\theta_n-\theta_*\|^2\Big]^3 &\leqslant \big(\|\theta_0-\theta_*\|^2+20n\gamma^2R^2\big)^3.\end{aligned}$$

Appendix F. Proof of Proposition 10

The proof follows from applying self-concordance properties (Lemma 9) to $\bar\theta_n$. We thus need to provide a control on the probability that $\|f'(\bar\theta_n)\| \geqslant \frac{3\mu}{4R}$.

F.1 Tail Bound for $\|f'(\bar\theta_n)\|$

We derive a large deviation bound, as a consequence of the bound on all moments of $\|f'(\bar\theta_n)\|$ (Proposition 7) and Lemma 12, that allows to go from moments to tail bounds:

$$\mathbb{P}\left(\big\|f'(\bar\theta_n)\big\| \geqslant \frac{2R}{\sqrt{n}}\left[10\sqrt{t}+40R^2\gamma t\sqrt{n}+\frac{3}{\gamma\sqrt{n}}\|\theta_0-\theta_*\|^2+\frac{3}{\gamma R\sqrt{n}}\|\theta_0-\theta_*\|\right]\right)\leqslant 4\exp(-t).$$

In order to derive the bound above, we need to assume that $p \leqslant n/4$ (so that $4p/n \leqslant 2\sqrt{p}/\sqrt{n}$), and thus, when applying Lemma 12, the bound above is valid as long as $t \leqslant n/4$. It is however valid for all t, because the gradients are bounded by R, and for $t > n$, we have $\frac{2R}{\sqrt{n}}10\sqrt{t} \geqslant R$, and the inequality is satisfied with zero probability.

F.2 Bounding the Function Values

From Lemma 9, if $\|f'(\bar\theta_n)\| \geqslant \frac{3\mu}{4R}$, then $f(\bar\theta_n)-f(\theta_*) \leqslant 2\frac{\|f'(\bar\theta_n)\|^2}{\mu}$. This will allow us to derive a tail bound for $f(\bar\theta_n)-f(\theta_*)$, for sufficiently small deviations. For larger deviations, we will use the tail bound which does not use strong convexity (Proposition 5).

We consider the event

$$A_t \;=\; \left\{\big\|f'(\bar\theta_n)\big\| \leqslant \frac{2R}{\sqrt{n}}\left[10\sqrt{t}+40R^2\gamma t\sqrt{n}+\frac{3}{\gamma\sqrt{n}}\|\theta_0-\theta_*\|^2+\frac{3}{\gamma R\sqrt{n}}\|\theta_0-\theta_*\|\right]\right\}.$$

We make the following two assumptions regarding γ and t:

$$\begin{aligned}10\sqrt{t}+40R^2\gamma t\sqrt{n} &\leqslant \frac{2}{3}\frac{3\mu}{4R}\frac{\sqrt{n}}{2R}=\frac{\mu\sqrt{n}}{4R^2}\\ \text{and } \frac{3}{\gamma\sqrt{n}}\|\theta_0-\theta_*\|^2+\frac{3}{\gamma R\sqrt{n}}\|\theta_0-\theta_*\| &\leqslant \frac{1}{3}\frac{3\mu}{4R}\frac{\sqrt{n}}{2R}=\frac{\mu\sqrt{n}}{8R^2},\end{aligned} \tag{13}$$

so that the upper-bound on $\|f'(\bar\theta_n)\|$ in the definition of A_t is less than $\frac{3\mu}{4R}$ (so that we can apply Lemma 9). We thus have:

$$\begin{aligned}A_t &\subset \left\{f(\bar\theta_n)-f(\theta_*)\leqslant\frac{8R^2}{\mu n}\left[10\sqrt{t}+40R^2\gamma t\sqrt{n}+\frac{3}{\gamma\sqrt{n}}\|\theta_0-\theta_*\|^2+\frac{2}{\gamma R\sqrt{n}}\|\theta_0-\theta_*\|\right]^2\right\}\\ &\subset \left\{f(\bar\theta_n)-f(\theta_*)\leqslant\frac{8R^2}{\mu n}\Big[10\sqrt{t}+20\square t+\triangle\Big]^2\right\},\end{aligned}$$

with $\square = 2\gamma R^2\sqrt{n}$ and $\triangle = \frac{3}{\gamma\sqrt{n}}\|\theta_0-\theta_*\|^2 + \frac{3}{\gamma R\sqrt{n}}\|\theta_0-\theta_*\|$.

This implies that for all $t \geqslant 0$, such that $10\sqrt{t} + 20\square t \leqslant \frac{\mu\sqrt{n}}{4R^2}$, that is, our assumption in Equation (13), we may apply the tail bound from Appendix F.1 to get:

$$\mathbb{P}\left(f(\bar{\theta}_n) - f(\theta_*) \geqslant \frac{8R^2}{\mu n}\left[10\sqrt{t} + 20\square t + \triangle\right]^2\right) \leqslant 4e^{-t}. \tag{14}$$

Moreover, we have for all $v \geqslant 0$ (from Proposition 5):

$$\mathbb{P}\left(f(\bar{\theta}_n) - f(\theta_*) \geqslant 30\gamma R^2 v + \frac{3\|\theta_0-\theta_*\|^2}{\gamma n}\right) \leqslant 2\exp(-v). \tag{15}$$

We may now use the last two inequalities to bound the expectation $\mathbb{E}[f(\bar{\theta}_n) - f(\theta_*)]$.

We first express the expectation as an integral of the tail bound and split it into three parts:

$$\begin{aligned}
\mathbb{E}\big[f(\bar{\theta}_n) - f(\theta_*)\big] &= \int_0^{+\infty} \mathbb{P}\big[f(\bar{\theta}_n) - f(\theta_*) \geqslant u\big]du \\
&= \int_0^{\triangle^2\frac{8R^2}{\mu n}} \mathbb{P}\big[f(\bar{\theta}_n) - f(\theta_*) \geqslant u\big]du \\
&\quad + \int_{\triangle^2\frac{8R^2}{\mu n}}^{\frac{8R^2}{\mu n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2} \mathbb{P}\big[f(\bar{\theta}_n) - f(\theta_*) \geqslant u\big]du \\
&\quad + \int_{\frac{8R^2}{\mu n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2}^{+\infty} \mathbb{P}\big[f(\bar{\theta}_n) - f(\theta_*) \geqslant u\big]du.
\end{aligned} \tag{16}$$

We may now bound the three terms separately. For the first integral, we bound the probability by one to get $\displaystyle\int_0^{\triangle^2\frac{8R^2}{\mu n}} \mathbb{P}\big[f(\bar{\theta}_n) - f(\theta_*) \geqslant u\big]du \leqslant \triangle^2\frac{8R^2}{n\mu}$.

For the third term in Equation (16), we use the tail bound in Equation (15) to get

$$\begin{aligned}
&\int_{\frac{8R^2}{\mu n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2}^{+\infty} \mathbb{P}\big[f(\bar{\theta}_n) - f(\theta_*) \geqslant u\big]du \\
=\ &\int_{\frac{8R^2}{\mu n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2 - \frac{3}{\gamma n}\|\theta_0-\theta_*\|^2}^{+\infty} \mathbb{P}\left[f(\bar{\theta}_n) - f(\theta_*) \geqslant u + \frac{3}{\gamma n}\|\theta_0-\theta_*\|^2\right]du \\
\leqslant\ &2\int_{\frac{8R^2}{\mu n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2 - \frac{3}{\gamma n}\|\theta_0-\theta_*\|^2}^{+\infty} \exp\big(-\frac{u}{30\gamma R^2}\big)du.
\end{aligned}$$

We may apply Equation (15) because

$$\frac{8R^2}{\mu n}\big(\frac{\mu\sqrt{n}}{4R^2}+\triangle\big)^2 - \frac{3}{\gamma n}\|\theta_0-\theta_*\|^2 \geqslant \frac{8R^2}{\mu n}\big(\frac{\mu\sqrt{n}}{4R^2}+\triangle\big)^2 - \frac{\mu}{8R^2} \geqslant \frac{8R^2}{\mu n}\big(\frac{\mu\sqrt{n}}{4R^2}\big)^2 - \frac{\mu}{8R^2} = \frac{3\mu}{8R^2} \geqslant 0.$$

We can now compute the bound explicitly to get

$$\begin{aligned}
&\int_{\frac{8R^2}{\mu n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2}^{+\infty} \mathbb{P}\big[f(\bar{\theta}_n)-f(\theta_*)\geqslant u\big]du \\
&\leqslant 60\gamma R^2\exp\left(\frac{-1}{30\gamma R^2}\left[\frac{8R^2}{\mu n}\big(\frac{\mu\sqrt{n}}{4R^2}+\triangle\big)^2-\frac{3}{\gamma n}\|\theta_0-\theta_*\|^2\right]\right)\leqslant 60\gamma R^2\exp\left(\frac{-1}{30\gamma R^2}\frac{3\mu}{8R^2}\right) \\
&\leqslant 60\gamma R^2\exp\left(-\frac{\mu}{80\gamma R^4}\right)\leqslant 60\gamma R^2\frac{80\gamma R^4}{2\mu}\text{ using } e^{-\alpha}\leqslant\frac{1}{2\alpha}\text{ for all }\alpha>0 \\
&= \frac{2400\gamma^2R^6}{\mu}.
\end{aligned}$$

We now consider the second term in Equation (16) for which we will use Equation (14). We consider the change of variable $u=\frac{8R^2}{\mu n}\Big[10\sqrt{t}+20\square t+\triangle\Big]^2$, for which $u\in\Big[\triangle^2\frac{8R^2}{\mu n},\frac{8R^2}{\mu n}\big(\frac{\mu\sqrt{n}}{4R^2}+\triangle\big)^2\Big]$ implies $t\in[0,+\infty)$. This implies that

$$\begin{aligned}
&\int_{\triangle^2\frac{8R^2}{\mu n}}^{\frac{8R^2}{\mu n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2}\mathbb{P}\big[f(\bar{\theta}_n)-f(\theta_*)\geqslant u\big]du \\
\leqslant\ & \int_0^\infty 4e^{-t}d\left(\frac{8R^2}{\mu n}\left[10\sqrt{t}+20\square t+\triangle\right]^2\right) \\
=\ & \frac{32R^2}{\mu n}\int_0^\infty e^{-t}\left(100+400\square^2 2t+400\square\frac{3}{2}t^{1/2}+20\triangle\frac{1}{2}t^{-1/2}+40\triangle\square\right)dt \\
=\ & \frac{32R^2}{\mu n}\left(100\Gamma(1)+400\square^2 2\Gamma(2)+400\square\frac{3}{2}\Gamma(3/2)+20\triangle\frac{1}{2}\Gamma(1/2)+40\triangle\square\Gamma(1)\right) \\
& \text{with }\Gamma\text{ denoting the Gamma function,} \\
=\ & \frac{32R^2}{\mu n}\left(100+400\square^2 2+400\square\frac{3}{2}\frac{1}{2}\sqrt{\pi}+20\triangle\frac{1}{2}\sqrt{\pi}+40\triangle\square\right).
\end{aligned}$$

We may now combine the three bounds to get, from Equation (16),

$$\begin{aligned}
\mathbb{E}\big[f(\bar{\theta}_n)-f(\theta_*)\big]\ &\leqslant\ \triangle^2\frac{8R^2}{n\mu}+\frac{2400\gamma^2R^6}{\mu} \\
&\quad+\frac{32R^2}{\mu n}\left(100+400\square^2 2+400\square\frac{3}{2}\frac{1}{2}\sqrt{\pi}+20\triangle\frac{1}{2}\sqrt{\pi}+40\triangle\square\right) \\
&\leqslant\ \frac{32R^2}{n\mu}\left[\frac{\triangle^2}{4}+75\gamma^2R^4n+100+800\square^2+300\square\sqrt{\pi}+10\triangle\sqrt{\pi}+40\triangle\square\right].
\end{aligned}$$

For $\gamma=\frac{1}{2R^2\sqrt{N}}$, with $\alpha=R\|\theta_0-\theta_*\|$, $\square=1$ and $\triangle=6\alpha^2+6\alpha$, we obtain

$$\begin{aligned}
\mathbb{E}\big[f(\bar{\theta}_N)-f(\theta_*)\big]&\leqslant\frac{32R^2}{N\mu}\left[\frac{1}{4}\triangle^2+1451+58\Delta\right] \\
&\leqslant\frac{32R^2}{N\mu}\left[9\alpha^4+18\alpha^3+9\alpha^2+1451+348\alpha^2+348\alpha\right] \\
&\leqslant\frac{R^2}{N\mu}\big(625\alpha^4+7500\alpha^3+33750\alpha^2+67500\alpha+50625\big)=\frac{R^2}{N\mu}\big(5\alpha+15\big)^4.
\end{aligned}$$

Note that the previous bound is only valid if $\frac{3}{\gamma\sqrt{n}}\|\theta_0-\theta_*\|^2+\frac{3}{\gamma R\sqrt{n}}\|\theta_0-\theta_*\|\leqslant\frac{\mu\sqrt{n}}{8R^2}$, that is, under the condition $6R^2\|\theta_0-\theta_*\|^2+6R\|\theta_0-\theta_*\|\leqslant\frac{\mu\sqrt{N}}{8R^2}$. If the condition is not satisfied, then the bound is still valid because of Lemma 1. We thus obtain the desired result.

F.3 Bound on Iterates

Following the same principle as for function values in Appendix F.2, we consider the same event A_t. With the same condition on γ and t, we have:

$$A_t\subset\left\{\|\bar{\theta}_n-\theta_*\|^2\leqslant\frac{16R^2}{\mu^2n}\left[10\sqrt{t}+20\square t+\triangle\right]^2\right\},$$

which leads to the tail bound:

$$\mathbb{P}\left(\|\bar{\theta}_n-\theta_*\|^2\geqslant\frac{16R^2}{\mu^2n}\left[10\sqrt{t}+20\square t+\triangle\right]^2\right)\leqslant 4e^{-t}.$$

We may now split the expectation in three integrals:

$$\begin{aligned}\mathbb{E}\|\bar{\theta}_n-\theta_*\|^2 &= \int_0^{\frac{16R^2}{\mu^2n}\triangle^2}\mathbb{P}\big[\|\bar{\theta}_n-\theta_*\|^2\geqslant u\big]du \\ &\quad+\int_{\frac{16R^2}{\mu^2n}\triangle^2}^{\frac{16R^2}{\mu^2n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2}\mathbb{P}\big[\|\bar{\theta}_n-\theta_*\|^2\geqslant u\big]du \\ &\quad+\int_{\frac{16R^2}{\mu^2n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2}^{\infty}\mathbb{P}\big[\|\bar{\theta}_n-\theta_*\|^2\geqslant u\big]du.\end{aligned} \tag{17}$$

The first term in Equation (17) is simply bounded by bounding the tail bound by one (like in the previous section): $\int_0^{\frac{16R^2}{\mu^2n}\triangle^2}\mathbb{P}\big[\|\bar{\theta}_n-\theta_*\|^2\geqslant u\big]du\leqslant\frac{16R^2}{\mu^2n}\triangle^2$. The last integral in Equation (17) may be bounded as follows:

$$\begin{aligned}&\int_{\frac{16R^2}{\mu^2n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2}^{\infty}\mathbb{P}\big[\|\bar{\theta}_n-\theta_*\|^2\geqslant u\big]du \\ =\;&\mathbb{E}\left[1_{\|\bar{\theta}_n-\theta_*\|^2\geqslant\frac{16R^2}{\mu^2n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2}\|\bar{\theta}_n-\theta_*\|^2\right] \\ \leqslant\;&\mathbb{P}\left[\|\bar{\theta}_n-\theta_*\|^2\geqslant\frac{16R^2}{\mu^2n}\big(\frac{\mu\sqrt{n}}{4R^2}+\triangle\big)^2\right]^{1/2}\left[\mathbb{E}\big(\|\bar{\theta}_n-\theta_*\|^4\big)\right]^{1/2} \\ &\text{using Cauchy-Schwarz inequality,} \\ \leqslant\;&\mathbb{P}\left[\|\bar{\theta}_n-\theta_*\|^2\geqslant\frac{16R^2}{\mu^2n}\big(\frac{\mu\sqrt{n}}{4R^2}+\triangle\big)^2\right]^{1/2}\left(\|\theta_0-\theta_*\|^2+9\gamma^2nR^2\right)\text{ using Proposition 3.}\end{aligned}$$

Moreover, if we denote by t_0 the largest solution of $10\sqrt{t_0} + 20\square t_0 = \frac{\mu\sqrt{n}}{4R^2}$, we have:

$$\begin{aligned}\sqrt{t_0} &= \frac{-10 + \sqrt{100 + 20\square\frac{\mu\sqrt{n}}{R}}}{40\square} = \frac{-10 + 10\sqrt{1 + 20\square\frac{\mu\sqrt{n}}{100R}}}{40\square} \\ &\geqslant \frac{9}{40\square}\sqrt{20\square\frac{\mu\sqrt{n}}{100R}},\end{aligned}$$

as soon as $20\square\frac{\mu\sqrt{n}}{100R} \geqslant 100$, since if $q \geqslant 100$, $-1 + \sqrt{1+q} \leqslant \frac{9}{10}\sqrt{q}$. This implies that

$$\begin{aligned}&\int_{\frac{16R^2}{\mu^2 n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2}^{\infty} \mathbb{P}\big[\|\bar{\theta}_n - \theta_*\|^2 \geqslant u\big] du \\ \leqslant\ & \left[4\exp(-t_0)\right]^{1/2}\left(\|\theta_0 - \theta_*\|^2 + 9\gamma^2 nR^2\right) \\ \leqslant\ & \frac{9}{2t_0^2}\left(\|\theta_0 - \theta_*\|^2 + 9\gamma^2 nR^2\right) \text{ using } \exp(-\alpha) \leqslant \frac{9}{16\alpha^2} \text{ for all } \alpha > 0, \\ \leqslant\ & \frac{9}{2}\frac{40^4\square^4 100^2 R^4}{9^4 20^2\square^2\mu^2 n}\left[\frac{9}{4}\square^2/R^2 + \frac{\gamma\sqrt{n}}{3}\triangle\right] \\ \leqslant\ & 686 \times 64\frac{\square^2 R^2}{\mu^2 n}\left[\frac{9}{4}\square^2 + \frac{1}{6}\square\triangle\right].\end{aligned}$$

The second term in Equation (17) is bounded exactly like in Appendix F.2, leading to:

$$\begin{aligned}&\int_{\triangle^2\frac{16R^2}{\mu^2 n}}^{\frac{16R^2}{\mu^2 n}\left(\frac{\mu\sqrt{n}}{4R^2}+\triangle\right)^2} \mathbb{P}\big[\|\bar{\theta}_n - \theta_*\|^2 \geqslant u\big] du \\ \leqslant\ & \int_0^\infty 4e^{-t} d\left(\frac{16R^2}{\mu^2 n}\left[10\sqrt{t} + 20\square t + \triangle\right]^2\right) \\ \leqslant\ & \frac{64R^2}{\mu^2 n}\int_0^\infty e^{-t}\left(100 + 400\square^2 2t + 400\square\frac{3}{2}t^{1/2} + 20\triangle\frac{1}{2}t^{-1/2} + 40\triangle\square\right)dt \\ \leqslant\ & \frac{64R^2}{\mu^2 n}\left(100\Gamma(1) + 400\square^2 2\Gamma(2) + 400\square\frac{3}{2}\Gamma(3/2) + 20\triangle\frac{1}{2}\Gamma(1/2) + 40\triangle\square\Gamma(1)\right)dt \\ \leqslant\ & \frac{64R^2}{\mu^2 n}\left(100 + 400\square^2 2 + 400\square\frac{3}{2}\frac{1}{2}\sqrt{\pi} + 20\triangle\frac{1}{2}\sqrt{\pi} + 40\triangle\square\right).\end{aligned}$$

We can now put all elements together to obtain, from Equation (17):

$$\begin{aligned}&\mathbb{E}\|\bar{\theta}_n - \theta_*\|^2 \\ \leqslant\ & \frac{64R^2}{\mu^2 n}\left(100 + 400\square^2 2 + 400\square\frac{3}{2}\frac{1}{2}\sqrt{\pi} + 20\triangle\frac{1}{2}\sqrt{\pi} + 40\triangle\square\right) \\ &\qquad + \frac{16R^2}{\mu^2 n}\triangle^2 + 686 \times 64\frac{\square^2 R^2}{\mu^2 n}\left[\frac{9}{4}\square^2 + \frac{1}{6}\square\triangle\right] \\ \leqslant\ & \frac{64R^2}{n\mu^2}\left[\frac{1}{4}\triangle^2 + 100 + 800\square^2 + 532\square + 32\triangle + 40\triangle\square + 686\frac{9}{4}\square^4 + 686\frac{\triangle\square^3}{6}\right].\end{aligned}$$

For $\gamma = \frac{1}{2R^2\sqrt{N}}$, with $\alpha = R\|\theta_0 - \theta_*\|$, $\square = 1$ and $\triangle = 6\alpha^2 + 6\alpha$, we get

$$\begin{aligned}
\mathbb{E}\|\bar{\theta}_N - \theta_*\|^2 &\leqslant \frac{8R^2}{N\mu^2}\Big[2\triangle^2 + 8\triangle(32+40+115) + 8(100+800+532+1544)\Big] \\
&\leqslant \frac{8R^2}{N\mu^2}\Big[2\triangle^2 + 1496\triangle + 23808\Big] \\
&\leqslant \frac{8R^2}{N\mu^2}\Big[72\alpha^4 + 144\alpha^3 + 72\alpha^2 + 1496\times 6\alpha^2 + 1496\times 6\alpha + 23808\Big] \\
&\leqslant \frac{R^2}{N\mu^2}\Big[1296\alpha^4 + 18144\alpha^3 + 95256\alpha^2 + 222264\alpha + 194481\Big] = \frac{R^2}{N\mu^2}\big(6\alpha+21\big)^4.
\end{aligned}$$

The previous bound is valid as long as $\frac{\mu\sqrt{N}}{R} \geqslant \frac{10000}{20} = 500$. If it is not satisfied, then Lemma 1 shows that it is still valid.

References

A. Agarwal, P. L. Bartlett, P. Ravikumar, and M. J. Wainwright. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *IEEE Transactions on Information Theory*, 58(5):3235–3249, 2012.

F. Bach. Self-concordant analysis for logistic regression. *Electronic Journal of Statistics*, 4: 384–414, 2010.

F. Bach and E. Moulines. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

F. Bach and E. Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate $O(1/n)$. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

L. Bottou and Y. Le Cun. On-line learning for very large data sets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005.

S. Boucheron, G. Lugosi, and P. Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

M. N. Broadie, D. M. Cicek, and A. Zeevi. General bounds and finite-time improvement for stochastic approximation algorithms. Technical report, Columbia University, 2009.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

V. Fabian. On asymptotic normality in stochastic approximation. *The Annals of Mathematical Statistics*, 39(4):1327–1332, 1968.

D. A. Freedman. On tail probabilities for martingales. *Annals of Probability*, 3(1):100–118, 1975.

E. Hazan and S. Kale. Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization. In *Proceedings of the Conference on Learning Theory (COLT)*, 2001.

A. Juditsky and Y. Nesterov. Primal-dual subgradient methods for minimizing uniformly convex functions. Technical Report 00508933, HAL, 2010.

S. M. Kakade and A. Tewari. On the generalization ability of online strongly convex programming algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer-Verlag, second edition, 2003.

S. Lacoste-Julien, M. Schmidt, and F. Bach. A simpler approach to obtaining an $O(1/t)$ convergence rate for projected stochastic subgradient descent. Technical Report 1212.2002, ArXiv, 2012.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

G. Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1-2):365–397, 2012.

N. Le Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for strongly-convex optimization with finite training sets. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

H. B. McMahan and M. Streeter. Open problem: Better bounds for online logistic regression. In *COLT/ICML Joint Open Problem Session*, 2012.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

A. S. Nemirovski and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley & Sons, 1983.

Y. Nesterov. *Introductory Lectures on Convex Optimization: a Basic Course*. Kluwer Academic Publishers, 2004.

Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM studies in Applied Mathematics, 1994.

Y. Nesterov and J. P. Vial. Confidence level solutions for stochastic programming. *Automatica*, 44(6):1559–1568, 2008.

I. Pinelis. Optimum bounds for the distributions of martingales in banach spaces. *The Annals of Probability*, 22(4):1679–1706, 1994.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

D. Ruppert. Efficient estimations from a slowly convergent Robbins-Monro process. Technical Report 781, Cornell University Operations Research and Industrial Engineering, 1988.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2001.

S. Shalev-Shwartz and N. Srebro. SVM optimization: inverse dependence on training set size. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2007.

S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan. Stochastic convex optimization. In *Proceedings of the Conference on Learning Theory (COLT)*, 2009.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

A. W. Van der Vaart. *Asymptotic Statistics*. Cambridge Univ. Press, 1998.

Z. Wang, K. Crammer, and S. Vucetic. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training. *Journal of Machine Learning Research*, 13:3103–3131, 2012.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 9:2543–2596, 2010.

Journal of Machine Learning Research 15 (2014) 629-654 Submitted 3/13; Revised 9/13; Published 2/14

Random Intersection Trees

Rajen Dinesh Shah RDS37@CAM.AC.UK
Statistical Laboratory
University of Cambridge
Cambridge, CB3 0WB, UK

Nicolai Meinshausen MEINSHAUSEN@STAT.MATH.ETHZ.CH
Seminar für Statistik
ETH Zürich
8092 Zürich, Switzerland

Editor: John Lafferty

Abstract

Finding interactions between variables in large and high-dimensional data sets is often a serious computational challenge. Most approaches build up interaction sets incrementally, adding variables in a greedy fashion. The drawback is that potentially informative high-order interactions may be overlooked. Here, we propose an alternative approach for classification problems with binary predictor variables, called *Random Intersection Trees*. It works by starting with a maximal interaction that includes all variables, and then gradually removing variables if they fail to appear in randomly chosen observations of a class of interest. We show that informative interactions are retained with high probability, and the computational complexity of our procedure is of order p^κ, where p is the number of predictor variables. The value of κ can reach values as low as 1 for very sparse data; in many more general settings, it will still beat the exponent s obtained when using a brute force search constrained to order s interactions. In addition, by using some new ideas based on min-wise hash schemes, we are able to further reduce the computational cost. Interactions found by our algorithm can be used for predictive modelling in various forms, but they are also often of interest in their own right as useful characterisations of what distinguishes a certain class from others.

Keywords: high-dimensional classification, interactions, min-wise hashing, sparse data

1. Introduction

In this paper, we consider classification with high-dimensional binary predictors. We suppose we have data that can be written in the form (Y_i, X_i) for observations $i = 1, \ldots, n$; Y_i is the class label and $X_i \subseteq \{1, \ldots, p\}$ is the set of active predictors for observations i (out of a total of p predictors). An important example of this type of problem is that of text classification, where then X_i is the set of frequently appearing words (in a suitable sense) for document i, and Y_i indicates whether the document belongs to a certain class. In this case, the dimension p can be of the order of several thousand or more. More generally, if data with continuous predictors are available, they can be converted to binary format by choosing various split-points, and then reporting whether or not each variable exceeds each of these thresholds.

Our aim here is to develop methodology that can discover important interaction terms in the data without requiring that any of their lower order interactions are also informative. More precisely, we are interested in finding subsets $S \subseteq \{1, \ldots, p\}$ of all predictor variables that occur more often for observations in a class of interest than for other observations. We will use the terms "leaf nodes", "rules", "patterns" and "interactions" interchangeably to describe such subsets S. For simplicity, suppose there are only two classes, the set of labels being $\{0, 1\}$. The case with more than two classes can be dealt with using one-versus-one, or one-versus-all strategies. Given a pair of thresholds, $0 \leq \theta_0 < \theta_1 \leq 1$, our goal is to find all sets S (or as many as possible), for which

$$\mathbb{P}_n(S \subseteq X | Y = 1) \geq \theta_1 \quad \text{and} \quad \mathbb{P}_n(S \subseteq X | Y = 0) \leq \theta_0. \tag{1}$$

Here and throughout the paper, we use the subscript n to indicate that the probabilities are empirical probabilities. For example, for $c \in \{0, 1\}$,

$$\mathbb{P}_n(S \subseteq X | Y = c) := \frac{1}{|C_c|} \sum_{i \in C_c} \mathbb{1}_{\{S \subseteq X_i\}},$$

where we have denoted the set of observations in class c by C_c. Of course, one would also be interested in sets S which satisfy a version of (1) with classes 1 and 0 interchanged, but we will only consider (1) for simplicity.

The interaction terms uncovered can be used in various ways. For example, they can be built into tree-based methods, or form new features in linear or logistic regression models. The interactions may also be of interest in their own right, as they can characterise distinctions between classes in a simple and interpretable way. These potentially high-order interactions that our method aims to target would be very difficult to discover using existing methods, as we now explain.

A pure brute force search examines each potential interaction S of a given size to check whether it fulfills (1). Restricting the order of interactions to size s, the computational complexity scales as p^s, rendering problems with even moderate values of p infeasible.

Instead of searching through every possible interaction, tree-based methods build up interactions incrementally. A typical tree classifier such as *CART* (Breiman et al., 1984) works by building a decision tree greedily from root node to the leaves; see also Loh and Shih (1997). The feature space is recursively partitioned based on the variable whose presence or absence best distinguishes the classes. The myopic nature of this strategy makes it a computationally feasible approach, even for very large problems. The downside is that it produces rather unstable results: small changes in the data can lead to very different partitions being produced at the leaf nodes. Moreover, because of the incremental way in which interactions are constructed, the success of this strategy in recovering an important interaction S rests on at least some of its lower order interactions being informative for distinguishing the classes.

Approaches based on tree ensembles can somewhat alleviate the problem of tree instability; *Random Forests* (Breiman, 2001) is a prominent example. Here the data with which the decision trees are constructed is sampled with replacement from the original data. Further randomness is introduced by randomising over the subset of variables considered for each split in the construction of the trees. While the results of *Random Forests* are very

complex and hard to interpret, one can examine what are known as variable importance measures. These aim to quantify the marginal or pairwise importance of predictor variables (Strobl et al., 2008). Though such measures can be useful, checking through all possible high-order interactions is too cumbersome, and so these may fail to be highlighted.

More recently, there has been interest in algorithms that start from deep splits or leaf nodes in trees and then try to build a simpler model out of many thousands of these leaves by regularisation and dimension reduction. Examples include *Rule Ensembles* (Friedman and Popescu, 2008), *Node Harvest* (Meinshausen, 2010) and the general framework of *Decision Lists* (Marchand and Sokolova, 2006; Rivest, 1987). Though these methods have been demonstrated to improve on *Random Forests* in some situations, they nevertheless crucially rely on a good initial basis of leaf nodes. These bases are usually generated by tree ensemble methods and so, if the base trees miss some important splits, they would also be absent in the results of these derivative algorithms.

A complementary approach has developed in data mining under the name of frequent itemset search, starting with the *Apriori* algorithm (Agrawal et al., 1994), which has since then developed into many improved and more specialised forms. The starting point for these was "market basket analysis", where the shopping behaviour of customers is analysed and the goal is to identify items that are often bought together. Many algorithms have been proposed that aim to improve on *Apriori* in terms of memory requirements and speed, such as the *FP-growth* (Han et al., 2000) and *H-mine* (Pei et al., 2001) algorithms. While generally very successful, all these methods are only computationally feasible in large-scale settings if among the itemsets of low size, there are many that are infrequent, and so using the principle that subsets of frequent itemsets are also frequent, the search space can be greatly reduced. However, if small itemsets all have roughly the same frequency, these methods cannot greatly improve over a brute force search.

We now give a simple example where tree-based approaches and those based on the *Apriori* algorithm will struggle. Let $Z = (Z_1, \ldots, Z_p) \in \{0,1\}^p$ be a random variable with p independent components each having a Bernoulli(1/2)-distribution. We take X to be the set of active entries $\{k : Z_k = 1\}$. Suppose the response $Y \in \{0,1\}$ is determined by an interaction between the first two variables such that $Y = \mathbb{1}_{\{Z_1 + Z_2 \neq 1\}}$. Then none of the variables have a marginal effect as Y is independent of Z_k for all $k = 1, \ldots, p$. In this case, when using trees or the *Apriori* algorithm, one would have to search among $O(p^2)$ potential interactions to find the interaction pattern $\{1, 2\}$.

This paper looks at a new way to discover interactions, which we call *Random Intersection Trees*. Rather than searching through potential interactions directly, our method works by looking for collections of observations whose common active variables together form informative interactions. We present a basic version of the *Random Intersection Trees* algorithm in the following section. This approach allows for computationally feasible discovery of interactions in settings where most existing procedures would perform poorly. Bounds on the complexity of our algorithm are given in Section 3. For example, our results yield that in the scenario discussed in the previous paragraph, the order of computational complexity of our method is at most $o(p^\kappa)$ for any $\kappa > 1$. In Section 4, we propose some modifications of our basic method to reduce its computational cost, based on min-wise hash schemes. Some numerical examples are given in Section 5. We conclude with a brief discussion in Section 6, and all technical proofs are collected in the appendix.

2. Random Intersection Trees

Our method searches for important interactions by looking at intersections of randomly chosen observations from class 1. We start with the full set of variables as an interaction and then iteratively prune away variables to make the interaction smaller. At each iteration, we just keep variables in the interaction that are present in a new randomly chosen observation of class 1. All variables in the interaction that are not present in the chosen observation are removed. Then we repeat with a new randomly chosen observation until an interaction of the desired size emerges. If a pattern S has high prevalence in class 1, that is, $\mathbb{P}_n(X = S|Y = 1)$ is large, it will be included in the observations chosen with high probability. Thus, provided the overall process is repeated often enough, S is likely to be retained in some of the final intersections. On the other hand, elements in S^c, the complement of S in $\{1, \ldots, p\}$, are unlikely to be present in all the observations being intersected. Thus of those intersections which contain S, there is a good chance that at least one of them is exactly S. Arranging the procedure in a tree-type search makes performing the intersections more computationally efficient; details are given in the following section. One would then consider each of these intersections as possible solutions of (1), checking whether their prevalence among class 0 is below θ_0.

It may at first seem strange that in the above, class 0 plays a part in the procedure only at the very end. One might expect that many candidate interactions could be generated that have high prevalence in both classes 1 and 0 and thus would not be useful for distinguishing between classes. In Section 4, we do present an improved version of our algorithm that makes use of class 0 at an earlier stage. However, in the sparse setting we are considering here, interactions with high prevalence in either class would typically be rather few in number. Thus even if all interactions with high prevalence in class 1, and not necessarily low prevalence in class 0, were generated by the procedure outlined above, this would be a manageable number of candidate sets. Note that the assumptions that allow this to happen certainly do not trivialise the problem: even if, given all solutions to the first equation in (1), it is easy to uncover those interactions that additionally satisfy the second equation, the first part of the task is still very challenging.

To describe the details of our algorithm, we first define some terms associated with trees that will be needed later. Recall that a tree is a pair (N, E) of nodes and edges forming a connected acyclic (undirected) graph. We will always assume (with no loss of generality) that $N = \{1, \ldots, |N|\}$. A *rooted tree* is the directed acyclic graph obtained from a tree by designating one node as root and directing all edges away from this root.

Let α and β be two nodes in a rooted tree, with β not the root node. If $(\alpha, \beta) \in E$, β is said to be the *child* of α, and α, the *parent* of β. We will denote by $\mathrm{ch}(\alpha)$, the set of children of a node α. Since we are only considering rooted trees here as opposed to general directed graphs, we will differ with convention slightly and will use $\mathrm{pa}(\beta)$ to mean the unique parent of β. Thus here, $\mathrm{pa}(\beta)$ is a node itself, whereas $\mathrm{ch}(\alpha)$ is a set of nodes.

If $\alpha \neq \beta$ lies on the unique path from the root to β, we say α is an *ancestor* of β, and β is a *descendant* of α. We denote the sets of all ancestors and descendants of α by $\mathrm{an}(\alpha)$ and $\mathrm{de}(\alpha)$ respectively. The *depth* of α, denoted $\mathrm{depth}(\alpha)$, is the number of ancestors of α: $\mathrm{depth}(\alpha) = |\mathrm{an}(\alpha)|$. In particular, the depth of the root node is 0. The *depth* (also known as the *height*) of a rooted tree is the length of the longest path, or equivalently, the greatest

number of ancestors of any particular node. By *level* d of the tree, we will mean the set of nodes with depth d.

We will say an indexing of the nodes is *chronological* if, for every parent and child pair, larger indices are assigned to the child than the parent. In particular, the root node will be 1. Note that both depth-first and breadth-first indexing methods are chronological in this way.

Algorithm 1 A basic version of Random Intersection Trees

for tree $m = 1$ **to** M **do**

Let m be a rooted tree of depth D, with each node j in levels $0, \ldots, D-1$ having B_j children, where the B_j are i.i.d. with a pre-specified distribution. Denote by J the total number of nodes in the tree, and index the nodes chronologically. For each of the nodes $j = 1, \ldots, J$, let $i(j)$ be an independently and uniformly chosen index in the set of class 1 observations $\{i : Y_i = 1\}$.

Set $S_1 = X_{i(1)}$.
for node $j = 2$ **to** J **do**
Set $S_j = X_{i(j)} \cap S_{\mathrm{pa}(j)}$.
end for

Denote the collection of resulting sets from all nodes at depth d, for $d = 1, \ldots, D$, by $L_{d,m} = \{S_j : \text{ depth}(j) = d\}$.
end for
return candidate set of interactions $L_D := \bigcup_{m=1}^{M} L_{D,m}$.

Algorithm 1 describes a basic version of the *Random Intersection Trees* procedure. The reason for allowing random choices of children is for the proof of Theorem 1, where we can randomly choose the number of children to be in $\{b, b+1\}$ for a suitable integer value b. Although we have allowed the number of children of each non-leaf node in the trees to be random, in practice we would take this as a fixed number.

Looking at the innermost for-loop, we see that each node in each tree is associated with a randomly drawn observation from class 1. For every tree, we visit each non-root node in turn, and compute the intersection of the observation assigned to it, and all those assigned to its ancestors. Because of the way the nodes are indexed, parents are always visited before their children, and this intersection can simply be computed as $S_j = X_{i(j)} \cap S_{\mathrm{pa}(j)}$. This is crucial to reducing the computational complexity of the procedure, as we shall see in the next section.

Each of the sets assigned to the leaf nodes of each of the trees yields a collection of potential candidate interactions, L_D. One could then proceed to test these as potential solutions to (1); we present a more efficient approach in Section 4, where we build this testing step into the construction of the trees.

An illustration of this improved algorithm applied to the Tic-Tac-Toe data discussed in Section 5 is given in Figure 1. Observations here correspond to winning endgame positions, coded such that the data is binary. Class labels record which player (black or white) won the game, and the goal is to infer the interactions (corresponding to positions of a few counters) that lead to a win for each player. In this example, the root node contains a randomly

drawn final win-state for black (class 1). This corresponds to S_1 in our algorithm. For each other node j, we draw a new random observation $i(j)$ from all class 1 observations. The randomly chosen additional black-win state $X_{i(j)}$ is shown along the edge from its parent node. The new intersection, S_j, is the intersection of the interaction in the parent node and the new set $X_{i(j)}$; it is shown in the corresponding node. The early stopping added in the improved algorithm also allows it to run until the algorithm has terminated in all nodes. Thus no prior specification of the tree depth will be necessary in practice, as will be shown in Section 4.

3. Computational Complexity

How many trees do we have to compute to have a very high probability of finding an interesting interaction S that fulfills (1)? And what is the required size of these trees? If the interaction is not associated with a main effect, most approaches like trees and association rules would require of order $p^{|S|}$ searches. In this section, we show that in many settings, *Random Intersection Trees* improves on this complexity. We consider a single interaction S of size $s := |S|$, and examine the computational cost for returning S as one of the candidate interactions, with a given probability. We will see that this depends critically on three factors:

- **Prevalence** $\theta_1 := \mathbb{P}_n(S \subseteq X | Y = 1)$ of the interaction pattern. If the pattern S in question appears frequently in class 1, the search is more efficient.
- **Sparsity** $\delta_k := \mathbb{P}_n(k \in X | Y = 1)$ of the predictor variables $k = 1, \ldots, p$. If δ_k is very low for many k (and sparsity of predictors consequently high), computation of the intersections is much cheaper, and so overall computational cost is greatly reduced. Indeed, for a fixed tree m, consider a node j with depth $d < D$. We have that

  $$\mathbb{E}(|S_j|) = \sum_{k=1}^{p} \delta_k^{d+1}.$$

  Thus, for $j' \in \mathrm{ch}(j)$, computation of $S_{j'}$ requires on average at most

  $$O\left(\log(p) \sum_{k=1}^{p} \delta_k^{d+1}\right)$$

  operations. This is because in order to compute the intersection, one can check whether each member of S_j is in $X_{i(j')}$, and each such check is $O(\log(p))$ if the sets X_i are ordered so a binary search can be used. If we compare this to the $O(p)$ computations required to calculate each of the S_j if no tree structure were used, we see that large efficiency gains are possible when $d \geq 1$ if many variables are sparse. For intersections with the root node, the tree structure offers no advantage, and in practice, branching the tree only after level 1 (so the root node has only one child), is more efficient, though this modification does not improve the order of complexity.
- **Independence of** S: Define $\nu := \max_{k \in S^c} \mathbb{P}_n(k \in X | S \subseteq X, Y = 1)$. If ν is low, less computational effort is required to recover S. Note that if, for some $k \in S^c$,

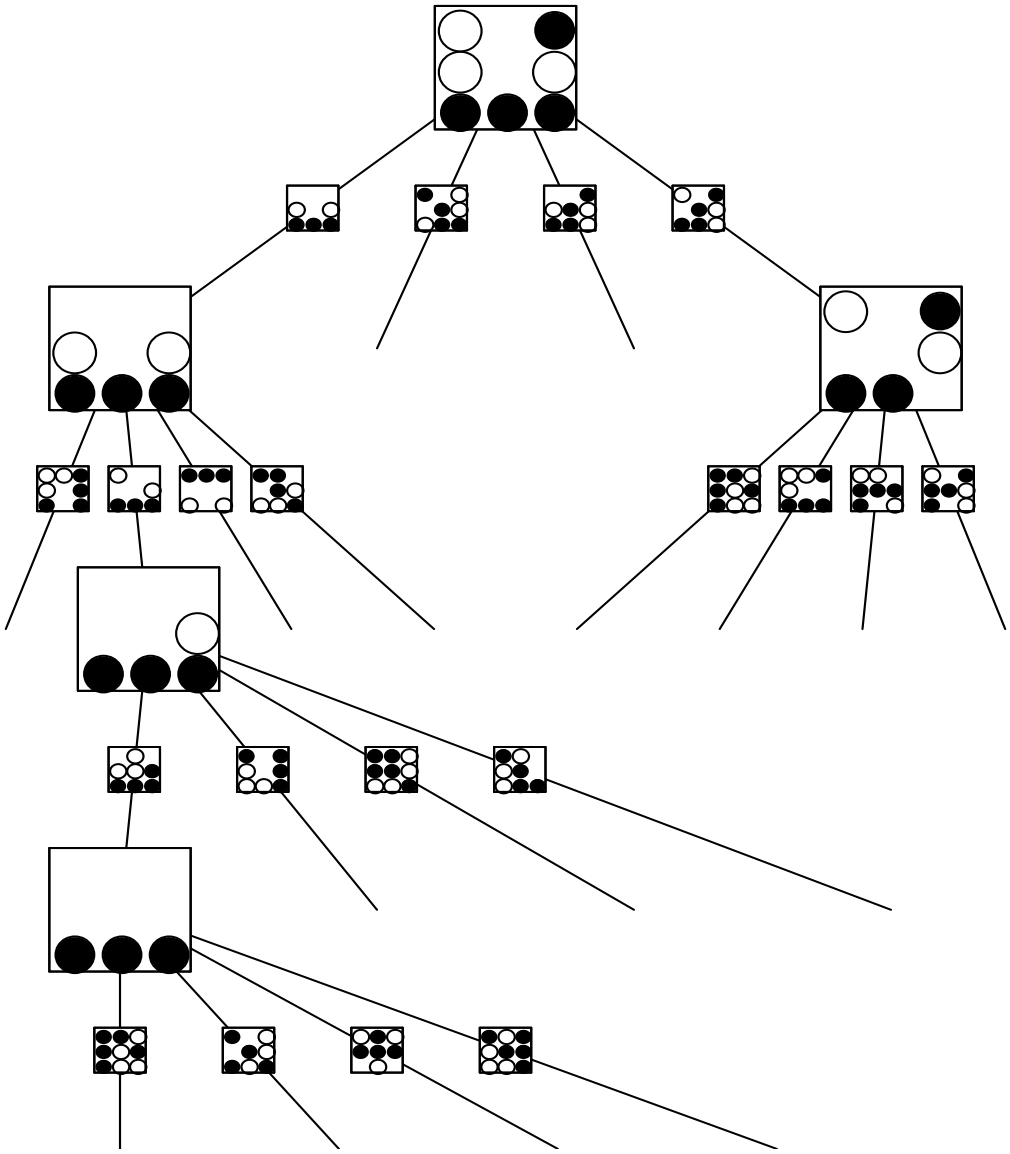


Figure 1: An intersection tree for the Tic-Tac-Toe game data set. Given winning positions of the black player, we intersect them randomly to produce the interactions (corresponding to positions of black or white stones) that are responsible for wins. Starting with a randomly chosen class 1 (black wins) observation at the root node, $B = 4$ randomly chosen class 1 observations are intersected with the pattern. These randomly chosen observations are shown along the edges and the resulting intersections S_j as the nodes in the next layer of the tree. Nodes are only shown if the corresponding patterns S_j have an estimated prevalence among class 0 below a set threshold; the branching of the tree terminates for all other nodes. The algorithm continues until all resulting S_j corresponding to the leaf nodes have prevalence among class 0 exceeding the threshold. Here, one of the winning states for black is filtered out after three intersections.

$\mathbb{P}_n(k \in X | S \subseteq X) = 1$, interest would centre on $S \cup \{k\}$ rather than S itself. Indeed, if S satisfied (1), so would $S \cup \{k\}$. In general, if ν is large, the search will tend to find sets containing S, though not necessarily S itself.

With the assumptions that $\theta_1 > 0$ and $\nu < 1$, we can give a bound on the computational complexity of the basic version of *Random Intersection Trees* introduced in the previous section.

Let us define

$$C(M, D, F_B)$$

to be the expected number of computations required to perform all the intersections in the algorithm when M trees of depth D are created and the distribution of the branching factors B_j is F_B.

Theorem 1 *Given $\eta, \epsilon \in (0, 1]$, there exist choices of M, D and F_B such that the set L_D returned by Algorithm 1 contains S with probability at least $1 - \eta$, and*

$$C(M, D, F_B) = O\left[\log(1/\eta)\frac{\log^2(p)}{\epsilon}\left\{p + \sum_{k:\,(1+\epsilon)\delta_k > \theta_1} p^{\frac{\log\{(1+\epsilon)\delta_k/\theta_1\}}{\log(1/\nu)}}\right\}\right]. \tag{2}$$

As a function of the number of variables p, there is a contribution of $p\log^2(p)$ and an additional contribution in the brackets that depends on the sparsity δ_k of each variable. Sparse variables do not contribute to this sum, which can be $O(1)$ if sparsity among variables is high enough. This would yield a computational complexity with order bounded above by $o(p^\kappa)$ for any $\kappa > 1$, compared to the corresponding complexity of p^s for a brute force search. In most interesting settings, however, we would not achieve a nearly linear scaling in complexity, but would hope to still be faster than a brute force search.

Before discussing the result further, we comment briefly on the values of M, D and the distribution of the B_j, that yield (2). From the proof, it follows that there exist choices of M and D giving (2) that satisfy

$$M \leq \frac{(1+2\epsilon)\log(1/\eta)}{2\epsilon\theta_1},$$
$$D \leq \frac{\log\{p(1+2\epsilon)\}}{\log(1/\nu)}.$$

The random number B_j used in the proof takes just one of two consecutive integers (essentially to avoid the discretisation effect when being restricted to integers), and $\mathbb{E}(B_j) \leq (1+\epsilon)/\theta_1$. Though the optimal choices of parameters for the theorem depend on the unknown ν and the minimising ϵ, which will in turn depend on ν, the functional relationships given above still provide rough qualitative guidelines for good choices for these parameters in practice.

Using the values of M, D and B_j necessary to guarantee that with high probability S is in the set L_D, we can also obtain a bound on the expected number of candidate interaction sets in L_D. This will in turn bound the expected number of "false positives" returned. The

expected number of sets returned is bounded by

$$\mathbb{E}(|L_D|) \leq M\mathbb{E}(B_j)^D \ \leq \frac{\log(1/\eta)}{\epsilon}\left\{\frac{(1+2\epsilon)p}{\nu}\right\}^{\frac{\log(1+\epsilon)/\theta_1}{\log(1/\nu)}}.$$

The value of ϵ can be chosen to minimise the bound above, but its value here and in the computational complexity bound of Theorem 1 have to be the same, as they are linked to the choice of the branching factor used when building the trees. We see that in many situations, we can expect the bound above to be very much lower than the $O(p^s)$ sets a complete list of s-way interactions would contain. Note that if s were known, the relevant quantity to consider would be

$$\mathbb{E}(|\{S' \in L_D : |S'| = s\}|),$$

which is likely to be much less than $\mathbb{E}(|L_D|)$. Even if s were unknown, one would only be interested in the expected number of non-empty sets in L_D, a quantity which may well also be substantially lower than the derived bound on $\mathbb{E}(|L_D|)$.

3.1 The Influence of Sparsity on Computational Complexity

It is interesting to make the influence of the sparsity of individual variables, δ_k, on the overall computational complexity, more explicit. We have the following corollary to Theorem 1.

Corollary 2 *Define β by $\nu = \theta_1^\beta$. Suppose that $\gamma, \alpha^\star, \alpha_\star$ are such that $\alpha^\star > \alpha_\star$, and*

$$\begin{aligned} \delta_k \ &\leq \ \theta_1^{1-\alpha^\star} && \text{for all } k \in \{1,\ldots,p\}, \\ \delta_k \ &> \ \theta_1^{1-\alpha_\star} && \text{for at most } p^\gamma \text{ variables.} \end{aligned}$$

Given $\eta \in (0,1]$, there exist choices of M, D and F_B such that the set L_D returned by Algorithm 1 contains S with probability at least $1-\eta$, and

$$C(M,D,F_B) = o\big(p^\kappa\big) \qquad \text{for any } \kappa > \max\left\{\frac{\alpha^\star}{\beta} + \gamma,\ \left[\frac{\alpha_\star}{\beta}\right]_+ + 1\right\}.$$

The implication of Corollary 2 is most apparent if we take $\gamma = 1$ as we can then set $\alpha_\star = 0$. In this case,

$$\alpha^\star = 1 - \frac{\log(\max_k \delta_k)}{\log(\theta_1)}.$$

We can then bound the computational complexity by

$$o\big(p^\kappa\big) \qquad \text{for any } \kappa > 1 + \frac{\log(1/\theta_1) - \log(1/\max_k \delta_k)}{\log(1/\nu)}.$$

The fraction on the right-hand side is a function of the prevalence of the pattern S, θ_1, the maximum sparsity of the variables, and the maximum sparsity of the variables in S^c, conditional on the presence of S. As long as this fraction is less than 1, the computational complexity is guaranteed to be better than a brute force search with the knowledge that $s = 2$, and the relative advantage grows for larger sizes of the pattern.

3.2 Independent Noise Variables

To gain further insight, we consider the special case where variables in S^c are independent of S (conditional on being in class 1), in the sense that for all $k \in S^c$,

$$\mathbb{P}_n(k \in X | S \subseteq X, Y = 1) = \mathbb{P}_n(k \in X | Y = 1) = \delta_k. \tag{3}$$

Corollary 3 *Assume* (3) *and that* $\delta_k < 1$ *for all* k. *Given* $\eta \in (0, 1]$, *there exist choices of* M, D *and* F_B *such that the set* L_D *returned by Algorithm 1 contains* S *with probability at least* $1 - \eta$, *and*

$$C(M, D, F_B) = o(p^\kappa) \qquad \textit{for any } \kappa > \frac{\log(1/\theta_1)}{\log(1/\max_k \delta_k)}. \tag{4}$$

We see that the computational complexity is approximately linear in p if the prevalence of the pattern S is as high as the prevalence of the least sparse predictor variables. This is the case in the example mentioned in the introduction, where $\theta = \delta_k = 1/2$.

We can also consider the situation where in addition to the independence (3), all variables have the same sparsity δ. If the prevalence θ_1 of S is only as high as that of a random occurrence of two independent predictor variables, we get $\kappa > 2$ and the computational complexity is approximately quadratic in p. In this case, the algorithm would not yield a computational advantage over brute force search if looking for patterns of size 2. This is to be expected since *every* pattern S of size 2 would have the same prevalence in this scenario, and so there is nothing special about a pattern S of size 2 with prevalence δ^2, and in general no hope of beating the complexity p^s of a brute force search. However, the bound in (4) is independent of s. Thus provided the prevalence, θ_1, drops more slowly than the rate δ^s, at which every pattern of size S would occur randomly among independent predictor variables, our results show that *Random Intersection Trees* is still to be preferred over a brute force search.

4. Early Stopping Using Min-Wise Hashing

While Algorithm 1 is computationally attractive, the following observation suggests that further improvements are possible. Suppose that, for a particular tree, we have just computed the intersection S_j corresponding to a node j at depth $d < D$. If

$$\mathbb{P}_n(S_j \subseteq X | Y = 0) > \theta_0,$$

then since for all $j' \in \mathrm{de}(j)$, $S_{j'} \subseteq S_j$, we also have

$$\mathbb{P}_n(S_{j'} \subseteq X | Y = 0) > \theta_0.$$

Thus no intersection sets corresponding to descendants of j have any hope of yielding solutions to (1), and so all further associated computations are wasted.

In view of this, one option would be to compute the quantity $\mathbb{P}_n(S_j \subseteq X | Y = 0)$ at each node j as the algorithm progresses, and if this exceeds the threshold θ_0, not visit any

descendants j' of j for computation of $S_{j'}$. This could be prohibitively costly, though, as it would require a pass over all observations in class 0, for each node of each tree. One could work with a subsample of the observations, but if θ_0 is low, the subsample size may need to be fairly large in order to estimate the probabilities to a sufficient degree of accuracy.

Instead, we propose a fast approximation, using some ideas based on min-wise hashing (Broder et al., 1998; Cohen et al., 2001; Datar and Muthukrishnan, 2002) applied to the columns of the data-matrix. We describe the scheme by leaving aside the conditioning on $Y = 0$, which can be added at the end by restricting to observations in class 0. Consider taking a random permutation σ of all observations $\{1, \ldots, n\}$. Let $h_\sigma(k)$ be the minimal value ι such that variable k is active in observation $\sigma(\iota)$:

$$h_\sigma(k) := \min\{\iota' : k \in X_{\sigma(\iota')}\}.$$

It is well known (Broder et al., 1998) that the probability that $h_\sigma(k)$ and $h_\sigma(k')$ agree for two variables k, k' under a random permutation σ is identical to the Jaccard-index for the two sets $I_k = \{i : k \in X_i\}$ and $I_{k'} = \{i : k' \in X_i\}$, that is

$$\mathbb{P}_\sigma(h_\sigma(k) = h_\sigma(k')) \;=\; \frac{|I_k \cap I_{k'}|}{|I_k \cup I_{k'}|}.$$

Here the subscript σ indicates that the probability is with respect to a random permutation σ of the observations. A min-wise hash scheme is typically used to estimate the Jaccard-index by approximating the probability on the left-hand side of the equation above.

Now,

$$\begin{aligned}\mathbb{P}_n(S \subseteq X) &= \mathbb{P}_n(k \in X \text{ for all } k \in S) \\ &= \mathbb{P}_n(k \in X \text{ for all } k \in S \,|\, \exists\, k' \in S \text{ such that } k' \in X) \\ &\quad \times \mathbb{P}_n(\exists k \in S \text{ such that } k \in X).\end{aligned}$$

Let us denote the first and second terms on the right-hand side by $\pi_1(S)$ and $\pi_2(S)$ respectively. Note that $\pi_1(S)$ is equal to the probability that all variables $k \in S$ have the same min-wise hash value $h_\sigma(k)$:

$$\pi_1(S) = \mathbb{P}_\sigma(\exists\, \iota : h_\sigma(k) = \iota \text{ for all } k \in S). \tag{5}$$

Turning now to $\pi_2(S)$, observe that

$$\mathbb{E}_\sigma(\min_{k \in S} h_\sigma(k)) = \frac{n+1}{\pi_2(S)n + 1}, \tag{6}$$

and so

$$\pi_2(S) = \frac{n+1}{n}\left\{\frac{1}{\mathbb{E}_\sigma(\min_{k \in S} h_\sigma(k))} - \frac{1}{n+1}\right\}. \tag{7}$$

A derivation of (6) is given in the appendix.

Equations (5) and (7) provide the basis for an estimator of $\mathbb{P}_n(S \subseteq X)$. First we generate L random permutations of $\{1, \ldots, n\}$: $\sigma_1, \ldots, \sigma_L$. We then use these to create an $L \times p$ matrix H whose entries are given by

$$H_{lk} = h_{\sigma_l}(k).$$

Now we estimate $\pi_1(S)$ and $\pi_2(S)$ by their respective finite-sample approximations, $\hat{\pi}_1(S)$ and $\hat{\pi}_2(S)$:

$$\hat{\pi}_1(L; S, H) := \tfrac{1}{L} \sum_{l=1}^{L} \mathbb{1}_{\{H_{lk} = H_{lk'} \text{ for all } k,k' \in S\}},$$
$$\hat{\pi}_2(L; S, H) := \frac{n+1}{n} \left\{ \frac{1}{\frac{1}{L}\sum_{l=1}^{L} \min_{k \in S} H_{lk}} - \frac{1}{n+1} \right\}.$$

Finally, we estimate $\mathbb{P}_n(S \subseteq X)$ by

$$\hat{\mathbb{P}}_n(L; S, H) := \hat{\pi}_1(L; S, H) \cdot \hat{\pi}_2(L; S, H). \tag{8}$$

To our knowledge, this use of min-wise hashing techniques, and in particular the estimator $\hat{\pi}_2(L; S, H)$, is new. The estimator enjoys reduced variance compared to that which would be obtained using subsampling, as the following theorem shows.

Theorem 4 *For $\hat{\mathbb{P}}_n(L; S, H)$, $\pi_1(S)$ and $\pi_2(S)$ defined as in (8), (5), and (7) respectively, as $L \to \infty$, we have*

$$\sqrt{L}(\hat{\mathbb{P}}_n(L; S, H) - \mathbb{P}_n(S \subseteq X)) \xrightarrow{d} N(0, \pi_2(S)^2 \pi_1(S)(1 - \pi_1(S)\pi_2(S))(1 + \epsilon(n))), \tag{9}$$

where

$$\epsilon(n) = \frac{1}{n} \frac{n^{-1} - \pi_2^2 - 2\pi_2 n^{-1}}{\pi_2(\pi_2 + 2n^{-1})(1 + n^{-1})} = O(n^{-1}). \tag{10}$$

A derivation is given in the appendix. If we tried to estimate $\pi_1\pi_2$ by evaluating the prevalence of S on a subset of the data of size L, the corresponding estimator multiplied by $\sqrt{L}$ would have variance

$$\pi_2(S)\pi_1(S)(1 - \pi_1(S)\pi_2(S)) + o_n(1),$$

where $o_n(1) \to 0$ as $n \to \infty$. Comparing this variance to the variance of the normal distribution in (9), we see that a factor of $\pi_2(S)$ is gained: matching the accuracy of the min-wise hash scheme with subsampling would require roughly $1/\pi_2(S)$ times as many samples. By using min-wise hashing, choosing $L = 100$ typically delivers a reasonable approximation as long as we just want to resolve values at $\theta_0 = 0.01$ and above.

An improved version of Algorithm 1, building in the ideas discussed above, is given in Algorithm 2 below. Note that $\hat{\mathbb{P}}_n(S_{\mathrm{pa}(j)}, H)$ need only be computed once for every j with the same parent.

Early stopping decreases the computational cost of the algorithm as many nodes in the trees generated may not need to have their associated intersections calculated. In addition, the set of candidate intersections L_D will be smaller but the chance of it containing interesting intersections would not decrease by much. These gains comes at a small price, since the min-wise hash matrix H must be computed, and the computational effort going into this will in turn determine the quality of the approximation in (8). We have previously shown the complexity bounds in the absence of early stopping and thus avoided the difficulty of making this trade-off explicit. We will use the improved version of *Random Intersection*

Algorithm 2 Random Intersection Trees with early stopping

Compute the $L \times p$ min-wise hash matrix H, using only class 0 observations.
for tree $m = 1$ **to** M **do**
  Let m be a rooted tree of depth D, with each node j in levels $0, \ldots, D-1$ having B_j children, where the B_j are i.i.d. with a pre-specified distribution. Denote by J the total number of nodes in the tree, and index the nodes chronologically. For each of the nodes $j = 1, \ldots, J$, let $i(j)$ be an independently and uniformly chosen index in the set of class 1 observations $\{i : Y_i = 1\}$.

  Set $S_1 = X_{i(1)}$.
  for node $j = 2$ **to** J **do**
    if $\hat{\mathbb{P}}_n(S_{\mathrm{pa}(j)}, H) \leq \theta_0$ **then**
      Set $S_j = X_{i(j)} \cap S_{\mathrm{pa}(j)}$.
    end if
  end for

  Denote the collection of resulting sets of all nodes at depth d, for $d = 1, \ldots, D$, by $L_{d,m} = \{S_j : \mathrm{depth}(j) = d\}$.
end for
return $L_D := \bigcup_{m=1}^{M} L_{D,m}$.

Trees with early stopping in all the practical examples to follow, taking small values of L in the range of a (few) hundred permutations.

The depth D of the tree is still given explicitly in Algorithm 2. An interesting modification creates the tree recursively. Starting with the root node, B children are added to all leaf nodes of the tree in which the early stopping criterion has not been triggered yet. When the algorithm terminates, all intersections in the leaf nodes of the final tree are collected.

5. Numerical Examples

In this section, we give two numerical examples to provide further insight into the performance of our method. The first is about learning the winning combinations for the well-known game Tic-Tac-Toe. This example serves to illustrate how *Random Intersection Trees* can succeed in finding interesting interactions when other methods fail. The second example concerns text classification. Specifically, we want to find simple characterisations (using only a few words, or word-stems in this case) for classes within a large corpus in a large-scale text analysis application.

5.1 Tic-Tac-Toe Endgame Prediction

The Tic-Tac-Toe endgame data set (Matheus and Rendell, 1989; Aha et al., 1991) contains all possible winning end states of the game Tic-Tac-Toe, along with which player (white or black) has won for each of these. There are just under 1000 possible such end states, and our goal is to learn the rules that determine which player wins from a randomly chosen subset of these. We use half of the observations for training, and the other half for testing.

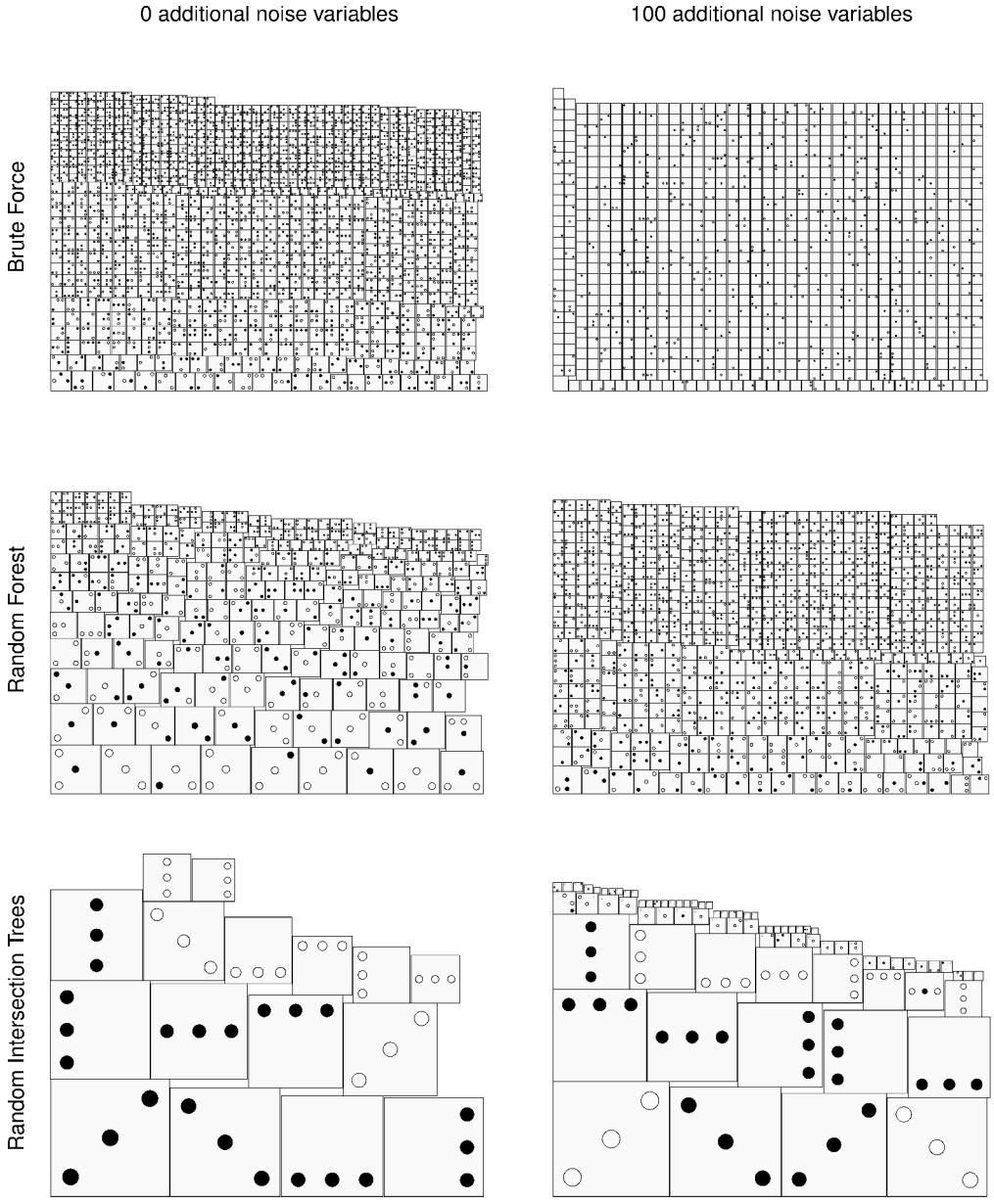


Figure 2: Left panel: patterns that are returned by *Random Intersection Trees* (bottom row), emphRandom Forests of depth 3 (middle row) and brute force search among all interactions of size 3 (top row) for the Tic-Tac-Toe data. Each pattern is scaled to make the area proportional to the empirical frequency with which each pattern is found by these search algorithms. Right panel: the same results in the case when 100 noise variables are added. Note that *Random Intersection Trees* were not constrained to find interactions of depth 3. In the case with noise variables, some of the patterns with the very smallest areas also contained a small number of noise variables, which are not shown. Just counting three- to five-way interactions, there are more than 10^8 potential interactions when 100 noise variables are added.

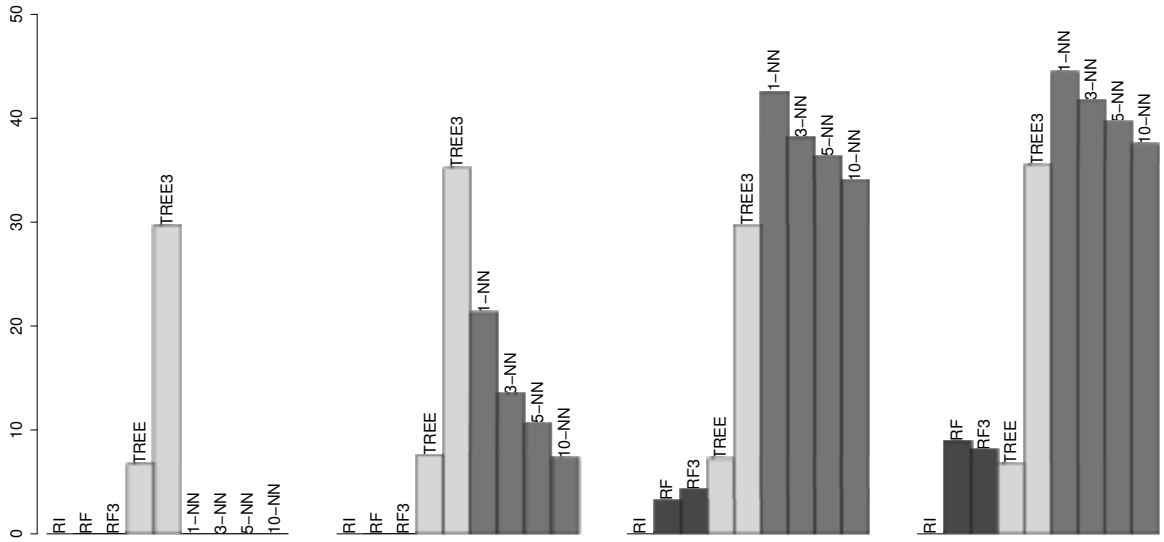


Figure 3: From left to right: the misclassification rate (in %) on Tic-Tac-Toe data for 0, 60, 300 and 400 added noise variables. Each classifier is tuned to have equal misclassification rate in both classes. The simple classifier based on *Random Intersection Trees* (RI) has a misclassification rate of 0% in all cases, as the winning patterns are sampled very frequently (see Figure 2). *Random Forests* (RF) and *Random Forests* limited to depth 3 trees (RF3) are competitive but the misclassification rate increases sharply when many noise variables are added.

There are 9 variables in the original data set which can take the values 'black', 'white' or 'blank'. These can trivially be transformed into a set of twice as many binary variables where the first block of variables encodes presence of black and the second block encodes presence of white.

Two properties of this data set that make it particularly interesting for us here are:

- The presence of interactions is obvious by the nature of the game.
- There are only very weak marginal effects. Knowing that the upper right corner is occupied by a black stone is only very weakly informative about the winner of the game. Greedy searches by trees fail in the presence of many added noise variables and linear models do not work well at all.

We apply *Random Intersection Trees* to finding patterns that indicate a black win (class 1), and also patterns that indicate a white win (class 0). We use the early stopping modifications proposed in Section 4, and create two min-wise hash tables from the available observations in each of the classes, taking $L = 200$. Figure 1 shows how the individual Intersection Trees are constructed and illustrates the use of the early stopping rule. We emphasise that we do not need to specify or know that the winning states are functions of only three variables. We let each tree run until all its branches terminate, and collect all resulting leaves.

Figure 2 illustrates the importance sampling effect of *Random Intersection Trees* when using only the training data, and adding a varying number of noise variables. When adding 100 noise variables, all 16 winning final combinations are among the 40 most frequently chosen patterns. All winning states are chosen hundreds of millions times more often than a random sampling of interactions would pick them.

As discussed in Section 1, the interactions or rules that are found could be entered into any existing aggregation method, such as *Rule Ensembles* (Friedman and Popescu, 2008) or *Decision Lists* (Marchand and Sokolova, 2006; Rivest, 1987). Here, we consider an even simpler aggregation method by selecting all patterns during 1000 iterations of *Random Intersection Trees* (with $B = 5$ samples as branching factor in each tree) that were selected by at least two trees. For each selected pattern, we compute the (empirical) class distributions conditional on the presence and absence of the pattern, using the training sample. That is, for each selected pattern S, we compute

$$\mathbb{P}_n(Y = 1 | X \subseteq S) \text{ and } \mathbb{P}_n(Y = 1 | X \nsubseteq S).$$

Then, given an observation from the test set, we classify according to the average of the log-odds of being in class 1 calculated from each of the conditional probabilities above.

Figure 3 shows the misclassification rates under situations with different numbers of added noise variables. The simple prediction based on *Random Intersection Trees* achieves perfect classification even when 400 noise variables are added. Neither k-NN nor *CART* (Breiman et al., 1984), either restricted to trees of depth 3 (TREE3) or depth chosen by cross-validation (TREE), are as successful, giving misclassification rates between 5% and 40%. Interestingly, trees of depth 3 perform much worse than deeper trees. The winning patterns are not identified in a pure form but only after some other variables have been factored in first. This also means that it is very hard to read the winning states of the trees, unlike the patterns obtained by our method. *Random Forests* also maintain a 0% misclassification rate up until about a hundred added noise variables but start to degrade in performance when further noise variables are added. It is easy to identify the noise variables from a variable importance plot (Strobl et al., 2008). However, within the signal variables the patterns are not easy to see since each variable is approximately equally important for determining the winner (with the slight exception of the middle field in the 3×3 board which is more important than the other fields) and the nature of the interactions is thus not obvious from analysing a *Random Forest* fit.

5.2 Reuters RCV1 Text Classification

The Reuters RCV1 text data contain the *tf-idf* (term frequency-inverse document frequency) weighted presence of $47,148$ word-stems in each document; for details on the collection and processing of the original data, see Lewis et al. (2004). Each document is assigned possibly more than one topic. Here we are interested in whether *Random Intersection Trees* is able to give a quick and accurate summary of each topic. For each topic, we seek sets of word-stems, S, whose simultaneous presence is indicative of a document falling within that topic.

To evaluate the performance of *Random Intersection Trees*, we divide the documents into a training and test set with the first batch of $23,149$ documents as training and the

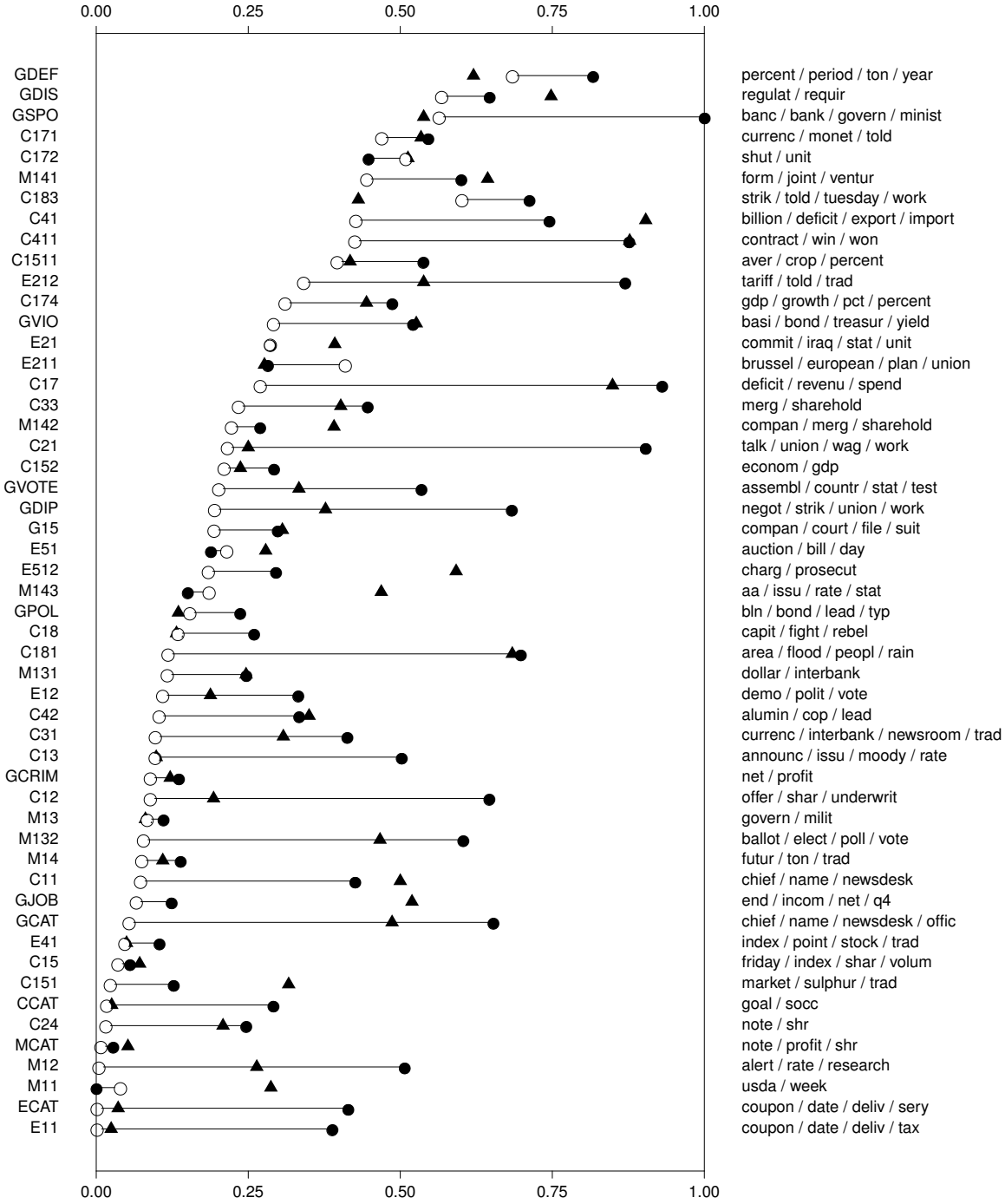


Figure 4: The misclassification rate $\mathbb{P}_n(c \notin Y | S \subseteq X)$ on the test data for a pattern S chosen with a tree ensemble node generation mechanism (black circles), *Random Intersection Trees* (white circles), and a linear method (black triangles) for topics $c \in C$ in the Reuters RCV1 text classification data. The topics are shown on the left and the word combinations chosen by *Random Intersection Trees* on the right.

following 30000 documents as test documents. We compare our procedure to an approach based on *Random Forests* and a simple linear method.

Random Forests and classification trees can be very time- and memory-intensive to apply on a data set of the scale we consider here. In order to be able to compute *Random Forests*, we only consider word-stems if they appear in at least 100 documents in the training data. This leaves 2484 word-stems as predictor variables. We also only consider topics that contain at least 200 documents. To simplify the problem further, we consider a binary version of the predictor variables for all methods, using a 1 or 0 to represent whether each tf-idf value is positive or not.

Let C be the set of topics in our modified data set. Let $Y \subseteq C$ indicate the topics that a given document belongs to. Consider a topic or class $c \in C$. Our goal is to find patterns S that maximise

$$\mathbb{P}_n(c \in Y | S \subseteq X), \tag{11}$$

whilst also maintaining that the prevalence of S among all observations be bounded away from 0. Specifically, we shall require that

$$\mathbb{P}_n(S \subseteq X) \geq p_c/10 \text{ where } p_c = \mathbb{P}_n(c \in Y). \tag{12}$$

To see how this can be cast within the framework set in (1), note that if $S^\star$ maximises (11) and $S^{\star\star}$ satisfies

$$\begin{aligned} \mathbb{P}_n(S^{\star\star} \subseteq X | Y \in c) \geq \mathbb{P}_n(S^\star \subseteq X | Y \in c) \text{ and} \\ \mathbb{P}_n(S^{\star\star} \subseteq X | Y \notin c) \leq \mathbb{P}_n(S^\star \subseteq X | Y \notin c), \end{aligned}$$

then

$$\begin{aligned} \mathbb{P}_n(c \in Y | S^\star \subseteq X) &= \frac{\mathbb{P}_n(S^\star \subseteq X | c \in Y)\mathbb{P}_n(c \in Y)}{\mathbb{P}_n(S^\star \subseteq X | c \in Y)\mathbb{P}_n(c \in Y) + \mathbb{P}_n(S^\star \subseteq X | c \notin Y)\mathbb{P}_n(c \notin Y)} \\ &\leq \frac{\mathbb{P}_n(S^{\star\star} \subseteq X | c \in Y)\mathbb{P}_n(c \in Y)}{\mathbb{P}_n(S^{\star\star} \subseteq X | c \in Y)\mathbb{P}_n(c \in Y) + \mathbb{P}_n(S^{\star\star} \subseteq X | c \notin Y)\mathbb{P}_n(c \notin Y)} \\ &= \mathbb{P}_n(c \in Y | S^{\star\star} \subseteq X), \end{aligned}$$

whence $S^{\star\star}$ also maximises (11) by optimality of $S^\star$. Thus treating those documents belonging to topic c as class 1, and all others as class 0, by solving (1) with θ_0 and θ_1 chosen appropriately, we can obtain all solutions to (11).

In view of this, we use each of the methods to search for patterns S that have high prevalence for a given topic c. We then remove all patterns that do not satisfy (12) on the test data. Then, from the remaining patterns, we select the one that maximises (11) on training data. Below, we describe specific implementation details of each of the methods under consideration.

To compute *Random Intersection Trees*, we create the min-wise hash table for the prevalence among all samples once, using 200 permutations with associated min-wise hash values for each word-stem. Then 1000 iterations of the tree search are performed with a cut-off value $\theta_0 = (3/20)p_c$ and all remaining patterns S with a length less than or equal to 4 are retained.

For a tree-based procedure, one approach is to fit classification trees on subsampled data and adding randomness in the variable selection as in *Random Forests* (Breiman, 2001) and then looking among all created leaf nodes for the most suitable node among all nodes created.

We generate 100 trees as in the *Random Forests* method: each is fit to subsampled training data using *CART* algorithm restricted to depth 4, and further randomness is injected by only permitting variables to be selected from a random subset of those available, for each tree. This takes on average between 90% to 110% of the computational time of a non-optimised pure R (R Core Team, 2013) implementation of *Random Intersection Trees* for these data. Note that this is when using the Fortran version of Breiman (2001) for the *Random Forests* node generation; we expect a significant speedup if Fortran or C code were used for *Random Intersection Trees*. We are currently working on such a version and plan to make it available soon. Furthermore, *Random Forests* would scale much worse if many more word-stems were included as variables.

For linear models, we fit a sparse model with at most ℓ predictors (with $\ell \leq 4$), using a logistic model with an ℓ_1-penalty (Tibshirani, 1996; Friedman et al., 2010). We constrain the regression coefficients to be positive since we are only looking for positive associations in the two previously discussed approaches, and want to keep the same interpretability for the linear model. For each value of $\ell \leq 4$, we take S_ℓ to be the set of variables with a positive regression coefficient. We select the largest value of ℓ such that the fraction of documents attaining the maximal value is at least $p_c/10$ and select the associated pattern S_ℓ. (An alternative approach would be to retain the documents with the highest predicted value when using a sparse regression fit. This approach gave very similar results.)

After screening the candidate patterns returned by each of the methods using (12) on all of the topics $c \in C$, we evaluate the misclassification rate $\mathbb{P}_n(c \notin Y | S \subseteq X)$ on the test data. The results for all of the topics are shown in Figure 4. The rules found with *Random Intersection Trees* have a smaller loss than those found with *Random Forests* in all but 5 of the topics. For those topics where *Random Forests* performs better, the difference in loss is typically small. Linear models achieve a smaller loss than *Random Forests* among most of the topics, but only have a smaller loss than *Random Intersection Trees* in 6 topics, performing worse in all remaining 46 topics.

6. Discussion

We have proposed *Random Intersection Trees* as an efficient way of finding interesting interactions. In contrast to more established algorithms, the patterns are not built up incrementally by adding variables to create interactions of greater and greater size. Instead we start from the full interaction $S = \{1, \ldots, p\}$ and remove more and more variables from this set by taking intersections with randomly chosen observations. Arranging the search in a tree increases efficiency by exploiting sparsity in the data. For the basic version of our method (Algorithm 1), we were able to derive a bound on the computational complexity. The bound depends on (a) the prevalence or frequency with which the pattern S appears among observations in class 1, and (b) the overall sparsity of the data, with higher sparsity making it easier to detect the interaction using a given computational budget. In the best case, we can achieve an almost linear complexity bound as a function of p; more generally

our complexity bound typically has a smaller exponent than that for a brute force search. Further improvements can be made by using min-wise hashing techniques to terminate parts of the search (i.e., branches of the Intersection Tree) that have no chance of leading to interesting interactions. Numerical examples illustrate the improved interaction detection power of *Random Intersection Trees* over other tree-based methods and linear models.

There are many diverse ways in which interactions that solve (1) can be used in further analysis. The interactions may be of interest in their own right as shown in both numerical examples. One can also simply use the search to make sure that a data set is unlikely to have strong interactions that could otherwise have been missed. If the aim is to build a classifier, they can be added to a linear model, or built into classifiers based on tree ensembles. For the latter approach one could consider, for example, averaging predictions in a linear way or averaging log-odds as in *Random Ferns* (Bosch et al., 2007). We believe developments along these lines will prove to be fruitful directions for future research. We also plan to generalise the idea to categorical and continuous predictor variables.

Appendix A.

Here we include proofs omitted earlier in the paper.

Proof of Theorem 1. Fix a tree $m \in \{1, \ldots, M\}$ and suppose this has node set $N = \{1, \ldots, J\}$ indexed chronologically (see Section 2). For $d \in \{1, \ldots, D\}$, define

$$N_d = \{j \in N : \text{depth}(j) = d \text{ and } S_j \supseteq S\},$$
$$W_d = |N_d|.$$

Let E be the event that S is contained in S_1, the random sample selected for the root node of tree m. Further, let $G_d(t) = \mathbb{E}(t^{W_d}|E)$, the probability generating function of W_d conditional on the event E.

We make a few simple observations from the theory of branching processes. Firstly, for $d \leq D-1$, $G_{d+1} = G_d \circ G$ where $G := G_1$. To see this, first note that

$$W_{d+1} = \sum_{j \in N_d} \sum_{j' \in \text{ch}(j)} \mathbb{1}_{\{S \subseteq X_{i(j')}\}}.$$

Now conditional on E, the random variables $\sum_{j' \in \text{ch}(j)} \mathbb{1}_{\{S \subseteq X_{i(j')}\}}$ for $j \in N_d$, are independent of N_d. Moreover, they are independent of each other and have identical distributions equal to that of

$$\sum_{j' \in \text{ch}(1)} \mathbb{1}_{\{S \subseteq X_{i(j')}\}} = W_1.$$

This entails

$$\mathbb{E}(t^{W_{d+1}}|W_d = w, E) = \{\mathbb{E}(t^{W_1}|E)\}^w = \{G(t)\}^w.$$

Thus

$$G_{d+1}(t) = \mathbb{E}(\mathbb{E}(t^{W_{d+1}}|W_d, E)|E) = \mathbb{E}(\{G(t)\}^{W_d}|E) = G_d(G(t)),$$

as claimed.

From this we can conclude that if G has a fixed point q, then this must be a fixed point for all G_d. Since each G_d is non-decreasing on $(0, 1]$, we have that for all $d \in \mathbb{N}$, if $q' \leq q$

and $q' \in (0,1]$, then $G_d(q') \leq q$. The relevance of these remarks will become clear from the following: for an $S' \in L_{D,m}$, we have

$$\begin{aligned} G_D(\mathbb{P}(S' \supsetneq S | S' \supseteq S)) &= \sum_{\ell=0}^{\infty} \mathbb{P}(W_D = \ell | E) \mathbb{P}(S' \supsetneq S | S' \supseteq S)^{\ell} \\ &= \sum_{\ell=0}^{\infty} \mathbb{P}(\{W_D = \ell\} \cap \{S \notin L_{D,m}\} | E) \\ &= \mathbb{P}(S \notin L_{D,m} | E). \end{aligned}$$

Thus if we can ensure that $\mathbb{P}(S' \supsetneq S | S' \supseteq S)$ is at most q, then the final probability in the above display will also be at most q. The rest of the proof proceeds with the following steps:

1. Find conditions on F_B, the distribution of the B_j, such that there exists a fixed point of G, q.
2. Find conditions on the tree depth D such that $\mathbb{P}(S' \supsetneq S | S' \supseteq S) \leq q$.
3. Given q establish conditions on M such that the overall probability of recovering S is at least $1 - \eta$.
4. Given F_B, D and M, compute the expected computational cost of the algorithm.

Step 1: Let the distribution of the B_j be such that

$$B_j = \begin{cases} b & \text{with probability } 1 - \alpha, \\ b + 1 & \text{with probability } \alpha. \end{cases}$$

Now given a $q \in (0,1]$, we shall pick $b \in \mathbb{Z}_+$ and $\alpha \in [0,1)$ to satisfy $G(q) = q$. To this end, observe that

$$\begin{aligned} G(q) &= (1-\alpha)(1 - \theta_1(1-q))^b + \alpha(1 - \theta_1(1-q))^{b+1} \\ &= [1 - \{(\alpha + b) - \lfloor \alpha + b \rfloor\}\theta_1(1-q)]\{1 - \theta_1(1-q)\}^{\lfloor \alpha + b \rfloor}. \end{aligned}$$

From the last displayed equation, we see that $G(q)$ varies with $\alpha + b$ continuously. Furthermore, when $\alpha + b = 0$, $G(q) = 1$, and by making $\alpha + b$ large, we can make $G(q)$ arbitrarily close to 0. Thus by the intermediate value theorem, for any $q \in (0,1]$, $\alpha + b$ can be chosen such that $G(q) = q$.

We now bound $\alpha + b$ from above in terms of q for use later in creating a bound on the complexity of the algorithm. We have

$$\begin{aligned} b + \alpha &= \frac{\log(q) - \log(1 - \alpha\theta_1(1-q))}{\log(1 - \theta_1(1-q))} + \alpha \\ &\leq \frac{-\log(q) + \log(1 - \alpha\theta_1(1-q))}{\theta_1(1-q)} + \alpha \\ &\leq \frac{-\log(q)}{\theta_1(1-q)} \\ &\leq \frac{1 + (1-q)/(2q)}{\theta_1}. \end{aligned} \tag{13}$$

In the final line, we used the inequality

$$\log(z) \geq (z-1) - \frac{(z-1)^2}{2z}, \quad 0 < z \leq 1.$$

Step 2: We now bound $\mathbb{P}(S' \supsetneq S | S' \supseteq S)$ from above in terms of D. The set S' is the intersection of $D+1$ observations selected independently of one another. In order for some $k \in S^c$ to be contained in S', it must have been present in all these $D+1$ observations. Thus by the union bound we have

$$\mathbb{P}(S' \supsetneq S | S' \supseteq S) \ \leq \ \sum_{k \in S^c} \mathbb{P}(k \in S' | S' \supseteq S) \ \leq \ p\nu^{D+1},$$

the rightmost inequality following from (A2).

To ensure this is at most q, we take

$$D = \left\lceil \frac{\log(p/q)}{\log(1/\nu)} \right\rceil - 1, \tag{14}$$

so

$$D \leq \frac{\log(p/q)}{\log(1/\nu)}. \tag{15}$$

Step 3: Turning now to the probability of recovering S, we have

$$\mathbb{P}(S \in L_D) = 1 - [1 - \{1 - \mathbb{P}(S \notin L_{D,m} | E)\}\theta_1]^M.$$

Given the choices of α and b (13), and D (14), we have that $\mathbb{P}(S \notin L_{D,m} | E) \leq q$. Thus taking M to be at least

$$\frac{-\log(\eta)}{(1-q)\theta_1} \geq \frac{-\log(\eta)}{\log\{1-(1-q)\theta_1\}} \tag{16}$$

guarantees recovery of S with probability at least $1-\eta$.

Step 4: To bound the complexity of the algorithm, observe that $\mathbb{E}(B_j) = b + \alpha$, so

$$\begin{aligned} C(M, D, F_B) &\leq \log(p) M \sum_{k=1}^{p} [(b+\alpha)\delta_k + \cdots + \{(b+\alpha)\delta_k\}^D] \\ &\leq \log(p) M D \left[p + \sum_{k:(b+\alpha)\delta_k > 1} \left\{ \left((b+\alpha)\delta_k\right)^D - 1 \right\} \right]. \end{aligned} \tag{17}$$

Substituting Equations (13), (15) and (16) into the complexity bound (17), and writing $\epsilon = (1-q)/(2q)$ gives a bound for the computational complexity of

$$\log(p) \frac{\log(1/\eta)}{\theta_1} \frac{1+2\epsilon}{2\epsilon} \frac{\log\{p(1+2\epsilon)\}}{\log(1/\nu)} \left[p + \sum_{k:(1+\epsilon)\delta_k > \theta_1} \left\{ \left(p(1+2\epsilon)\right)^{\frac{\log\{(1+\epsilon)\delta_k/\theta_1\}}{\log(1/\nu)}} - 1 \right\} \right]. \tag{18}$$

Given that ϵ is bounded above, removing constant factors not depending on p, we get that the order of the computational complexity is bounded above by

$$\log(1/\eta)\frac{\log^2(p)}{\epsilon}\Big\{p + \sum_{k:(1+\epsilon)\delta_k>\theta_1} \big(p^{\frac{\log\{(1+\epsilon)\delta_k/\theta_1\}}{\log(1/\nu)}} - 1\big)\Big\}.$$

■

Proof of Corollary 2. Note that

$$\sum_{k:(1+\epsilon)\delta_k>\theta_1} p^{\frac{\log((1+\epsilon)\delta_k/\theta_1)}{\log(1/\nu)}}$$

is bounded by

$$(1+\epsilon)^{\frac{\log(p)}{\log(1/\nu)}}\Big(p^{\gamma}\cdot p^{\alpha^\star/\beta}\mathbb{1}_{\{\alpha^\star/\beta>0\}} \;+\; p\cdot p^{\alpha_\star/\beta}\mathbb{1}_{\{\alpha_\star/\beta>0\}}\Big).$$

The result then follows from substituting into (18) and taking $\epsilon \propto 1/\log(p)$ ■

Proof of Equation (6)**.** Writing $r = n\pi_2(S)$, we have

$$\begin{aligned}\binom{n}{r}\mathbb{E}_\sigma(\min_{k\in S} h_\sigma(k)) &= \sum_{\ell=1}^{n-r+1} \ell\binom{n-\ell}{r-1} \\ &= \sum_{\ell=1}^{n-r+1}\Big\{(\ell-1)\binom{n-(\ell-1)}{r} - \ell\binom{n-\ell}{r}\Big\} + \sum_{\ell=1}^{n-r+1}\binom{n-\ell+1}{r}.\end{aligned}$$

The first two terms sum to zero leaving only the final term. Thus

$$\begin{aligned}\binom{n}{r}\mathbb{E}_\sigma(\min_{k\in S} h_\sigma(k)) &= \sum_{\ell=1}^{n-r+1}\Big\{\binom{n-\ell+2}{r+1} - \binom{n-\ell+1}{r+1}\Big\} \\ &= \binom{n+1}{r+1}, \qquad (19)\end{aligned}$$

whence

$$\mathbb{E}_\sigma(\min_{k\in S} h_\sigma(k)) = \frac{n+1}{r+1}. \qquad (20)$$

■

Proof of Theorem 4. Writing

$$\tilde{\pi}_2^{-1}(L;S,H) := \tfrac{1}{L}\sum_{l=1}^{L}\min_{k\in S} H_{lk}$$

and suppressing dependence on S and H, we have

$$\begin{aligned}\hat{\pi}_1\hat{\pi}_2 - \pi_1\pi_2 &= \frac{(n+1-\tilde{\pi}_2^{-1})\hat{\pi}_1}{n\tilde{\pi}_2^{-1}} - \pi_1\pi_2 \\ &= \frac{n+1-\tilde{\pi}_2^{-1}}{n\tilde{\pi}_2^{-1}}\Big\{(\hat{\pi}_1-\pi_1) - \pi_1\frac{n\pi_2+1}{n+1-\tilde{\pi}_2^{-1}}\Big(\tilde{\pi}_2^{-1} - \frac{n+1}{n\pi_2+1}\Big)\Big\}. \qquad (21)\end{aligned}$$

Consider $L \to \infty$. By the weak law of large numbers and the continuous mapping theorem, we have

$$\frac{n+1-\tilde{\pi}_2^{-1}(L)}{n\tilde{\pi}_2^{-1}(L)} \xrightarrow{p} \pi_2 \quad \text{and}$$
$$\frac{n\pi_2+1}{n+1-\tilde{\pi}_2^{-1}(L)} \xrightarrow{p} \frac{(\pi_2+n^{-1})^2}{\pi_2(1+n^{-1})}.$$

By the central limit theorem, Slutsky's lemma and Lemma 5,

$$A_L := \sqrt{L}(\hat{\pi}_1(L)-\pi_1) \xrightarrow{d} N(0,\pi_1(1-\pi_1)) \quad \text{and}$$
$$B_L := -\pi_1\frac{n\pi_2+1}{n+1-\tilde{\pi}_2^{-1}(L)} \times \sqrt{L}\left(\tilde{\pi}_2^{-1}(L)-\frac{n+1}{n\pi_2+1}\right) \xrightarrow{d} N(0,\pi_1^2(1-\pi_2)(1+\epsilon(n))),$$

with $\epsilon(n)$ defined as in (10). Define $I_S := \{i : S \subseteq X\}$ and let $k \in S$. Now observe that

$$\{\exists \iota' : h_\sigma(k) = \iota' \text{ for all } k' \in S\} = \{\sigma^{-1}(h_\sigma(k)) \in I_S\} \text{ and } \{\min_{k\in S} h_\sigma(k) = \iota\}$$

are independent: in words, the distribution of $\min_{k\in S} h_\sigma(k)$ conditional on the fact that an observation index in I_S was permuted to a lower value than any in $I_k \setminus I_S$ is the same as its unconditional distribution. This implies the independence of $\hat{\pi}_1$ and $\tilde{\pi}_2^{-1}$ and thence also that of A_L and B_L. Thus we have that for all $t_1, t_2 \in \mathbb{R}$,

$$\mathbb{E}(e^{i(t_1A_L+t_2B_L)}) = \mathbb{E}(e^{it_1A_L})\mathbb{E}(e^{it_2B_L}) \to \exp[\tfrac{1}{2}t_1^2\pi_1(1-\pi_1) + \tfrac{1}{2}t_2^2\{\pi_1^2(1-\pi_2)(1+\epsilon(n))\}].$$

pointwise as $L \to \infty$. Returning to (21), by Lévy's continuity theorem we have

$$\sqrt{L}\{\hat{\pi}_1(L)\hat{\pi}_2(L) - \pi_1\pi_2\} \xrightarrow{d} N(0,\pi_2^2\pi_1(1-\pi_1\pi_2)(1+\epsilon(n))).$$

■

Lemma 5 *Let $r = n\pi_2(S)$ and suppose $n \geq r+2$. Then*

$$\mathrm{Var}_\sigma(\min_{k\in S} h_\sigma(k)) = \frac{r(n+1)(n-r)}{(r+1)^2(r+2)}.$$

Proof We have,

$$\begin{aligned}
\binom{n}{r}\mathbb{E}_\sigma\{(\min_{k\in S} h_\sigma(k))^2\} &= \sum_{\ell=1}^{n-r+1} \ell^2 \binom{n-\ell}{r-1} \\
&= \sum_{\ell=1}^{n-r+1} \left\{(\ell-1)^2\binom{n-(\ell-1)}{r} - \ell^2\binom{n-\ell}{r}\right\} \\
&\quad + \sum_{\ell=1}^{n-r+1} \left\{2(\ell-1)\binom{n-(\ell-1)}{r} + \binom{n-\ell+1}{r}\right\} \\
&= 2\binom{n+1}{r+2} + \binom{n+1}{r+1},
\end{aligned}$$

where in the last line we used (19) and (20). Simplifying and using (6) gives the result. ■

References

R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, volume 1215, pages 487–499, 1994.

D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *IEEE 11th International Conference on Computer Vision, 2007*, pages 1–8. IEEE, 2007.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, 1984.

A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 327–336. ACM, 1998.

E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, and C. Yang. Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering*, 13:64–78, 2001.

M. Datar and S. Muthukrishnan. Estimating rarity and similarity over data stream windows. *Lecture Notes in Computer Science*, 2461:323, 2002.

J. Friedman and B. Popescu. Predictive learning via rule ensembles. *Annals of Applied Statistics*, 2:916–954, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33:1–22, 2010.

J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29:1–12, May 2000.

D. Lewis, Y. Yang, T. Rose, and F. Li. Rcv1: a new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

W. Loh and Y. Shih. Split selection methods for classification trees. *Statistica Sinica*, 7: 815–840, 1997.

M. Marchand and M. Sokolova. Learning with decision lists of data-dependent features. *Journal of Machine Learning Research*, 6:427, 2006.

C. Matheus and L. Rendell. Constructive induction on decision trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, volume 645650. Citeseer, 1989.

N. Meinshausen. Node harvest. *Annals of Applied Statistics*, 4:2049–2072, 2010.

J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang. H-mine: hyper-structure mining of frequent patterns in large databases. *ICDM 2001, Proceedings of the IEEE International Conference on Data Mining*, pages 441–448, 2001.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. URL `http://www.R-project.org/`.

R. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

C. Strobl, A. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis. Conditional variable importance for random forests. *BMC Bioinformatics*, 9:307, 2008.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

Journal of Machine Learning Research 15 (2014) 655-696 Submitted 9/12; Revised 12/13; Published 2/14

Reinforcement Learning for Closed-Loop Propofol Anesthesia: A Study in Human Volunteers

Brett L Moore BRETT.MOORE@IEEE.ORG
Department of Computer Science
Texas Tech University
Lubbock, TX 79409, USA

Larry D Pyeatt LARRY.PYEATT@SDSMT.EDU
Department of Mathematics and Computer Science
South Dakota School of Mines and Technology
Rapid City, SD 57701, USA

Vivekanand Kulkarni VKULKARNI@STANFORD.EDU
Periklis Panousis PANOUSIS@STANFORD.EDU
Kevin Padrez KPADREZ@GMAIL.COM
Anthony G Doufas AGDOUFAS@STANFORD.EDU
Department of Anesthesiology, Perioperative and Pain Medicine
Stanford University School of Medicine
Stanford, CA, 94305, USA

Editor: Peter Dayan

Abstract

Clinical research has demonstrated the efficacy of closed-loop control of anesthesia using the bispectral index of the electroencephalogram as the controlled variable. These controllers have evolved to yield patient-specific anesthesia, which is associated with improved patient outcomes. Despite progress, the problem of patient-specific anesthesia remains unsolved. A variety of factors confound good control, including variations in human physiology, imperfect measures of drug effect, and delayed, hysteretic response to drug delivery. Reinforcement learning (RL) appears to be uniquely equipped to overcome these challenges; however, the literature offers no precedent for RL in anesthesia. To begin exploring the role RL might play in improving anesthetic care, we investigated the method's application in the delivery of patient-specific, propofol-induced hypnosis in human volunteers. When compared to performance metrics reported in the anesthesia literature, RL demonstrated patient-specific control marked by improved accuracy and stability. Furthermore, these results suggest that RL may be considered a viable alternative for solving other difficult closed-loop control problems in medicine. More rigorous clinical study, beyond the confines of controlled human volunteer studies, is needed to substantiate these findings.

Keywords: reinforcement learning, bispectral index, propofol, anesthesia, hypnosis, closed-loop control

1. Introduction

When compared to standard population-based dosing, patient-specific drug administration is generally preferred in the clinical practice of anesthesia. Computer-controlled drug deliv-

ery systems have been investigated as a means of achieving patient-specific anesthesia(Liu et al., 2013, 2012; Hahn et al., 2011; Hemmerling et al., 2010), and their application is associated with a number of favorable patient outcomes, including decreased intraoperative drug consumption and shortened postoperative recovery times (Liu et al., 2006; Servin, 1998; Theil et al., 1993). Historically, the application of conventional control techniques, such as proportional-integral-derivative (PID) control, in closed-loop anesthesia has shown moderate success (Absalom and Kenny, 2003). However, these historical successes have been constrained by the PID method's inherent limitations, as well as the complexity of human physiology (Wood, 1989). To improve control performance, clinical study has broadened to include techniques commonly associated with intelligent systems, most notably Bayesian filtering and fuzzy control(Ching et al., 2013; Shanechi et al., 2013; De Smet et al., 2008; Esmaeili et al., 2008; Carregal et al., 2000; Schaublin et al., 1996).

Reinforcement learning (RL), one of many intelligent system techniques, has demonstrated proficiency in difficult robotic control tasks (Gullapalli, 1993). However, RL has no reported application to clinical control problems, with the exception of work leading to this study (Moore et al., 2011a,b, 2004). Nonetheless, RL has a presence in medicine, and reported applications include ultrasound image segmentation (Sahba et al., 2008) and planning tasks, such as scheduling of HIV therapy (Ernst et al., 2006), optimizing deep-brain stimulation in epilepsy treatment (Guez et al., 2008), dosing strategies for anemia management in patients with chronic renal failure (Martín-Guerrero et al., 2009; Gaweda et al., 2006), and clinical trial design (Zhao et al., 2009). These applications support the assertion that reinforcement learning can serve as a "medical decision aid" (Martín-Guerrero et al., 2009). However, RL's aptitude for specialized clinical application remains incompletely explored since these applications were non-clinical. In the examples cited, RL was applied to data collected from patients, but no RL algorithm contributed directly to patient care.

This lack of direct application does not imply that RL is unsuited for computer-controlled drug delivery since the method has been successfully applied to critical real-time industrial control tasks (Ernst et al., 2009). Furthermore, the basic principles of reinforcement learning (dynamic programming and value function optimization) have been studied in depth-of-anesthesia control with favorable results (Hu et al., 1994). Thus, the two-fold objectives of this study were to a) investigate the clinical suitability of reinforcement learning for closed-loop control of intravenous propofol anesthesia in healthy human volunteers, and b) compare the performance of RL control against published clinical metrics. To accomplish these objectives, an RL agent was developed, tested *in silico*, and then evaluated in healthy volunteers under an IRB-approved study protocol in the Stanford University School of Medicine Department of Anesthesiology, Perioperative, and Pain Medicine.

2. Background

To begin answering the question "why should reinforcement learning be applied in anesthesia," this section establishes the problem with an introduction to the motivation and challenges of closed-loop control of intraoperative hypnosis. Discussion continues by summarizing the manner in which RL can address deficiencies in some contemporary approaches.

2.1 Propofol-Induced Hypnosis

Propofol is a short-acting sedative agent administered intravenously to achieve induction and maintenance of general anesthesia in the operating room and other critical care arenas. Propofol suppresses higher brain function to produce *hypnosis*, a suppression of consciousness.[1] Propofol, like other hypnotic agents, achieves unconsciousness "by altering neurotransmission at multiple sites in the cerebral cortex, brain stem, and thalamus." (Brown et al., 2010, p. 2641). For a thorough treatment of propofol's mechanism of action, see Brown et al.

The anesthesia community has studied automated delivery of propofol-induced hypnosis, in part, because the drug and its pharmacodynamic effects satisfy basic requirements for closed-loop control. To accomplish such *feedback* control, a controller must first be equipped to a) influence the desired control parameter, and b) observe the affects of its actions on that control parameter. The short-acting nature of propofol, characterized by rapid onset and recovery, readily satisfies the first requirement (Vanlersberghe and Camu, 2008). The complexities of the human central nervous system and its interaction with propofol make objective, quantitative measurement of hypnosis (control effect) challenging, but—as the following sections show—measurement of propofol effect is feasible.

2.2 Depth of Hypnosis Measurement

This section introduces the use of the electroencephalogram and its derivatives in the assessment of hypnotic depth. Some of the challenges associated with these methods are also discussed.

2.2.1 Electroencephalogram (EEG)

In closed-loop regulation of hypnosis, the controlled variable is the patient's level of consciousness, or awareness. Cerebral electrical activity is correlated with consciousness (Brown et al., 2010), and hypnosis (suppression of awareness) displays as change in cortical electrical activity. Electroencephalography, the measurement of cerebral electrical activity, produces the *electroencephalogram* (EEG). The EEG is often obtained with an non-invasive array of scalp sensors. Normal, waking cortical electrical activity is marked by periodic signals in five narrow frequency bands, α, β, γ, δ, and θ. When an EEG is obtained transcutaneously, signals within these bands range in the tens of millivolts. Accurate capture of this low-power signal is complicated by non-cortical biologic artifacts: eye motion and blinking, facial muscle movement, and cardiac pulse (Fitzgibbon et al., 2007).[2] Other factors, like changes in skin conductance, can impact the fidelity of signal acquisition. The EEG is also susceptible to contamination from electrical sources found in the intraoperative environment: power lines, overhead lighting, electrocautery, and other medical devices. For these reasons, isolation and removal of non-cortical artifacts remains a challenging problem for EEG analysis and interpretation.

1. Hypnosis is just one member of a collection of clinical endpoints that comprise "general anesthesia"; others include akinesia (immobility), amnesia, analgesia, and autonomic system stability.
2. Electrooculography (EOG), electromyography (EMG) and electrocardiography (ECG) are the practices of measuring these "unwanted" signals.

Research has shown that when Fourier analysis (or another time-frequency analysis method) is applied to the EEG, the energy content within the five spectral bands can provide insight into depth of hypnosis since the spectral features of the EEG signal are modulated with level of consciousness. Studies demonstrate that the hypnotic component of general anesthesia "produces distinct patterns on the electroencephalogram (EEG), the most common of which is the progressive increase in low-frequency, high-amplitude activity as the level of general anesthesia deepens." (Brown et al., 2010, p. 2638).

2.2.2 Bispectral Analysis of the EEG

The practiced anesthetist can discern and interpret changes in EEG power spectra associated with hypnosis induction, maintenance, and emergence; however, the relationship of these spectral components (and their changes) to depth of hypnosis is not obvious. Thus, processed EEG variables have been studied with the aim of developing a "simplified interpretation of the EEG" for objective, broadly-applicable measures of anesthetic depth (Sigl and Chamoun, 1994, p.392). One such indicator, the bispectral index (Covidien, Mansfield, MA), is well-reported in the anesthesia literature. BIS, as it is known, differs from conventional quantitative EEG parameters in that it augments traditional power spectral (Fourier) methods with *bispectral analysis*, a means of measuring *phase coupling* between pairs of frequency components. This added dimension of *bicoherence* can improve identification of EEG patterns associated with varying levels of cortical activity.

2.2.3 The Bispectral Index of the EEG (BIS)

Sigl and Chamoun define the bispectral index of the EEG as "a multivariate measure incorporating bispectral and time-domain parameters derived from the EEG," (1994, p. 402). This proprietary index was developed by statistically linking the EEG's time- and frequency-domain features to a database of hand-selected "behavioral assessments of sedation and hypnosis," (Rampil, 1997, p. 998). The result, BIS, is a weighted sum of processed EEG features tied to the clinical endpoints of hypnosis that is "insensitive to the specific anesthetic or sedative agent," (Rampil, 1997, p. 1000). This bispectral index lies in the range [0, 100] (Sigl and Chamoun, 1994; Rampil, 1997). A measure of 100 is associated with normal wakefulness; a value of 0 correlates to an iso-electric brain state.[3]

Research has shown that evidence of propofol's pharmacodynamic effect may be observed in the bispectral index of the EEG: "The BIS both correlated well with the level of responsiveness and provided an excellent prediction of the loss of consciousness. These results imply that BIS may be a valuable monitor of the level of sedation and loss of consciousness for propofol, midazolam, and isoflurane." (Glass et al., 1997). This finding is consistent with expectation: BIS was developed to be a statistical correlation between EEG patterns and clinical attributes of hypnosis: loss of consciousness, progressive loss of reflexes, return of consciousness, etc.

However, BIS has been observed to be an imperfect indicator of hypnotic condition. Some of the challenges stem from noise contamination in the underlying EEG signal. For example, EMG signals, such as those resulting from eye or facial motion, may overlap the

3. Thorough treatments of bispectral analysis of the EEG may be found in Sigl and Chamoun (1994) and Rampil (1997).

EEG's higher frequency β and γ bands. This sort of EEG signal contamination has been associated with elevated BIS values in surgical patients (Renna et al., 2002). To attenuate the influence of electrical noise, the A-2000 BIS monitor, like the one used in this study, applies selective band-pass and low-pass digital filters in its process of computing BIS.

Research also indicates that contamination of the BIS signal extends beyond external electrical noise; normal physiologic processes can play a role. BIS and EMG variability, characterized by relatively high-frequency fluctuation, has been observed to predict somatic responses to noxious stimulus (Bloom et al., 2008; Greenwald and Rosow, 2006). In more recent work, measures of BIS and EMG variability were computed as standard deviations over a 3-min window of samples. The resulting sBIS and sEMG indicators predicted somatic response to painful stimuli (Mathews et al., 2012). The implication is meaningful to closed-loop control of hypnosis: BIS variability seems positively correlated to lack of analgesia, rather than hypnosis. Since propofol is not an analgesic agent, it's reasonable to conclude that high-frequency changes in BIS should not contribute to propofol delivery decisions. In acknowledgment of these issues, the A-2000 BIS monitor provides a user-selectable option that applies either a 15-sec or 30-sec smoothing window to its BIS measurements. The manual advises the user to select the smoothing windows according to a desire for "decreased delay" or "decreased variability."

Other research highlights additional sources of "noise" that may influence the BIS signal. Dahaba provides an excellent survey of clinical and physiological conditions that perturb BIS measurement (2005). In light of these factors, it's reasonable to consider BIS as a probabilistic indicator of hypnotic depth, not an absolute one. As such, probabilistic control methods, like RL, become increasingly relevant.

2.3 Motivation for Good Control of Hypnosis

BIS has been recently studied as a mitigation for risk of unintended intraoperative awareness, defined as conscious behavior (motion, vocalization, etc.) during surgery or post-operative recall of intraoperative events. Unintentional intraoperative awareness can challenge the anesthetist because doses ensuring adequate hypnosis may lead to hemodynamic and/or respiratory instabilities in sensitive patients (i.e., trauma, critically-ill, and elderly). While the incidence of intraoperative awareness is estimated to be low, 0.13% (Sebel et al., 2004), it can be severely traumatic for the patient. BIS monitoring has been recommended as a preventative measure (Sandin et al., 2000) and has been reported to reduce the incidence of unintended intraoperative awareness (Myles et al., 2000). This finding remains controversial since this evidence comes from observational clinical trials (Avidan et al., 2008), and the execution of a convincing prospective clinical trial is logistically difficult.

At first glance, the risk of unintended intraoperative awareness implies that "deeper is better." However, higher doses of propofol are correlated with respiratory and hemodynamic depression. Emerging research substantiates a balance in hypnosis with reports of a possible causal link between deep anesthesia (BIS < 45) and postoperative morbidity (Lindholm et al., 2009). Again, this conclusion requires further substantiation before wide-spread acceptance.

These opposing concerns, awareness versus toxicity, as well as the favorable outcomes cited previously, link good control of intraoperative anesthesia to good patient care. Conse-

quently, closed-loop control of propofol-induced hypnosis is well-represented in the literature (Liu et al., 2013, 2012; Hahn et al., 2011; Hemmerling et al., 2010; Struys et al., 2007, 2004; Absalom and Kenny, 2003; Leslie et al., 2002; Absalom et al., 2002; Sakai et al., 2000; Struys et al., 2001), yet accurate and stable control of intraoperative hypnosis remains an incompletely solved problem.

2.4 Challenges to Optimal Control of Hypnosis

Optimal control of propofol-induced hypnosis is a difficult problem for several reasons. Properties of the patient, the drug, and the intraoperative environment all contribute confounding influences. The patient's age, gender, and ethnicity, as well as disease and surgical intervention (Schnider et al., 1998; Barvais et al., 1996), are known to affect response to propofol infusion. Additionally, "intra-subject heterogeneity", or tendency for change in an individual (Rigby-Jones and Sneyd, 2012), assures that any accurate characterization of a patient's propofol response has a limited lifetime. For these reasons, commercially available target-controlled infusion (TCI) systems rely on general, population models of drug effect, leaving them unequipped for patient-specific drug delivery.

Additionally, a system regulating a patient's propofol concentration is limited to an *asymmetric* influence that further hinders good control. Propofol concentrations can be readily increased via intravenous infusion; however, the system lacks a direct means of decreasing concentration. Instead, the controller must wait for the patient to decrease propofol concentration through metabolism or redistribution. As a consequence, the controller possesses a direct means of increasing hypnosis, but an indirect means of decreasing hypnosis.

Other aspects of propofol infusion are problematic. The delay between action (infusion) and effect (hypnosis) can exceed two minutes. This delay (*transport delay* in control literature) is variable, hysteretic, and demonstrates flow rate dependence (Struys et al., 2007; Pilge et al., 2006). In addition, propofol's effect on consciousness is nonlinear, meaning that a fixed dose of propofol can impact BIS differently, depending on the patient's level of hypnosis at the time of infusion. As a result, the controller cannot always assume that a chosen dose will always have the same effect.

Finally, hypnosis is a balance of stimulus and drug effect. In the absence of stimulus, a relatively low concentration of propofol can yield the desired BIS. The onset of a routine surgical event (incision, manipulation, etc.) can disturb this equilibrium, rendering the patient's previously adequate concentration insufficient and leading to an undesired increase in consciousness. Thus, a clinically relevant hypnosis control system should be prepared to compensate for those external influences that can negatively impact control (Röpcke et al., 2001b; Ausems et al., 1986).

2.5 Conventional Control

Much of the initial progress in closed-loop anesthesia has been accomplished using conventional control techniques, like Proportional-Integral-Derivative (PID) control (Absalom et al., 2002; Struys et al., 2001; Sakai et al., 2000; Kenny and Mantzaridis, 1999; Mortier et al., 1998). These classical control methods enjoy widespread industrial application due to their simplicity of design and implementation, as well as their success in many control

problems. Furthermore, a measure of the PID technique's popularity is due to its foundation in classical control theory, its lack of dependence upon an accurate process model, and its ease in implementation.

Given the clinical interest in well-controlled hypnosis, it is no surprise that PID (along with its PI and PD variants) has been applied to hypnosis control. Kenny and Mantzaridis used a basic proportional-integral controller to regulate hypnosis in surgical patients (1999). This automated system delivered satisfactory anesthesia in a population of 100 patients and demonstrated that the hypnotic process, although noisy and uncertain, may be regulated using conventional techniques. Further research has demonstrated similar results: Absalom et al. coupled the bispectral index with a PID controller and observed largely satisfactory results in the administration of general anesthesia in ten patients (2002).

Despite instances of successful PID control in general anesthesia, the technique should not be applied universally with an expectation of similar results. Constant-coefficient PID methods, like those historically applied in hypnosis control, are not equipped to satisfactorily control processes with variable time delays, variable plant parameters, significant nonlinearities, and non-negligible process noise. Olkkola summarizes the use of PID in closed-loop control of anesthesia: "PID controllers are in general not universally applicable to nonlinear concentration-response curves..." (Olkkola et al., 1991, 420). Our simulation work supports this assertion (Moore, 2003). In general, a PID controller may be tuned to perform well for an arbitrary patient at a fixed level of hypnosis. However, the same controller would perform poorly when patient characteristics or hypnosis target varied. More convincingly, clinical observations support Olkkola's claim, as well. Absalom et al. observed oscillation around the BIS setpoint in the operating room (2002), and Leslie et al. observed similar oscillations in a conscious sedation experiment (2002).

Given the known limitations of the constant-coefficient PID controller, as well as the reported instances of sub-optimal control, it can be reasonably concluded that constant-coefficient PID is not the ideal solution (Struys et al., 2001). Current anesthesia literature suggests the ideal solution is a model-based, adaptive system(Ching et al., 2013; Shanechi et al., 2013). These systems do not exclude the PID class of controllers since neural networks, among other methods, have been used to establish relationships between system inputs and variable PID coefficients (Omatu et al., 1996). However, it should be noted that adding a model increases the complexity of the PID controller, thereby eroding its advantage of simplicity.

2.6 Reinforcement Learning

Reinforcement learning (RL) is an intelligent control method that provides a structured, mathematically robust mechanism for goal-directed decision making in which long-term gain is maximized (Sutton and Barto, 1998; Kaelbling et al., 1996). Unlike supervised learning methods, no examples of desired behavior are provided during training; instead, favorable action choices are encouraged through positive and/or negative reinforcements. Under this framework, knowledge is gained through experimentation: actions are chosen, effects are observed, and rewards are gained.

3. Methods

To begin assessing the suitability of reinforcement learning for closed-loop control of hypnosis, an RL agent was developed in the Texas Tech University Computer Science Department under the supervision of the study's principal technical investigator, Dr. Pyeatt. In cooperation with the Stanford University School of Medicine Department of Anesthesiology, Perioperative and Pain Medicine, intraoperative patient models were developed for agent training and *in silico* evaluation under the supervision of the study's principal clinical investigator, Dr. Doufas. Section 3.1 summarizes evolution of the RL agent; however, more thorough treatment of the design, development, and *in silico* testing of the RL agent may be found in our previously reported work (Moore et al., 2011a,b). After the agent was validated in simulation and the clinical study protocol gained Institutional Review Board approval, fifteen healthy volunteers underwent RL-controlled propofol hypnosis in surgical facilities of the Stanford University School of Medicine.

3.1 The Clinical-Grade RL Agent

This section addresses the development and application of the clinical-grade RL agent. As such, the architecture, training, and *in-silico* evaluation are covered. The section then presents the application of *Reagent*, a data collection and control system using the RL agent to administer propofol hypnosis in a population of healthy human volunteers.

3.1.1 Agent Architecture

The RL agent was implemented as a *Markov Decision Process* (MDP), a mathematical framework for optimal decision-making in stochastic systems. A principal feature of the MDP is the *Markov Property*, a characteristic in which the conditional probability of state transition depends solely on the action chosen in the current state—as opposed to some longer historical sequence of state visitation and action selection (Russel and Norvig, 2002; Sutton and Barto, 1998). Littman (1994) formally defines the general MDP as a system consisting of:

- the set of states $\mathcal{S} = \{s_0, s_1, s_2, \ldots, s_{|\mathcal{S}|-1}\}$,
- the transition probabilities $\Pr(s'|s,a)\,\forall s, s' \in \mathcal{S}, a \in A(s)$,
- the set of actions $\mathcal{A} = \{a_0, a_1, a_2, \ldots, a_{|\mathcal{A}|-1}\}$,
- the set of actions $A(s) \subseteq \mathcal{A}$ for each state $s \in \mathcal{S}$ that can be executed in s,
- and the immediate rewards $r^a(s)\,\forall a \in A(s), s \in \mathcal{S}$ that are available after taking any legal action from any state.

For the purposes of this research, the sets $\mathcal{S}$ and $\mathcal{A}$ were discrete. Of these components, only the transition probabilities $\Pr(s'|s,a)$ were initially unknown. (Agent training is tantamount to the discovery of these transition probabilities. Were they initially known, an optimal control policy could be determined using *dynamic programming*.) With the nature of an RL agent formally defined, discussion continues with the application of the MDP in the context of the hypnosis control task.

3.1.2 Agent Percepts

To achieve and maintain a desired level of hypnosis (BIS_{target}), the agent first observed the patient's bispectral index ($BIS_{measured}$) on five-second intervals as reported by an A-2000 BIS monitor (Covidien, Mansfield, MA). The monitor's BIS smoothing window was set to 15 seconds, the minimum, to grant the agent flexibility in managing BIS measurement noise (see Section 2.2.3).

To reduce the effect of measurement noise on the agent's estimate of patient condition, $BIS_{measured}$ was smoothed using a low-pass filter. From the resulting $BIS_{smoothed}$ signal, two control inputs were then computed: BIS_{error} and ΔBIS_{error}. BIS_{error} was defined as $(BIS_{smoothed} - BIS_{target})$, and ΔBIS_{error} was defined as the change in BIS_{error} over 15 s, or $(BIS_{error}(t) - BIS_{error}(t-2))$. These control signals allowed the agent to observe the magnitude of control error, as well its direction of change. This observation of BIS_{error} and ΔBIS_{error} served as the agent's estimation of the patient's state of hypnosis. Table 1 presents a high-level summary of patient states that may be distinguished using these control signals.

BIS_{error}	ΔBIS_{error}	Interpretation	
< 0	< 0	Good	Below target, improving
< 0	≈ 0	Neutral	Below target, steady
< 0	> 0	Poor	Below target, worsening
≈ 0	< 0	Good	At target, improving
≈ 0	≈ 0	Good	At target, steady
≈ 0	> 0	Poor	At target, worsening
> 0	< 0	Good	Above target, improving
> 0	≈ 0	Neutral	Above target, steady
> 0	> 0	Poor	Above target, worsening

Table 1: Interpreting the agent's control signals

In pilot studies of human volunteers, the combined effects of BIS measurement noise, filtering, and transport delay resulted in oscillatory control behavior. These confounding influences were successfully mitigated by conditioning BIS_{error} and ΔBIS_{error} with sets of fuzzy membership functions (Zadeh, 1965). The fuzzy set memberships for BIS_{error} and ΔBIS_{error} were assessed using two sets of triangular membership functions, $\mu_N(x)$, $\mu_Z(x)$, and $\mu_P(x)$ (Figure 1). The resulting six-dimensional feature vector served as the agent's perceptual input:

$$f = [\mu_N(E),\ \mu_Z(E),\ \mu_P(E), \mu_N(\Delta E),\ \mu_Z(\Delta E),\ \mu_P\,(\Delta E)]$$

(where E represents BIS_{error} and ΔE indicates ΔBIS_{error} for brevity). Since fuzzy set membership is expressed as a real number in the range $[0, 1]$, the continuous feature vector f required transformation before the discrete RL algorithms used in this study could be applied. Section 3.2.2 provides greater detail in the methods used to map the feature vector f to the set of discrete states $\mathcal{S}$ employed in this study.

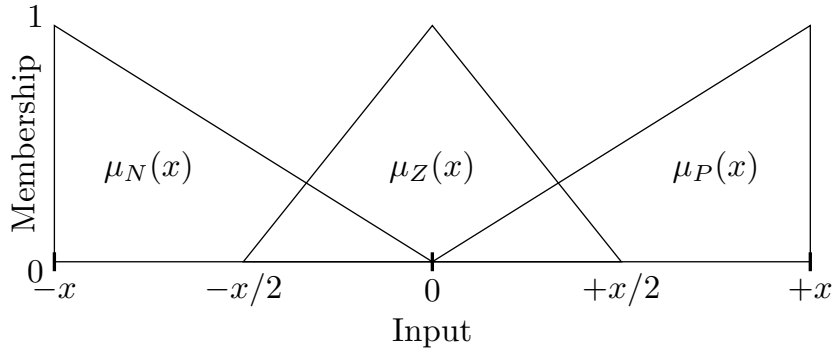


Figure 1: The system input variables, $\text{BIS}_{\text{error}}$ and $\Delta\text{BIS}_{\text{error}}$, were conditioned with sets of fuzzy membership functions. A set of three membership functions operated on $\text{BIS}_{\text{error}}$ ($x = 20$), and a second set of functions operated on $\Delta\text{BIS}_{\text{error}}$ ($x = 10$). The resulting membership values formed a six-dimensional feature vector that served as the agent's patient state descriptor.

3.1.3 Agent Actions

The agent delivered propofol to the volunteer via a catheter placed in the antecubital (elbow) vein using a precision syringe pump (Pump 33, Harvard Apparatus, Holliston, MA). During control, the agent selected an infusion rate from $\mathcal{A}$, a discrete set of 15 flowrates ranging from 0.0 – 6.0 ml/min:

$$\mathcal{A} = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 6.0\} \text{ ml/min}.$$

Once a rate was selected, the chosen action remained in effect for five seconds. A concentration of 1% propofol was assumed for all members of $\mathcal{A}$.

3.1.4 Reinforcements

Although reinforcement learning is unsupervised in the sense that no explicit training exemplars are provided during training, the method assumes the existence of a *critic* that grades behavior as the agent learns. During learning, the critic's role is to dispense reinforcements in order to guide the agent's action selection. In RL, this critic is implemented as the application-specific *reward function*. In the hypnosis control task, the agent's objective was to achieve and maintain the selected BIS target. Expressed alternatively, the agent's goal was to minimize control error for the duration of the control interval. The reward function below presents one system of reinforcement for guiding action selection toward this goal

$$r(t+1) = -|BIS_{error}(t)|. \tag{1}$$

This negative-bounded reward function provided instantaneous rewards proportional to the observed control error. Under this scheme, the agent's sole means of minimizing negative reinforcement was to select actions yielding minimal control error. This reward function

highlights an important characteristic of the hypnosis control task, namely the lack of an explicit goal state. Because the task lacked a definitive persistent terminal state, it was classified as a continuing, non-episodic control task.

3.2 Agent Training

During learning, the naive, uninformed agent was expected to make arbitrarily poor propofol dosing decisions; thus, a simulated intraoperative patient was developed to facilitate agent training in a safe, off-line manner. Consequently, the principal role of this virtual patient was to model the time-dependent effects of propofol infusion, collectively known as the *pharmacokinetic* and *pharmacodynamic* (PK/PD) responses. A drug's pharmacokinetic properties describe its distribution within the body; pharmacodynamic attributes characterize the dose effect. Through experimentation with this virtual patient, the agent was expected to learn the general characteristics of propofol-induced hypnosis with respect to bispectral index: BIS is linked to propofol infusion in an inverse, time-delayed, and nonlinear manner. It should also be noted that this *in silico* patient presented an advantage in its rapid simulation of hypnotic episodes. Reinforcement learning is inherently a process of statistical estimation, and a large number of training episodes were needed to learn the control policy and achieve clinical readiness.

3.2.1 Modeling Propofol Effect

Propofol pharmacokinetics were simulated using a three-compartment model (Schnider et al., 1998), a system which uses central, rapid, and slow compartments to estimate the time-dependent distribution of propofol within the human body. In this model, propofol is introduced into the central compartment via intravenous infusion; the drug is then free to interact with the rapid and slow compartments through first-order, gradient-driven transport. These compartments, which represent collections of tissues with high and low propofol transport coefficients, were derived from empirical observations and sometimes lack direct, obvious mapping to actual physiological systems.

Figure 2 illustrates the Schnider model and its transport coefficients, which vary with patient height, weight, gender, and age. As shown, the coefficients are subscripted to indicate direction of flow (from, to) since the coefficients may differ directionally, that is, the central-to-slow coefficient (k_{cs}) differs from the slow-to-central coefficient (k_{sc}). Metabolic losses of propofol are represented in k_{c0}, establishing the only means of absolutely reducing propofol concentration. This limitation presented a substantial challenge, the agent was required to learn that inaction (realized as a zero propofol infusion rate) was the only means of decreasing hypnosis and increasing BIS.

In prior clinical study, an infusion of propofol averaged a 2.7-minute time-to-peak effect in BIS (Schnider et al., 1998). Accordingly, our PK model was augmented with a fourth *effect site* compartment. The resulting transport coefficient, $k_{e0} = 0.17$, accounted for the delay between infusion and BIS effect, which included physiological delay (mixing, circulatory, etc.) and BIS measurement delay (Doufas et al., 2004). The effect-site compartment was assumed to possess negligible volume when modeling propofol distribution.

To model the hypnotic effect of propofol, a nonlinear pharmacodynamic model was developed using previously obtained data (Doufas et al., 2004). A three-layer perceptron

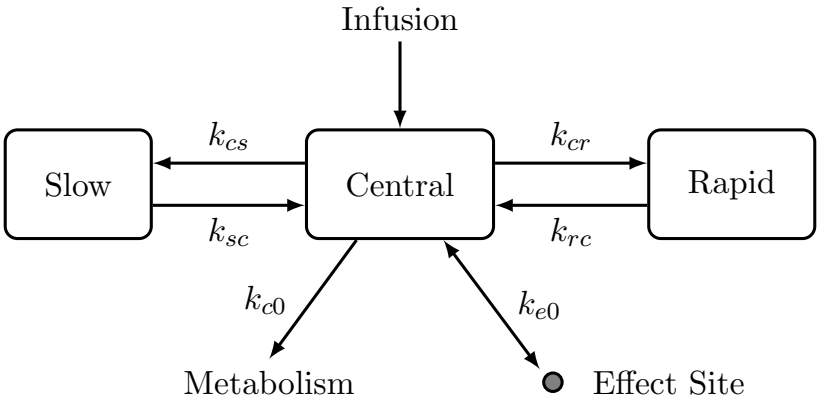


Figure 2: Schnider's pharmacokinetic model of propofol is based on the three-compartment mammalian model of pharmacokinetic action. Propofol is infused into the central volume through intravenous infusion. Concentration gradients then drive transport to the rapid and slow compartments, so named for their relative uptake rates. The site of propofol effect is modeled as an additional "virtual compartment" of infinitesimal volume in order to model observed delays between infusion and hypnotic effect.

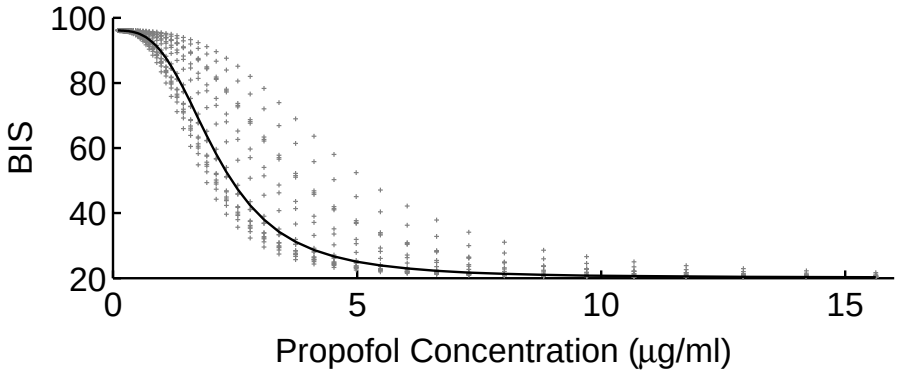


Figure 3: Doufas et al. observed the propofol/BIS response in eighteen young, healthy volunteers (2004). To model propofol pharmacodynamic effect for this study, a neural network function approximator was used to fit the median dose curve (highlighted here).

network was trained to associate arterial concentrations of propofol with observed BIS, thereby allowing the model to generally predict propofol effect from estimated effect site concentration. Figure 3 illustrates the observations of BIS and propofol concentration, as well as the median fit approximated by the neural network. The nonlinear relationship of propofol effect-site concentration to BIS is evident.

3.2.2 Knowledge Representation

During control, the RL agent is expected to observe the patient's state and then select the appropriate propofol dose using its control policy. To learn that optimal control policy, the agent accumulated its experience in *value functions*, mathematical descriptions of state utility commonly denoted as $V(s) \forall s \in \mathcal{S}$. Knowledge of V alone is not sufficient for optimal decision-making since this function only expresses the utility of an observed state. For control, it is also necessary to identify an action choice that can move the patient to more favorable conditions (or preserve existing favorable ones). In RL, the state-action value function, $Q(s,a) \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$, provides the necessary information for rational action selection. With Q, an RL agent can assess the utility of a patient state and then identify the proper infusion rate to achieve optimality for that state.

Initially, Q is unknown. The discovery of Q (learning) is accomplished through iterative function approximation. Consequently, Q must be represented in a form suitable for computational inspection and adjustment. Tables, decision trees, neural networks, and weighted polynomials have been used for this purpose in the literature. Of these, the uniformly discretized table is favored for its ease of implementation and mathematical robustness (Boyan and Moore, 1995; Baird, 1995).

In this study, state value (Q) functions were represented in a six-dimensional table. The agent's percepts, represented by the feature vector f, were mapped to a finite set of states $\mathcal{S}$ through uniform discretization. Recall that f consisted of six fuzzy state membership values (real numbers in the range $[0,1]$). To obtain a state observation $\mathcal{S}_i$ for a feature f_i, each dimension of the feature vector was partitioned into ten uniformly distributed bins, yielding a value function approximator with 10^6 entries. To permit identification of the optimal propofol infusion rate for all possible patient states, one such tabular function approximator was associated with each of the agent's actions.

3.2.3 Learning Algorithm

Watkins' Q-learning algorithm, a temporal-differencing learning method characterized by model-free, off-policy learning, was used to train the agent (Watkins, 1989). Q-learning is mathematically robust (Tsitsiklas and Van Roy, 1996; Dayan, 1992), and this robustness has contributed to the method's popularity in applied reinforcement learning. To accelerate learning, an improved form of Q-learning, called Q(λ), was applied in this study. This algorithm assimilates experience more quickly through extended temporal credit assignment. Whereas the one-step version considers only the previous action step when updating $Q(s_t, a_t)$, the improved version credits an historical chain of action selections. The "length" of this chain is governed by λ, which was set to the recommended value of 0.8 (Sutton and Barto, 1998).

3.2.4 Control Policy Identification

Watkins' Q-learning does not directly yield an optimal control policy. The algorithm only develops an approximation of the state-action value function, Q. However, the optimal control policy is trivial to determine once Q has been discovered. For each patient state s,

the optimal action choice $a^\star(s)$ may be expressed as:

$$a^\star(s) = \underset{a}{\operatorname{argmax}}\, Q\,(s,a)\ \forall a \in \mathcal{S}.$$

As a result, a state's optimal action may be represented as an integer number that indexes the ordered set of actions $\mathcal{A}$. Since the control policy identifies the best action choice for all patient states, the complete control policy may be represented as a six-dimensional table of these indices.

3.2.5 Training

Agent training consisted of a sequence of simulated hypnosis episodes using a standardized intraoperative patient prototype (male, 21 yr, 170 cm, 75 kg). To aid in learning a general association of propofol infusion and patient response, the patient's k_{e0} was randomly selected [$0.17 \pm 25\%$] at the beginning of each episode. This perturbation, of which the agent remained unaware, influenced the timing and magnitude of peak BIS effect.

To ensure sufficient exploration of the state-action space, each episode began with an *exploring start* in which a BIS target was randomly selected, and random propofol quantities were assigned to the three major PK compartments (Section 3.2.1). The agent was then permitted to interact with the patient and accumulate reinforcements for 1,000 consecutive action choices (5,000 simulated seconds). At the conclusion of an episode, a new one began with a newly randomized patient state.

Training began with a step-size parameter $\alpha = 0.2$, horizon parameter $\gamma = 0.69$, and an exploration parameter $\epsilon = 0.01$.[4] To assess the progress of learning, the sum of squared difference (SSD) was computed between intermediate control polices. When the SSD metric fell below a small threshold θ, α was halved, and learning resumed. This procedure continued until α measured 10^{-5} or less. In total, training required 5×10^7 episodes over approximately one week of CPU time on a contemporary desktop computer.

3.3 *In silico* Control Policy Evaluation

Prior to clinical application, the agent was evaluated in simulation to assess the fitness of the agent and its control policy. Although the agent was trained using an ideal simulated patient (fixed demographic parameters and near-ideal PK/PD characteristics), an actual surgical patient was not expected to present so favorably. Because intraoperative patients vary in height, weight, age, and gender (and other attributes), their PK/PD responses to propofol cannot be so neatly characterized.

To challenge the agent in a more realistic manner, the patient model illustrated in Figure 4 was modified to express patient-specific variation. The first point of individual variability was found in simple demographics. Schnider reported lean body mass, age, and gender to be significant covariates in propofol pharmacokinetic response (Schnider et al., 1998). Accordingly, the RL agent was tested on simulated patients with a range of demographic parameters. Since the Schnider model considers these parameters in its estimation of propofol distribution, demographic variation was not judged sufficient challenge for the agent.

4. For a more thorough discussion of these parameters and their import, see Moore et al. (2011b).

For additional challenge, the ideal PK and PD models were perturbed in ways mimicking the variation routinely observed in the operating room. A few quantitative (Röpcke et al., 2001b; Schnider et al., 1999, 1998; Bailey et al., 1996) and qualitative (Kearse Jr. et al., 1994; Ausems et al., 1986) descriptions of intraoperative patient variability may be found in the anesthesia literature. Taken collectively, evidence suggests that individual patient variation may be expressed as deviation in propofol pharmacokinetics (Gentilini et al., 2000; Schwilden et al., 1987) or pharmacodynamics (Struys et al., 2004, 2001).

In this study, individualized response in the simulated intraoperative patient was achieved with a *Patient Variability Model* (PVM), a mechanism for perturbing the patient's PK/PD in a manner removed from the agent's direct observation (illustrated in Figure 7). The PVM was implemented as two distinct components: one which affected the ideal pharmacokinetics (PK_{PVM}), and one which perturbed ideal pharmacodynamics (PD_{PVM}).

3.3.1 Pharmacokinetic Variation

The anesthesia literature provides evidence that patients commonly exhibit pharmacokinetic variation. Gepts observed: "When individuals are given identical doses per kg of body weight, large differences in pharmacological response may be seen" (Gepts, 1998, 10) and "Pharmacokinetic variability is much greater in sick compared with healthy people..." (Gepts, 1998, 11). The findings of Doufas et al. support those observations of variability. In a propofol pharmacokinetic study of 18 healthy volunteers, k_{e0} was determined to be 0.17 min^{-1} (range $[0.08, 0.25]$ min^{-1}) (Doufas et al., 2004). To model this source of patient variation, the PK_{PVM} block varied k_{e0} in conjunction with variation in patient demographics.

Figure 5 illustrates the effect of k_{e0} variation in simulated patients. A bolus of propofol was applied at $t = 0$ min and allowed to distribute under the Schnider pharmacokinetic model at the selected k_{e0} values. As shown, larger k_{e0} coefficients represented more "tightly coupled" systems in which propofol was transported to the effect site more readily, resulting in deeper hypnosis for a given dose. For emphasis, Figure 5 highlights the minimum hypnotic levels, as well as the times of their occurrence. While the time of peak effect varied by approximately 25 seconds, the range in peak effect varied by more than 20 points, a range that can span the clinically meaningful endpoints of light to deep hypnosis (as measured by BIS).

3.3.2 Pharmacodynamic Variation

Other sources of patient variation were better modeled as perturbations in propofol pharmacodynamics (i.e., effect, rather than distribution). For example, propofol sensitivity or tolerance may be modeled intuitively as a respective heightened or attenuated pharmacodynamic response to a given concentration of propofol. Exogenous factors, such as measurement noise and surgical stimuli, can not be reasonably expected to alter the pharmacokinetic distribution of propofol within the patient; however, these influences may directly alter the hypnotic action of the drug.

The role of the PD_{PVM} block was to model those factors best expressed as change in pharmacodynamics. The PD_{PVM} block accomplished this by decomposing pharmacodynamic variability into three classes: propofol sensitivity, intraoperative stimuli, and measurement

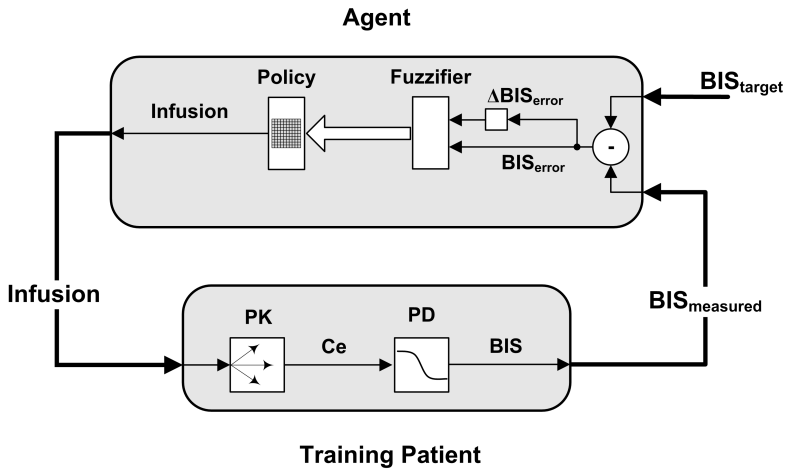


Figure 4: This figure illustrates the agent and its relationship to the simulated intraoperative patient used for training. The agent observed two external inputs, $\mathrm{BIS}_{\mathrm{target}}$ and $\mathrm{BIS}_{\mathrm{measured}}$, to compute the control error ($\mathrm{BIS}_{\mathrm{error}}$) and the change in control error over time ($\Delta\mathrm{BIS}_{\mathrm{error}}$). The intraoperative patient was modeled with near-ideal propofol PK/PD parameters.

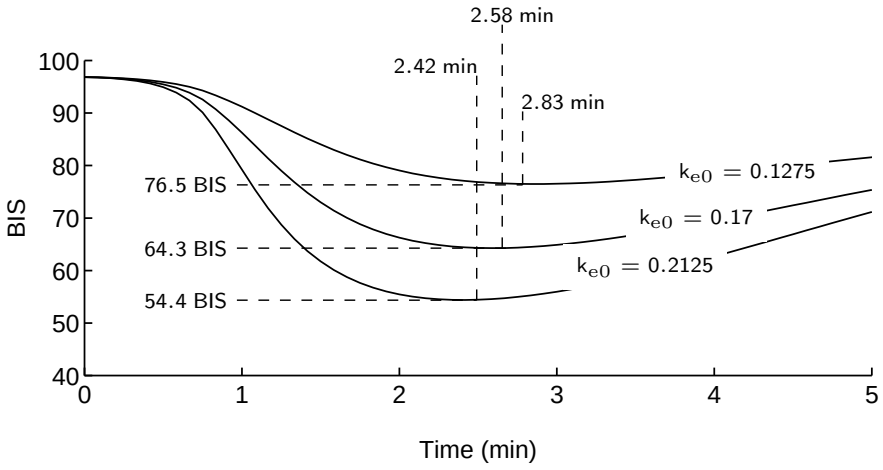


Figure 5: To demonstrate the variation associated with changes in k_{e0}, a bolus of propofol was delivered to a simulated patient, and distribution of propofol was modeled over time. For comparison, k_{e0} was selected at values of 0.17, 0.1275 (0.17-25%), and 0.2125 (0.17+25%). The points of peak BIS effect and their associated times are highlighted.

noise. The simulated patient's ideal BIS was then perturbed with a sum of time-dependent and independent combinations of these factors.

Modeling changes in propofol sensitivity begins as a straightforward process. In the tolerant patient, a given concentration of propofol may produce higher-than-expected BIS levels (more conscious than predicted). Conversely, lower-than-expected BIS levels may be observed in the patient with increased propofol sensitivity (less aware than predicted). In the evaluation of the RL agent, these differences were considered as a constant bias in pharmacodynamic effect. This time-independent parameter, denoted as $\Delta\text{BIS}^{\text{i}}_{\text{static}}$, was implemented as an additive factor, ≥ 0 in the resistant patient and ≤ 0 in the sensitive patient.

A credible model of propofol sensitivity should also consider exogenous sources of variation, one of which is noxious surgical stimuli. When studying patient variation, it is reasonable to conclude that some surgical procedures are more painful than others. For example, in a common heart procedure, such as the coronary artery bypass graft (CABG), the patient's chest is opened with an approximate six-inch incision, and the sternum is separated for access to the heart. Compare this procedure to the arthroscopic repair of a rotator cuff injury and its small 1-cm incisions. Intuitively, the degree of noxious stimulation in the CABG procedure is expected to exceed that of rotator cuff repair. Ausems et al. support this expectation in a report that found upper abdominal procedures required more analgesia than other smaller procedures (Ausems et al., 1986). Likewise, intraoperative stimuli were correlated to increased analgesic requirements in patients undergoing lower abdominal gynecologic, upper abdominal, and breast surgery. From these observations, as well others (Barvais et al., 1996), it is reasonable to conclude that some surgical procedures are inherently more noxious than others. Accordingly, the time-independent positive constant $\Delta\text{BIS}^{\text{s}}_{\text{static}}$ was used to denote this persistent noxious surgical stimulus.

Noxious stimulus may also be presented in a time-dependent manner. Absalom observes, "It is not always possible to predict when a surgeon will suddenly inflict a noxious stimulus on the patient..." (Absalom et al., 2002, 73). Ausems et al. reported that different intraoperative stimuli, including tracheal intubation, skin incision, and closure, required different levels of analgesia to maintain satisfactory anesthesia (Ausems et al., 1986). More recently, decreases in hypnotic level have been associated with surgical stimulation (Röpcke et al., 2001b), while increases in bispectral index have been correlated with skin incision (Kearse Jr. et al., 1994). Ausems et al. also observed that "single short-duration" stimuli, such as skin incision, required higher concentrations of opioid analgesic to ensure adequate anesthesia (Ausems et al., 1986). Given these observations, the short-duration surgical stimulus can reasonably be considered a transient perturbation in propofol pharmacodynamics that presents as a temporary decrease in hypnosis.

The effects of intraoperative stimuli are not limited to arousal events, those that decrease hypnotic effect. Röpcke found that concomitant administration of propofol and remifentanil (an opioid analgesic) resulted in lower than expected measurements of bispectral index in the intraoperative patient (Röpcke et al., 2001a). Whereas the noxious stimulus could be viewed as transient propofol tolerance, this synergistic drug interaction may present as temporarily heightened propofol sensitivity. These depressive events pose a unique challenge for hypnosis control since the agent cannot directly intervene and reduce the patient's propofol concentration.

During *in silicon* verification of the RL agent, irregular transient stimuli were presented to the agent to evaluate its ability to handle the dynamic conditions commonly found in the

intraoperative patient. To challenge the agent in an unpredictable manner, the duration, timing, and intensity of the short-duration stimuli were randomly chosen. In addition, the "direction" of challenge was randomized. A positive magnitude, which simulated an arousing event, was chosen with probability 0.8. Depressive events were chosen with probability 0.2. For this study, the time-dependent affect on patient pharmacodynamics was denoted as $\Delta\text{BIS}_{\text{dynamic}}(\text{t})$, where t indicated time dependence.

Finally, BIS is intrinsically noisy since the underlying EEG is a low-power signal requiring amplification for adequate measurement (as discussed previously). Prior study has modeled this noise as a stationary, normally-distributed signal ($\mu = 0$, $\sigma = 3$) (Struys et al., 2004). We modeled BIS measurement noise in accordance with this precedent.

In summary, the PVM modeled individual patient variability with changes in propofol pharmacokinetics and pharmacodynamics that remained hidden from agent observation. The PK_{PVM} component modeled changes in k_{e0}, while the PD_{PVM} block modeled changes in propofol sensitivity ($\Delta\text{BIS}_{\text{PVM}}$) as a sum of time-dependent and time-independent parameters (Moore et al., 2009). The cumulative PVM influence can thus be summarized as:

$$\Delta BIS_{static} = \Delta BIS^{i}_{static} + \Delta BIS^{s}_{static},$$

$$\Delta BIS_{PVM}(t) = \Delta BIS_{static} + \Delta BIS_{dynamic}(t) + \mathcal{N}(0,3),$$

$$BIS_{measured}(t) = BIS_{ideal}(t) + \Delta BIS_{PVM}(t).$$

3.4 Assessment of Agent Performance

The clinical study protocol included performance analysis of RL control under steady-state (maintenance of hypnosis) and non-steady-state conditions (induction of hypnosis and change in $\text{BIS}_{\text{target}}$). In the clinical practice of anesthesia, precise control has less value during non-steady-state periods. Conditions that a control engineer might consider unfavorable, like target overshoot, are expected during a manual induction as the clinician seeks to quickly achieve the desired target. Because the agent's principal goal of fine control is less relevant during induction, performance analysis of this interval has been omitted from this discussion.

3.4.1 Evaluation Population

To evaluate the agent's ability to provide well-controlled propofol hypnosis in a diverse population, a set of 1,000 individualized patients was generated *in silico*. Control performance was assessed over one episode of hypnosis for each of these individualized patients. In each episode, the agent was first presented a fully conscious patient and then tasked with achieving and maintaining propofol-induced hypnosis for 240 minutes, an interval that the clinical team considered representative. During the episode, BIS targets were randomly selected (without replacement) from the set {40,50,60}. Once selected, a target remained in effect for 80 minutes.

3.4.2 Maintenance Interval Identification

Although the induction interval is not addressed in this discussion, induction, along with BIS target change events, delimit the maintenance control intervals. The first maintenance

interval began with the completion of anesthetic induction. The induction period began when RL control was engaged to induce anesthesia in the conscious simulated patient (BIS $\approx$ 95). Induction continued until steady-state conditions, as defined by O'Hara et al., were observed at the selected target (1992). (The O'Hara metrics include T_{sp} = Time to Setpoint,T_{peak} = Time of Peak BIS, T_{sp} = Time to Steady State, and BIS_{peak} =$BIS_{measured}$ at T_{peak}.) This steady-state point, identified as T_{ss} in Figure 6, marked the beginning of the first maintenance control interval. The maintenance interval continued until the time of BIS target change, denoted as $T_{ss} + 80$ min, or ΔBIS_{target}.

The beginning of the next maintenance period was delineated similarly since the conditions at target transition resembled those at induction. After a step change in BIS_{target}, the agent acted to reestablish control and achieve steady-state conditions at the new target. Accurate identification of the new steady state was slightly complicated. High-to-low target changes (i.e., BIS_{target}=60 to BIS_{target}=40) directly compared to induction, while low-to-high changes (i.e., BIS_{target}=40 to BIS_{target}=50) resembled induction in an *inverted* sense. Once the new T_{ss} was achieved, the second maintenance control interval continued until the second target change.

The beginning of the third maintenance period was handled just as the second maintenance period. However, this control period was terminated by the end of automated control. Propofol infusion was discontinued, and the virtual patient was allowed to recover normal consciousness.

3.4.3 Performance Metrics

The steady-state control performance was evaluated using the four metrics of Varvel et al. (1992), which comprise the standard performance measures in closed-loop infusion control. These metrics build upon the instantaneous performance error (PE):

$$PE = \frac{BIS_{smoothed} - BIS_{target}}{BIS_{target}} \cdot 100. \tag{2}$$

The first metric, the median performance error (MDPE), indicates the control bias observed in a single patient and is computed as

$$MDPE_i = \text{median}\,(PE_{ij}) \qquad j = 1 \ldots N, \tag{3}$$

where i identifies a subject, and j iterates over the set of PE measurements for a subject. Median absolute performance (MDAPE) error reflects the accuracy of the controller in a subject:

$$MDAPE_i = \text{median}\,(|PE_{ij}|) \qquad j = 1 \ldots N. \tag{4}$$

Wobble measures the intra-subject variability in performance error:

$$Wobble_i = \text{median}\,(|PE_{ij} - MDPE_i|)\ j = 1 \ldots N. \tag{5}$$

Divergence is defined as the slope of the regression line computed through the observed MDAPE measurements. Positive values indicate an increasing difference in measured and target values; a negative divergence indicates more stable control.

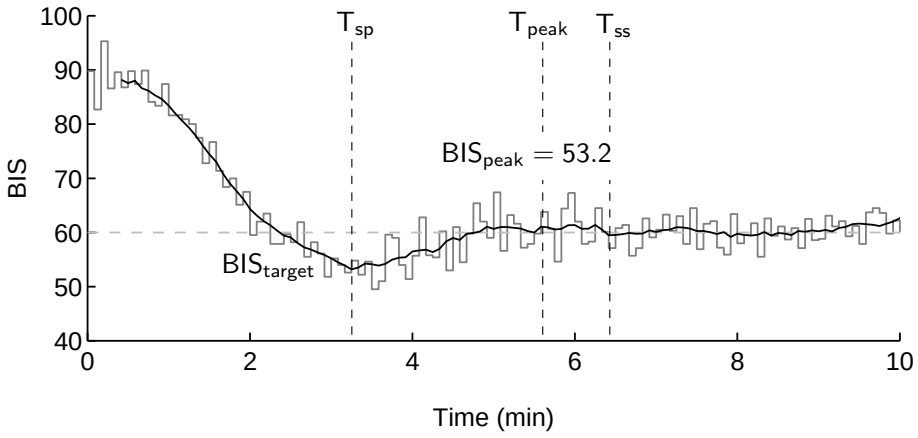


Figure 6: During analysis of control performance, the dynamic performance parameters T_{sp},T_{ss}, T_{peak}, and BIS_{peak} first reported by O'Hara et al. (1992) were programmatically identified to precisely delineate the maintenance control periods.

In addition to the Varvel metrics, contemporary studies of closed-loop anesthesia report the *Controlled* metric, the percentage of measurements in which the measured BIS was observed to be within ± 10 BIS (Struys et al., 2004) or ± 5 BIS (De Smet et al., 2008) of target. As an additional performance comparator, this study also reports the root-mean-square error (RMSE) computed for each maintenance control interval.

3.4.4 Acceptance Criteria

The anesthesia literature does not provide a definitive guideline for clinically suitable control of propofol-induced hypnosis, but a survey of three contemporary studies (De Smet et al., 2008; Struys et al., 2004; Absalom and Kenny, 2003) provides some reasonable performance goals (Table 2). These performance objectives should be interpreted carefully since specific values of these measures have not been correlated to favorable clinical outcomes. In other words, no study strongly indicates that an MDPE of 5% is x times better than an MDPE of 10%. In the absence of such data, we aimed for performances levels that surpassed reported values by reasonable margins.

3.4.5 Simulation Results

As indicated by comparison of observed performance and respective targets (Tables 2 and 3), the median values for all observed steady state parameters met their respective acceptance criteria. MDPE, MDAPE, Wobble, Divergence, and RMSE all presented values below the respective requirements. (Note the change in units in the Divergence measure.) Likewise, the Controlled metric was above its minimum threshold. These results suggested that the RL agent was suitable for evaluation in healthy volunteers. However, no definitive conclusion could be drawn since the accuracy of the Patient Variability Model was not verified prior

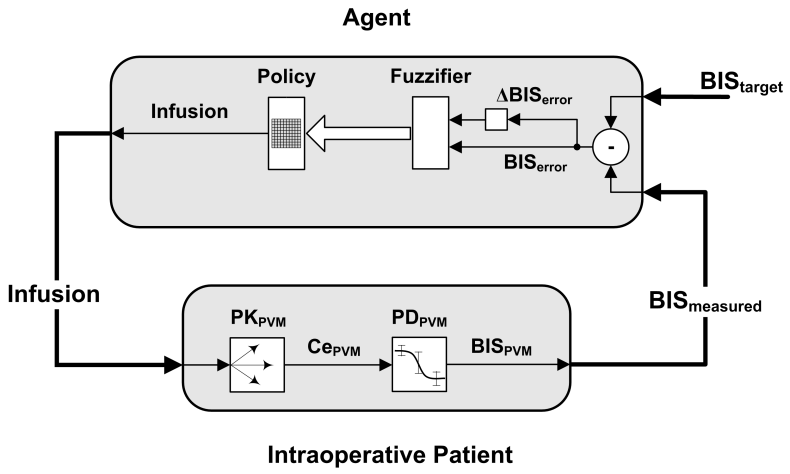


Figure 7: This figure illustrates the agent and its relationship to the simulated intraoperative patient used for evaluation. Like the training system, the agent relied on $\mathrm{BIS}_{\mathrm{target}}$ and $\mathrm{BIS}_{\mathrm{measured}}$ to compute control error, as well as change in control error. Unlike the training system, the intraoperative patient presented variable propofol PK/PD responses.

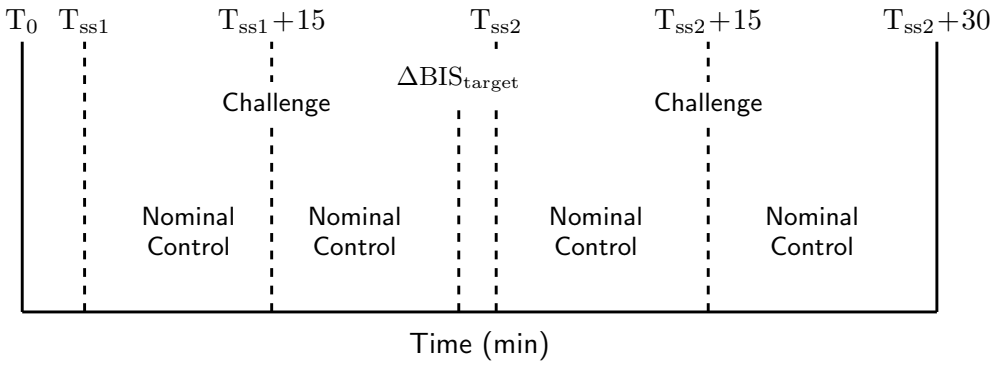


Figure 8: Induction (T_0) marked the beginning of the first BIS target evaluation period. Nominal control periods, as well as the surgical challenge, were scheduled in relation to $\mathrm{T}_{\mathrm{ss1}}$ (the time at which steady-state control was observed). Control continued to the maintenance interval's end at ($\mathrm{T}_{\mathrm{ss1}}$ + 30) min. At that time, a new $\mathrm{BIS}_{\mathrm{target}}$ was selected (labeled $\Delta\mathrm{BIS}_{\mathrm{target}}$ here) and a second, similar event schedule was observed. Recovery began at ($\mathrm{T}_{\mathrm{ss2}}$ + 30) min after automated control was discontinued.

to this study. After review of the simulation protocol and results, the principal clinical investigator granted approval for human study.

Parameter	Criterion
MDPE‡	± 5.0
MDAPE‡	7.5
Wobble‡	5.0
Divergence⋆	± 0.001
Controlled‡◇	80
RMSE§	5.0

‡(%), ⋆(%/hr), §(BIS)

◇ Percentage of time within ±5 BIS of target.

Table 2: Steady state performance acceptance criteria

Parameter	Observation	
MDPE‡	-0.17	(-0.50, 0.25)
MDAPE‡	3.33	(3.17, 3.50)
Wobble‡	3.30	(3.13, 3.50)
Divergence⋆	0.001	(-0.001, 0.003)
RMSE§	2.79	(2.58, 3.07)
Controlled‡◇	82.4	(80.6, 84.0)

median (IQR) ‡(%), ⋆(%/hr), §(BIS)

◇ Percentage of time within ±5 BIS of target.

Table 3: Simulated steady-state performance metrics

3.4.6 Clinical Application of RL Control

After IRB approval in the Stanford University School of Medicine, we recruited fifteen healthy (BMI $\leq$ 25 kg/m^2, 18–45 yr) volunteers. The clinical study was conducted in an operating room in the Stanford University Medical Center under informed consent. To facilitate clinical study, a custom data collection and control system, dubbed *Reagent*, was developed. The hypnosis control hardware consisted of a standard desktop computer, an A-2000 BIS monitor (Covidien, Mansfield, MA), and a Harvard Pump 33 dual syringe pump (Harvard Apparatus, Holliston, MA). The software consisted of a graphical user interface for clinician use, an embedded RL agent for propofol dosing, and various other modules for BIS monitor and syringe pump communication.

Volunteers fasted for at least six hours prior to the study and their vital signs were monitored according to the standards of the American Society of Anesthesiologists (ASA). After placement of the monitors, an intravenous catheter was inserted at the elbow for agent-directed propofol infusion. The study began when the anesthesiologist engaged RL control to achieve a randomly selected initial target (40 or 60). Once $\text{BIS}_{\text{target}}$ was achieved, the agent was permitted to regulate the level of hypnosis undisturbed for 15 minutes. A tetanic stimulus was then administered to the volunteer's thigh to simulate a noxious, destabilizing surgical event. Control was allowed to continue for an additional 15 minutes

(see Figure 8). At that time, the agent was directed to achieve the second BIS_{target}. Once the volunteer had stabilized at the second target, a similar procedure of maintenance and stimulus followed. Finally, automated hypnosis control was disengaged, and the volunteer was allowed to recover normal consciousness.

3.4.7 Performance Analysis

The agent's steady state control performance was assessed using the same procedures applied in the *in silico* evaluation. Automated tools identified induction, maintenance, and target change intervals. Maintenance intervals were then scored using the methods applied in the *in silico* performance analysis (Equations 2–5). The $BIS_{target} = 40$ and $BIS_{target} = 60$ control periods were evaluated independently and then in aggregate form.

The expected infrequency of BIS target change and relatively short duration between targets place the importance of transition control performance below maintenance performance; however, well-controlled behavior during BIS target change remains valued since the patient's need for hypnosis may vary over the course of the surgical procedure. In response, the dynamic O'Hara metrics are also presented in order to more thoroughly characterize the agent's control abilities. As discussed previously, these metrics were programmatically determined to identify maintenance intervals and were readily available.

4. Results

This section tabulates the study's observations. Results were grouped into three primary sets for analysis: Target 40 Maintenance, Target 60 Maintenance, and Aggregate Maintenance. The subordinate transition control metrics are reported, as well.

4.1 Volunteers

Fifteen healthy volunteers (11 males and 4 females) were recruited for the study of agent-guided propofol hypnosis. Table 4 presents the observed demographic parameters, and Table 5 summarizes those parameters. As the tables show, the volunteer population appeared to be young, healthy (ASA I), and predominantly male—characteristics reflecting the student population with ready access to study recruitment postings.

4.2 Target 40 Maintenance Control Metrics

Figure 9 graphically illustrates the $BIS_{measured}$ and $BIS_{predicted}$ values observed in each of target 40 episodes. The X-axes have been standardized to a 30-minute window. Note that the duration of an episode did not always equal 30 minutes due to timing differences between events hand-marked during the study and the more rigorous, post-study automated segmentation. The Y-axes have been standardized to a 60-BIS interval.

Some immediate observations can be made from Figure 9. The vertical black line indicates the time at which the tetanic stimulus was applied. Volunteers 5, 6, 7, 14, and 15 showed clearly distinguished arousal responses to noxious stimuli. The figure also highlights notable behavior in the predicted BIS. In most Target 40 episodes, $BIS_{predicted}$ demonstrated marked deviation from the observed BIS ($BIS_{measured}$). The degree of mis-prediction varied with volunteer, and prediction error appeared to vary within individual volunteers in a

ID	Gender	Age (yr)	Weight (kg)	Height (cm)	BMI (kg/m^2)
01	Male	21	84.0	183	25.1
02	Male	18	63.6	173	21.2
03	Male	20	69.0	178	21.8
04	Male	20	77.0	185	22.5
05	Male	18	61.4	175	20.1
06	Female	19	50.0	152	21.5
07	Male	22	77.3	178	24.4
08	Female	26	56.8	163	21.4
09	Female	19	61.4	163	23.1
10	Male	21	75.0	188	21.2
11	Male	19	70.5	180	21.7
12	Male	25	82.0	183	24.5
13	Male	20	61.4	175	20.0
14	Male	24	60.0	173	20.1
15	Female	19	59.1	168	20.9

Table 4: Human subject demographics

Age (yr)	Weight (kg)	Height (cm)	BMI (kg/m^2)
20.7 ± 2.5	72.2 ± 10.0	174.5 ± 9.6	22.0 ± 1.6
Mean $\pm$ SD		$n = 15$ ($n_{male} = 11, n_{female} = 4$)	

Table 5: Human subject demographic summary

time-dependent manner. No obvious systematic bias is evident; the model over-predicted in some volunteers but under-predicted in others.

Table 6 presents the observed Target 40 control metrics for each volunteer. The metrics are generally indicative of good control; however, two notable exceptions appear in the table. First, Volunteer 2's control metrics stand out as outliers. In this case, the volunteer exhibited strong bouts of coughing at Target 40. Although signs of illness were not obvious prior to study, the volunteer admitted to "having a cold" in a post-study interview. Likewise, Volunteer 11's study duration is anomalous; this Target 40 interval was abbreviated due to mis-configuration of the syringe pump after a fresh syringe was loaded.

4.3 Target 60 Maintenance Control Metrics

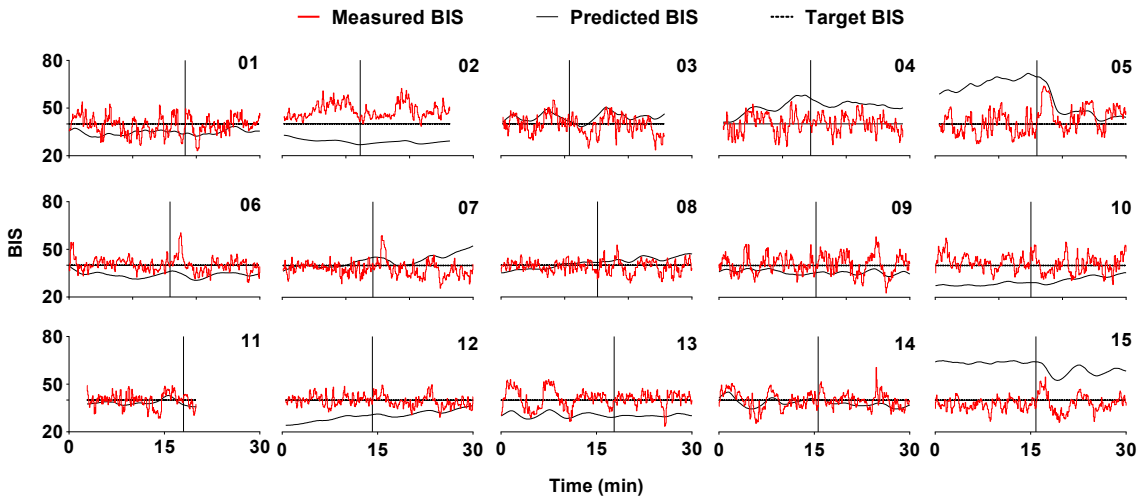
Figure 10 follows the format of Figure 9 in illustrating the BIS values observed in the Target 60 episodes. As before, the vertical black line indicates the point of tetanic stimulus. Volunteers 4, 5, 6, 7, 8, 13, 14, and 15 show responsive arousal behavior. Not unexpectedly, more volunteers exhibited obvious stimulus responses at this lighter hypnosis level. Table 7

ID	Target Order	Duration (min)	MDPE (%)	MDAPE (%)	Wobble (%)	Divergence (%/hr)	RMSE (BIS)	Controlled (%)
01	2	39.5	-3.5	11.0	10.5	-0.0002	6.5	56.8
02	1	26.1	17.2	17.2	7.2	0.0002	9.4	27.7
03	2	25.4	-0.1	11.2	11.1	0.0002	5.9	59.5
04	1	28.2	2.0	9.8	9.9	-0.0002	5.6	60.9
05	2	30.3	5.5	15.0	15.5	0.0004	8.4	42.2
06	1	36.3	1.8	6.8	7.0	-0.0001	4.9	77.6
07	2	30.2	-4.8	7.0	6.6	0.0004	5.1	69.8
08	1	30.2	-2.0	6.5	6.0	0.0002	3.9	81.3
09	2	30.8	2.0	10.2	10.5	0.0002	6.0	59.6
10	1	35.4	5.1	10.5	10.4	-0.0006	7.6	59.2
11	2	17.2	0.5	6.5	6.2	0.0004	4.5	72.0
12	1	29.6	-1.2	5.7	5.5	0.0001	3.6	84.3
13	1	32.9	2.2	8.8	7.8	-0.0005	5.4	67.2
14	1	30.6	-0.3	8.0	8.1	-0.0003	5.0	69.8
15	2	30.5	-6.0	8.8	7.0	0.0001	5.1	67.6

Table 6: Observed performance at BIS_{target}=40

ID	Target Order	Duration (min)	MDPE (%)	MDAPE (%)	Wobble (%)	Divergence (%/hr)	RMSE (BIS)	Controlled (%)
01	1	27.7	-1.0	1.4	1.5	0.0003	3.5	89.8
02	2	29.2	1.3	2.8	2.5	0.0001	3.1	90.3
03	1	24.6	0.7	3.2	3.2	0.0000	2.7	94.9
04	2	30.7	0.7	2.5	2.2	0.0000	2.2	97.8
05	1	30.6	-1.8	4.2	3.8	0.0003	4.6	77.4
06	2	32.2	0.6	4.0	3.8	0.0001	4.5	79.6
07	1	29.0	-0.7	2.8	2.7	0.0001	2.8	91.1
08	2	33.4	-0.7	4.5	4.7	0.0001	5.2	71.4
09	1	27.7	2.8	4.3	3.0	0.0000	4.0	80.8
10	2	30.0	0.2	3.5	3.5	0.0000	3.7	82.5
11	1	35.0	0.2	1.8	2.0	-0.0002	3.8	89.1
12	2	31.0	1.7	3.3	2.7	-0.0001	3.1	89.5
13	2	30.6	0.8	2.3	2.2	-0.0001	2.6	92.1
14	2	29.5	-2.3	6.7	5.7	0.0000	5.8	59.7
15	1	30.4	0.2	4.0	3.8	0.0001	4.2	79.5

Table 7: Observed performance at BIS_{target}=60



*Volunteer 11's episode was interrupted due to misconfiguration of the syringe pump.

Figure 9: The figure presents the observed BIS measurements at BIS_{target}=40. The X-axis represents a 30-minute window of time, and the Y-axis represents a 60-BIS interval. Each plot is labeled with the respective volunteer ID, and the vertical black line identifies the time of noxious stimulus. As shown, Volunteers 5, 6, 7, 14, and 15 demonstrated clear arousal responses to noxious stimulus.

presents the observed Target 60 control metrics for each volunteer. All of these metrics indicative of good hypnosis control. It should also be noted that these results are similar to those reported in the simulation (Table 3). Like the Target 40 observations, the Target 60 cases displayed varying degrees of PK/PD model mis-prediction. No systematic bias was detected.

4.4 Aggregate Maintenance Control Metrics

Table 8 presents each volunteer's aggregate control results. These metrics were computed by pooling the Target 40 and Target 60 observations to produce a set of global control measures. Table 9 summarizes the control metrics of the three groups (Target 40, Target 60, and Aggregate) with basic descriptive statistics. As shown, the mean aggregate control metrics exceed the desired performance levels presented in Table 2. The individual results were mixed: performance at BIS_{target}=60 met the desired goals by comfortable margins, while performance at BIS_{target}=40 narrowly missed desired levels in the Controlled and Wobble metrics.

4.5 Target Transition Metrics

Table 10 presents the observations obtained at changes from BIS Target 60 to Target 40. This high-to-low target change presented the most direct transition, and these observations were consistent with those observed in simulation. Table 11 presents the observations

ID	Duration (min)	MDPE (%)	MDAPE (%)	Wobble (%)	Divergence (%/hr)	RMSE (BIS)	Controlled (%)
01	67.2	-2.5	7.0	6.8	0.0000	5.3	70.5
02	55.3	8.8	9.6	4.7	0.0001	6.1	60.8
03	50.0	0.3	7.3	7.2	0.0001	4.3	76.9
04	59.0	1.3	6.0	5.9	-0.0001	3.8	80.1
05	60.9	1.8	9.6	9.6	0.0004	6.5	59.9
06	68.6	1.2	5.5	5.5	-0.0000	4.8	78.5
07	59.2	-2.8	5.0	4.7	0.0003	4.0	80.2
08	63.6	-1.3	5.4	5.3	0.0002	4.6	76.1
09	58.6	2.4	7.4	6.9	0.0001	5.1	69.6
10	65.4	2.9	7.3	7.2	-0.0003	5.8	69.9
11	52.2	0.3	3.4	3.4	-0.0000	4.0	83.4
12	60.6	0.2	4.5	4.1	0.0000	3.3	87.0
13	63.5	1.6	5.7	5.1	-0.0003	4.1	79.2
14	60.1	-1.3	7.3	6.9	-0.0002	5.4	64.9
15	60.9	-2.9	6.4	5.4	0.0001	4.6	73.5

Table 8: Observed aggregate maintenance performance

	BIS_{target} 40	BIS_{target} 60	Aggregate
Duration†	30.2 ± 5.2	30.1 ± 2.5	60.3 ± 5.1
MDPE‡	1.0 ± 5.6	-0.2 ± 1.2	0.4 ± 3.0
MDAPE‡	7.4 ± 3.5	2.8 ± 1.2	5.1 ± 1.7
Wobble‡	6.2 ± 2.6	2.6 ± 1.2	4.5 ± 1.5
Divergence*	< 0.001	< 0.001	< 0.001
RMSE§	4.5 ± 1.7	2.9 ± 1.1	3.7 ± 0.9
Controlled‡	79.0 (70.9, 89.0)	92.8 (83.3, 100.0)	85.5 (72.9, 88.5)
Mean ± SD Median (IQR)	n=15	†(min), ‡(%), *(%/hr), §(BIS)	

Table 9: Summary of observed maintenance performance

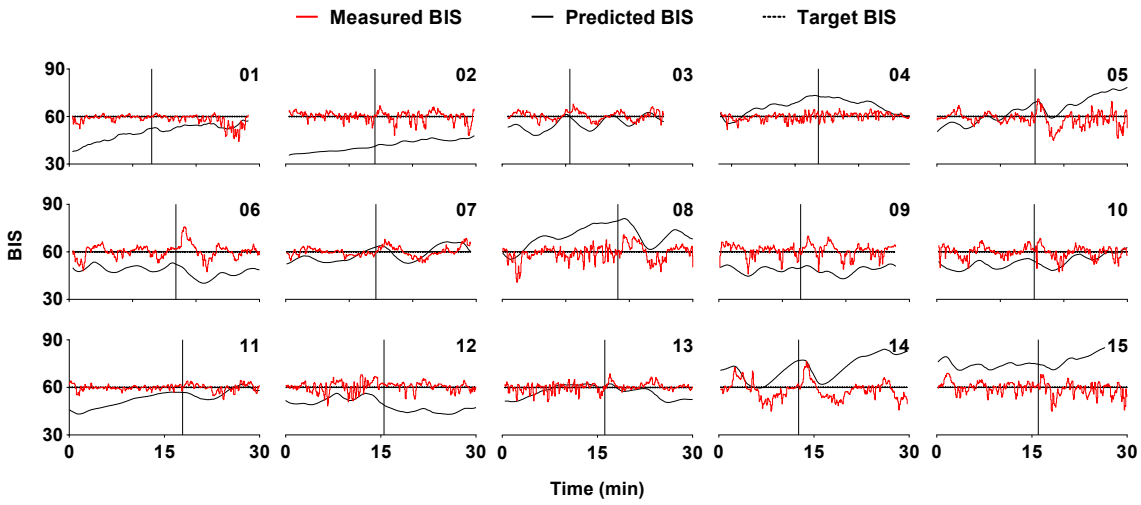


Figure 10: The figure presents the observed BIS measurements at BIS_{target}=60. The X-axis represents a 30-minute window of time, and the Y-axis represents a 60-BIS interval. Each plot is labeled with the respective volunteer ID, and the vertical black line identifies the time of noxious stimulus. As shown, Volunteers 3, 5, 6, 7, 8, 13, 14, and 15 demonstrated clear arousal responses to noxious stimulus.

Vol	T_{sp} (min)	T_{peak} (min)	T_{ss} (min)	BIS_{peak} (BIS)
1	2.04	4.62	7.62	11.13
3	2.46	4.21	4.79	6.49
5	1.47	2.30	3.13	14.87
7	2.84	4.42	5.92	5.13
9	2.61	3.61	4.44	10.88
11	3.24	3.74	4.24	3.14
15	2.67	4.76	6.34	6.12
	2.47 ± 0.57	3.95 ± 0.84	5.21 ± 1.51	8.25 ± 4.13
Mean ± SD				n=7

Table 10: Observed transition performance: Target 60 to 40

associated with changes from BIS Target 40 to Target 60. All three time measurements exceed those high-to-low transition observations by notable margins. These observations may be initially interpreted as evidence of the previously-discussed asymmetrical control influence.

Vol	T_{sp} (min)	T_{peak} (min)	T_{ss} (min)	BIS_{peak} (BIS)
2	7.13	8.13	8.13	1.29
4	11.25	12.09	14.25	14.22
6	11.19	11.94	16.61	13.06
8	11.39	12.36	14.86	15.91
10	10.79	12.54	14.20	13.66
12	10.84	12.17	14.26	10.36
13	6.58	7.58	7.58	2.10
14	8.59	8.92	9.59	0.38
	9.72 ± 1.99	10.71 ± 2.11	12.43 ± 3.45	8.87 ± 6.51

Mean ± SD n=8

Table 11: Observed transition performance: Target 40 to 60

5. Discussion

This section presents additional discussion highlighting the promising aspects of RL in closed-loop anesthesia control. The section also identifies some limitations of the clinical study and presents some opportunities for future research.

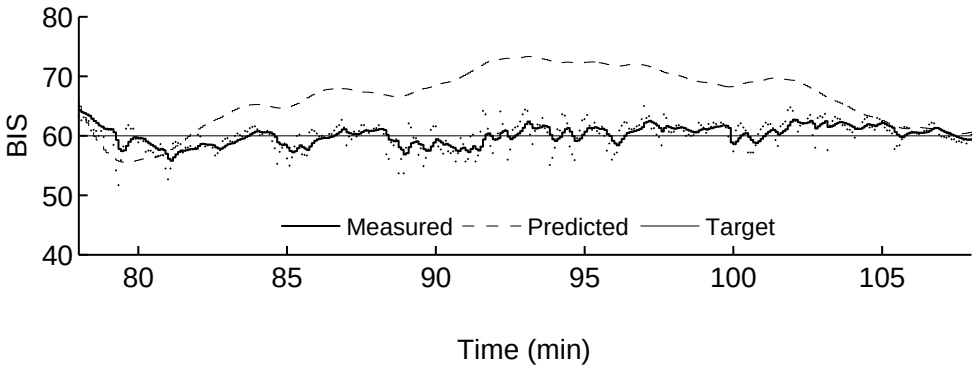
5.1 Clinically-acceptable Performance

The RL agent delivered propofol hypnosis in a manner consistent with well-controlled anesthesia, and control performance met or exceeded most performance targets. Control was considered accurate, as measured by MDPE, RMSE, and Controlled Percentage. The negligible Divergence values indicated that control was stable. The MDAPE and Wobble metrics were generally good, although an undesirable degree of oscillation was observed in some volunteers.

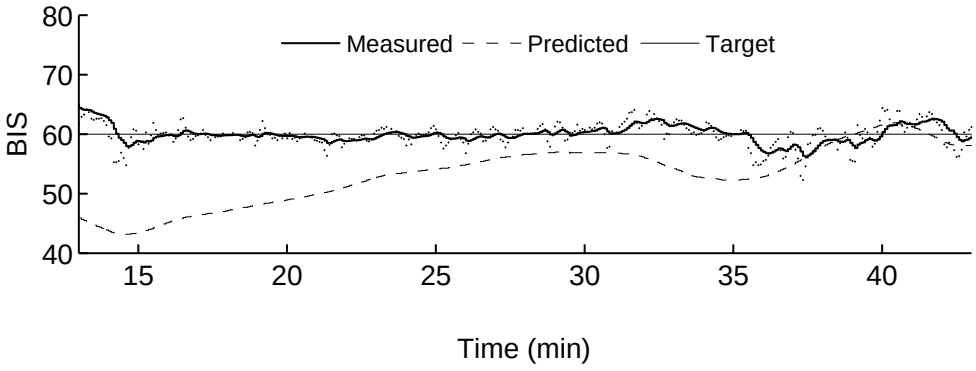
Furthermore, the agent demonstrated resistance to the disrupting tetanic stimulus. In cases where clear arousal response was observed, the agent reasserted control after the noxious event (Figures 9 and 10). Testing the boundaries of agent's capabilities in this manner is enlightening, but may be overly aggressive because propofol does not provide analgesia and cannot effectively manage pain as well as other anesthetics. In the intraoperative setting, propofol is commonly administered alongside an opioid analgesic, drugs that tend to suppress pain-induced arousal events, like those observed in this study.

5.2 Patient-specific Hypnosis

Figure 11 illustrates one favorable aspect of RL control: patient-specific hypnosis. During each study, the data collection system computed the predicted bispectral index as the agent controlled the volunteer's level of hypnosis. Using the volunteer's demographic data, the agent's action history, and the Schnider-Doufas PK/PD model, an estimate of propofol effect was computed on five-second intervals. By comparing predicted and observed BIS



(a) Indications of propofol sensitivity



(b) Indications of propofol tolerance

Figure 11: Although the RL agent was trained using a standardized patient prototype, the agent demonstrated good control in subjects deviating from the training model. In (a), the volunteer's observed BIS consistently measured below the predicted value, indicating the drug had greater effect than expected. In (b), the predicted BIS was consistently lower than measured, indicating the propofol dose yielded a higher BIS than anticipated.

values (Figure 11), the RL agent's ability to compensate for model mis-specification is highlighted. In the figure, Volunteer A demonstrated an apparent sensitivity to propofol. The observed hypnosis level consistently measured below the predicted value for most of the 30-minute period. Likewise, the RL agent compensated for an apparent propofol tolerance in Volunteer B. In this 30-minute period, the observed BIS consistently measured above the predicted value, indicating that the volunteer required more propofol than the population PK/PD models predicted. These observations suggest that the reinforcement learning process yielded a patient-specific control policy that may be applied to a general, variable population of volunteers with favorable results.

It should be noted that this RL implementation did not provide patient *adaptive* hypnosis, that is, no model parameters were adjusted during the control process. Rather, the agent exhaustively explored the discretized, bounded space of all possible $\text{BIS}_{\text{error}}$ and $\Delta\text{BIS}_{\text{error}}$ combinations during the exploring-start driven training. As a result, the agent formulated a control plan for all observable patient states, obviating a need for "on-the-fly" changes in its control policy. (Note that this style of exploration is limited to relatively small, discrete state spaces.) Adaptive closed-loop controllers in which model parameters are adjusted online have been studied in similar clinical control tasks (Ching et al., 2013; Shanechi et al., 2013; De Smet et al., 2008). Indeed, online reinforcement learning, a fixture in the intelligent systems literature, was a viable candidate for this application. However, a fixed-policy solution was preferred when seeking IRB approval for human study; likewise, the regulatory demands for any subsequent commercialization activities are lower when compared to an adaptive system. A convincing case is more easily made for a "safe and efficacious" system when the agent's control policy does not vary.

5.3 Limitations

The principal limitation of this study lies in its controlled nature. The human volunteers were healthy and mirrored those populations from which the PK/PD models were derived. Although the agent was challenged with credible intra-subject and inter-subject variation, it did not experience the full rigor of the intraoperative environment. In several instances of volunteer hypnosis, this limitation was realized with episodes of unanticipated natural sleep.

Because the bispectral index is an indirect measure of cortical activity, BIS is known to be affected by natural sleep (Nieuwenhuijs et al., 2002; Sleigh et al., 1999), as well as other conditions (including head trauma and hypothermia). In our study, conditions indicative of unanticipated natural sleep were observed after $\Delta\text{BIS}_{\text{target}}$ in some volunteers first receiving anesthesia at $\text{BIS}_{\text{target}}$=40. Figure 12 illustrates one instance in which the clinician directed a target change from 40 to 60 at $t \approx 64$ min. Since the desired target was higher than the subject's observed BIS, the agent correctly halted propofol infusion and waited for the volunteer to "recover" and awaken. Over the following ten minutes, the volunteer's predicted BIS rose as expected, but the volunteer's $\text{BIS}_{\text{measured}}$ remained near 40. $\text{BIS}_{\text{measured}}$ and $\text{BIS}_{\text{predicted}}$ increasingly diverged and the volunteer ultimately presented a predicted BIS near waking levels.

Given the ten-minute absence of propofol infusion and paradoxically low BIS measurements, the clinical team suspected the volunteer had transitioned from propofol-induced hypnosis to natural sleep. To continue the study and re-establish agent control, the clinician intervened with voice commands ("wake up", etc.) and a brief shoulder shake at $t \approx 74$ min. The volunteer's subsequent arousal was marked by a rapid convergence of $\text{BIS}_{\text{measured}}$ and $\text{BIS}_{\text{predicted}}$. As the volunteer awakened, the agent responded with propofol to reassert control at the new target of 60.

In summary, the volunteer shown in Figure 12 fell asleep shortly after the propofol infusion was interrupted—instead of waking as expected. In retrospect, this behavior was reasonable since our volunteer study lacked the usual surgical stimuli that would normally prevent natural sleep in the OR. Our volunteers were not subjected to persistent noxious

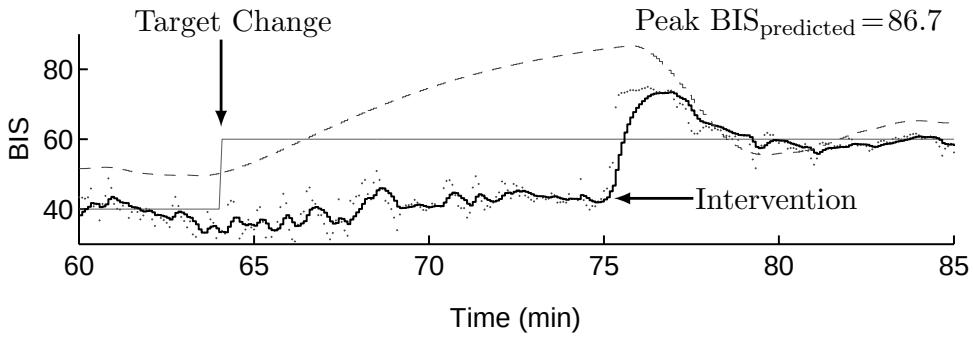


Figure 12: Our volunteer study modeled intraoperative hypnosis in a limited sense. Here, the agent stopped propofol infusion at the upward target change ($t \approx 64$ min). In response, the predicted BIS rose as the estimated effect-site concentration of propofol fell; however, $\text{BIS}_{\text{measured}}$ showed no corresponding increase. After ten minutes of no infusion, the subject failed to achieve a BIS > 40, although the predicted BIS approached waking levels. To continue study, the clinician intervened with rousing events (shoulder shake, voice commands, etc.). The volunteer responded immediately, and control resumed. This behavior was seen in several volunteers experiencing a low-to-high target change and was attributed to an unplanned transition to natural sleep after the target change.

stimulus, nor did they experience the usual bustling, noisy conditions of the operating suite. The ease in which the anesthetist roused the supposedly sedated volunteer, as well as the rate in which the measured BIS converged to the predicted, support the premise of natural sleep.

During the course of this study, the clinical team observed presumed natural sleep in 5 of 8 volunteers experiencing low-to-high target transition. (No sleep-like behavior was observed in volunteers undergoing high-to-low transitions.) Closer inspection of the low-to-high transition data (Table 11) reveals an apparent bimodal distribution in the corresponding time metrics. These observations appear to be naturally clustered in two well-differentiated groups: a *fast* transitioning group (non-sleepers) and a *slow* transitioning group (sleepers) that required an additional 3.5 min to emerge from Target 40 (Table 12). Exploratory parametric and non-parametric statistical tests suggest that two distinct groups do exist, but the small sample counts do not permit strong inferencing. The argument for sleep classification was bolstered when we confirmed *post-hoc* that all volunteers in the *slow* group required clinician intervention in order to complete the low-to-high transition. No *fast* group volunteers showed similar need. These findings suggest that presumed natural sleep occurred frequently in upward transitioning volunteers, thereby revealing a limitation of this study. Accordingly, our favorable results should be extrapolated to surgical patients in an appropriately limited fashion.

Fast Cluster				
	Vol	T_{sp} (min)	T_{peak} (min)	T_{ss} (min)
	2	7.13	8.13	8.13
	13	6.58	7.58	7.58
	14	8.59	8.92	9.59
Mean ± SD		7.43 ± 1.04	8.21 ± 0.67	8.43 ± 1.04
Median (IQR)		7.13 (1.01)	8.13 (0.67)	8.13 (1.01)
Slow Cluster				
	Vol	T_{sp} (min)	T_{peak} (min)	T_{ss} (min)
	4	11.25	12.09	14.25
	6	11.19	11.94	16.61
	8	11.39	12.36	14.86
	10	10.79	12.54	14.2
	12	10.84	12.17	14.26
Mean ± SD		11.09 ± 0.26	12.22 ± 0.23	14.84 ± 1.03
Median (IQR)		11.19 (0.41)	12.17 (0.27)	14.26 (0.61)

Table 12: Cluster Analysis of Target 40 to 60 Timed Metrics

5.4 Future Directions

Given the favorable performance in both simulation and healthy human volunteers, it seems reasonable to evaluate the agent in a study of actual surgical patients to more completely assess the clinical utility of RL control. Evaluation under the full rigor of the intraoperative environment, along with varying conditions of patient health, should provide further insight into RL's applicability. It is important to note that the studied RL agent does not directly represent a closed-loop drug delivery system suitable for general clinical use. For example, the current iteration of the agent is not equipped to reliably manage a prolonged open-loop condition due to BIS input failure. As such, it is more appropriate to consider the agent to be one player in a greater, more robust system.

The agent's aptitude for managing propofol response deviating from the training model (i.e., unexpected volunteer tolerance or sensitivity to propofol) is also cause for additional study. Like other PK/PD models, the Schnider PK model and the Doufas PD models characterize propofol response in a narrow, idealized population. Some poorly modeled populations, such as the critically-ill or morbidly obese, gain the most from patient-specific drug administration.

Finally, it should be noted that the application of reinforcement learning to medicine is not limited to depth-of-anesthesia management. Other potential applications exist, such

as neuromuscular blockade, mechanical ventilation, and management of cardiovascular parameters, including heart rate, blood pressure, and cardiac output.

5.5 Improvements

This study demonstrates the feasibility of RL hypnosis control, but it cannot yet be positioned as the optimal solution to the problem. Some areas of improvement are readily identifiable. For example, most of the metrics indicate that the agent controlled hypnosis more proficiently at $\mathrm{BIS}_{\mathrm{target}} = 60$. The authors theorize that the performance difference can be attributed to the apparent "mass" of a heavily-dosed patient. Patients at deeper levels of hypnosis accumulate higher concentrations of propofol, and the sigmoidal BIS response approaches saturation at levels near 4 μg/ml (Figure 3). Thus, an ever-increasing amount of propofol is required to meaningfully change the observed BIS in this region of the dose response curve. Likewise, the propofol-saturated patient responds to the zero infusion rate more slowly as peripheral tissue reservoirs continue to dump propofol into the patient's bloodstream well after the agent has discontinued infusion. These factors muddle the agent's interpretation of its actions, impairing its ability to regulate the patient's BIS level. In the following discussion, we suggest a few possible approaches to improve control.

We anticipate that the non-linearity illustrated in Figure 3 can be handled more effectively if the agent considers the current BIS measurement as an input. This additional percept provides a cue to handle the gross slope changes occurring in the ranges [0,1] μg, [1,4] μg, and [4,15] μg and should improve control at deeper levels of hypnosis.

An improvement may also be realized if the agent's goals can be modified to better reflect clinical practice. In general, control engineers are rightly concerned with achieving target setpoints with limited adverse behaviors, such as overshoot and ringing. Few anesthetists are control engineers, and few surgeons would appreciate the agent's observed $\tilde{1}$2.5-min induction of anesthesia. The agent's current reward strategy (Equation 1) discourages overshoot and promotes a "soft landing" on target; however, the clinician takes a different approach. A bolus of propofol is given, the patient is quickly rendered unconscious, and then the anesthetist manages any overshoot as needed. In other words, the goal of anesthetic induction differs fundamentally from the goal of anesthetic maintenance. Indeed, the goals oppose one another in time and control accuracy. A more effective solution might involve two independent, cooperative agents in which one agent is used for induction, and the other for maintenance.

The RL agent might also be implemented more effectively. In pilot studies, undesirable oscillation in the volunteer's BIS was occasionally observed. The authors theorized that the software filters used to attenuate BIS measurement noise compounded the 2.5-min lag in propofol effect, causing the agent to "chase" the target in an oscillatory fashion. To counter this noise without exacerbating lag, fuzzy state classifiers replaced aggressive smoothing so that the agent might better classify the volunteer's hypnotic state. The fuzzy classifiers reduced the significance of transient fluctuation in the $\mathrm{BIS}_{\mathrm{error}}$ and $\Delta\mathrm{BIS}_{\mathrm{error}}$ signals, thereby improving control performance so that human study could proceed as planned. Note that the fuzzy classifiers were selected without a comprehensive survey of filtering techniques. Clinical trials are expensive, challenging affairs not amenable to interruption once begun.

Fortunately, state generalization methods, like fuzzy classifiers, are well-represented in the RL literature,[5] Sparse coding (Sutton, 1996) and neural networks (Tesauro, 1992) are recognized methods for state aggregation in RL. Perhaps more fittingly, state generalization can be "rolled into" the Q-learning algorithm; fuzzy Q-learning, like that reported by Bonarini et al. (2009), Glorennec and Jouffe (1997) and Berenji and Kehdkar (1992), as well as delayed Q-learning (Chapman and Kaelbling, 1991), appear to be logical next steps in the evolution of this research.

It should also be noted that the agent was implemented as a discounted, infinite-horizon task ($\gamma < 1$). As mentioned previously, the closed-loop hypnosis task lacked an explicit goal state since the agent was expected to minimize control error for an undetermined duration. Alternatives exist for reinforcement learning in such infinite-horizon problems. Techniques that maximize returns over a window of time, like R-learning (Mahadevan, 1996), may be viable candidates for improving control performance.

Finally, when RL has been applied to real-world control tasks, the problem is usually modeled as an Markov Decision Process (MDP). This approach assumes complete observability of system states and influences that govern transitions among those states. In reality, full observability can be reasonably expected only in toy problems. Fortunately, hidden influences may be ignored without great consequence in many applications, leaving unadorned MDPs sufficient for the control task. However, closed-loop control of propofol hypnosis is a textbook example of a partially observable control process (Russel and Norvig, 2002). The task relies on an imperfect measurement (the bispectral index of the EEG) of a poorly-defined quantity (patient consciousness). Therefore, we believe that techniques used to solve Partially Observable Markov Decision Processes (POMDPs) (Kaelbling et al., 1998) are relevant in future studies.

6. Conclusion

The RL agent demonstrated clinically-suitable performance in the closed-loop control of propofol-induced hypnosis in healthy human volunteers. In doing so, the agent provided generalizing control that compensated for varying degrees of intra-subject and inter-subject variation in propofol effect, suggesting that RL control can improve propofol delivery in the general surgical population, as well as populations lacking good PK/PD models. Furthermore, RL's success in this clinical control task establishes precedence and positions the method as a viable candidate for solving other challenging clinical problems. Yet, as promising as these results appear, no strong conclusions regarding RL's place in closed-loop anesthesia can be made until similar results are observed under actual intraoperative conditions.

Acknowledgments

The clinical portion of this study was funded by the Department of Anesthesiology, Perioperative and Pain Management, Stanford University School of Medicine; the technical

5. More traditional predictive filtering techniques, like the Kalman filter, remain viable candidates for state generalization but are not discussed here.

aspects were funded by the authors. The authors would like to thank the Stanford University School of Medicine operating room staff for their support, as well as Aspect Medical (now Covidien) for providing an A-2000 BIS monitor.

References

A R Absalom and G N C Kenny. Closed-loop control of propofol anaesthesia using Bispectral Index^{TM}: Performance assessment in patients receiving computer-controlled propofol and manually controlled remifentanil infusions for minor surgery. *Brit J Anaesth*, 90(6):737–41, 2003.

A R Absalom, N Sutcliffe, and G N C Kenny. Closed-loop control of anesthesia using Bispectral Index^{TM}: Performance assessment in patients undergoing major orthopedic surgery under combined general and regional anesthesia. *Anesthesiology*, 96(1):67–73, Jan 2002.

M E Ausems, C C Hug, Jr, D R Stanski, and A G Burm. Plasma concentrations of alfentanil required to supplement nitrous oxide anesthesia for general surgery. *Anesthesiology*, 65 (4):362–73, Oct 1986.

M S Avidan, L Zhang, B A Burnside, K J Finkel, A C Searleman, J A Selvidge, L Saager, M S Turner, S Rao, M Bottros, C Hantler, E Jacobsohn, and A S Evers. Anesthesia awareness and the bispectral index. *New Eng J Med*, 11(358):1097–1108, Mar 2008.

J M Bailey, C T Mora, and S L Shafer. Pharmacokinetics of propofol in adult patients undergoing coronory revascularization. *Anesthesiology*, 84:1288–97, 1996.

L Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proc. 12th International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann, 1995.

L Barvais, I Rausin, J B Glen, S C Hunter, D D'Hulster, F Cantraine, and A d'Hollander. Administration of propofol by target-controlled infusion in patients undergoing coronary artery surgery. *J Cardiothorac Vasc Anesth*, 10(7):877–83, Dec 1996.

H R Berenji and P Kehdkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Transactions on Neural Networks*, 3(5):724–740, 1992.

M J Bloom, A Bekker, C V Seshagiri, and S D Greenwald. Changes in BIS variability reflect changes in remifentanil infusion during spinal surgery. Presented at the American Society of Anesthesiologists Annual Meeting, Oct 2008.

A Bonarini, A Lazaric, F Montrone, and M Restelli. Reinforcement distribution in fuzzy Q-learning. *Fuzzy Sets and Systems*, 160(10):1420–1443, 2009.

J Boyan and A W Moore. Generalization in reinforcement learning: Safely approximating the value function. In G Tesauro, D S Touretzky, and T K Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 369–376, Cambridge, MA, 1995. The MIT Press.

E Brown, R Lydic, and N Schiff. General anesthesia, sleep, and coma. *N Engl J Med*, 363 (27):2638–50, Dec 2010.

A Carregal, A Lorenzo, J A Taboada, and J L Barreiro. Intraoperative control of mean arterial pressure and heart rate with alfentanyl with fuzzy logic. *Rev Esp Anestesiol Reanim*, 47(3):108–113, Mar 2000.

D Chapman and L P Kaelbling. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991.

S Ching, B M Westover, M Liberman, J J Chemali, J Kenny, K Solt, P L Purdon, and E N Brown. Real-time closed-loop control in a rodent model of medically induced coma using burst suppression. *Anesthesiology*, 119(4):848–860, Oct 2013.

A Dahaba. Different conditions that could result in the bispectral index indicating an incorrect hypnotic state. *Anesth Analg*, 101(3):765–73, Sep 2005.

P Dayan. The convergence of TD(λ) for general λ. *Machine Learning*, 8:341–362, 1992.

T De Smet, M M R F Struys, M M Neckebroek, K Van den Hauwe, S Bonte, and E P Mortier. The accuracy and clinical feasibility of a new Bayesian-based closed-loop control system for propofol administration using the bispectral index as a controlled variable. *Anesth Analg*, 107:1200–1210, 2008.

A G Doufas, M Bakhshandeh, A R Bjorksten, S L Shafer, and D I Sessler. Induction speed is not a determinant of propofol pharmacodynamics. *Anesthesiology*, 101:1112–21, 2004.

D Ernst, G B Stan, J Goncalves, and L Wehenkel. Clinical data based optimal STI strategies for HIV; a reinforcement learning approach. In *Machine Learning Conference of Belgium and The Netherlands (Benelearn)*, pages 65–72, 2006.

D Ernst, M Glavic, F Capitanescu, and L Wehenkel. Reinforcement learning versus model predictive control: A comparison on a power system problem. *Trans Sys Man Cyber Part B*, 39(2):517–529, 2009. ISSN 1083-4419.

V Esmaeili, A Assareh, M B Shamsollahi, M H Moradi, and N M Arefian. Estimating the depth of anesthesia using fuzzy soft computation applied to EEG features. *Intell Data Anal*, 12(4):393–407, 2008.

S P Fitzgibbon, D M Powers, K J Pope, and C R Clark. Removal of EEG noise and artifact using blind source separation. *J Clin Neurophysiol*, 24(3):232–43, Jun 2007.

A E Gaweda, M K Muezzinoglu, A A Jacobs, G R Aronoff, and M E Brier. Model predictive control with reinforcement learning for drug delivery in renal anemia management. *Conf Proc IEEE Eng Med Biol Soc*, 1:5177–80, 2006.

A Gentilini, C Frei, A H Glattfelder, M Morari, and T Schnider. Identification and targeting policies for computer controlled infusion pumps. *Crit Rev Biomed Eng*, 28(1&2):179–185, 2000.

E Gepts. Pharmacokinetic concepts for TCI anaesthesia. *Anaesthesia*, 53(Suppl 1):4–12, Apr 1998.

P S Glass, M Bloom, L Kearse, C Rosow, P Sebel, and P Manberg. Bispectral analysis measures sedation and memory effects of propofol, midazolam, isoflurane, and alfentanil in healthy volunteers. *Anesthesiology*, 86(4):836–847, Apr 1997.

P Y Glorennec and L Jouffe. Fuzzy Q-learning. In *Proceedings of Fuzz-IEEE'97, Sixth International Conference on Fuzzy Systems*, volume 3, pages 659–662, 1997.

S D Greenwald and C E Rosow. BIS and EMG variability increase before somatic responses during surgery. Presented at the American Society of Anesthesiologists Annual Meeting, Oct 2006.

A Guez, R D Vincent, M Avoli, and J Pineau. Adaptive treatment of epilepsy via batch-mode reinforcement learning. In *IAAI'08: Proceedings of the 20th national conference on Innovative Applications of Artificial Intelligence*, pages 1671–1678. AAAI Press, 2008. ISBN 978-1-57735-368-3.

V Gullapalli. Learning control under extreme uncertainty. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 327–334. Morgan Kaufmann, San Mateo, CA, 1993. URL `citeseer.nj.nec.com/312133.html`.

J O Hahn, G A Dumont, and J M Ansermino. Closed-loop anesthetic drug concentration estimation using clinical-effect feedback. *IEEE Trans Biomed Eng*, 58(1):3–6, Jan 2011.

T M Hemmerling, S Charabati, C Zaouter, C Minardi, and P A Mathieu. A randomized controlled trial demonstrates that a novel closed-loop propofol system performs better hypnosis control than manual administration. *Can J Anaesth*, 57(8):725–735, Aug 2010.

C Hu, W S Lovejoy, and S L Shafer. Comparison of some control strategies for three-compartment PK/PD models. *Journal of Pharmacokinetics and Biopharmaceutics*, 22 (6):525–550, 1994.

L P Kaelbling, M L Littman, and A W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

L P Kaelbling, M L Littman, and A R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

L A Kearse Jr., P Manberg, N Chamoun, F deBros, and A Zaslavsky. Bispectral analysis of the electroencephalogram correlates with patient movement to skin incision during propofol/nitrous oxide anesthesia. *Anesthesiology*, 81(6):1365–70, Dec 1994.

G N C Kenny and H Mantzaridis. Closed-loop control of propofol anaesthesia. *Brit J Anaesth*, 83(2):223–8, Aug 1999.

K Leslie, A R Absalom, and G N C Kenny. Closed loop control of sedation for colonoscopy using the Bispectral Index. *Anaesthesia*, 57(7):690–709, Jul 2002.

M Lindholm, S Träff, F Granath, S D Greenwald, A Ekbom, C Lennmarken, and R H Sandin. Mortality within 2 years after surgery in relation to low intraoperative bispectral index values and preexisting malignant disease. *Anesth Analg*, 108(2):508–512, Feb 2009.

M Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann.

N Liu, T Chazot, A Genty, A Landais, A Restoux, K McGee, P A Laloë, B Trillat, L Barvais, and M Fischler. Titration of propofol for anesthetic induction and maintenance guided by the bispectral index: Closed-loop versus manual control: A prospective, randomized, multicenter study. *Anesthesiology*, 104(4):686–695, April 2006.

N Liu, M Le Guen, F Benabbes-Lambert, T Chazot, B Trillat, D I Sessler, and M Fischler. Feasibility of closed-loop titration of propofol and remifentanil guided by the spectral M-Entropy monitor. *Anesthesiology*, 116(2):286–295, Feb 2012.

N Liu, O Pruszkowski, J E Leroy, T Chazot, B Trillat, A Colchen, F Gonin, and M Fischler. Automatic administration of propofol and remifentanil guided by the bispectral index during rigid bronchoscopic procedures: A randomized trial. *Can J Anaesth*, 60(9):881–887, Sep 2013.

S Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22(1–3):159–195, 1996.

J D Martín-Guerrero, F Gomez, E Soria-Olivas, J Schmidhuber, M Climente-Martí, and N V Jiménez-Torres. A reinforcement learning approach for individualizing erythropoietin dosages in hemodialysis patients. *Expert Syst Appl*, 36(6):9737–9742, Aug 2009.

D M Mathews, L Clark, J Johansen, E Matute, and C V Seshagiri. Increases in electroencephalogram and electromyogram variability are associated with an increased incidence of intraoperative somatic response. *Anesth Analg*, 114(4):759–70, Apr 2012.

B L Moore. Intelligent control of closed-loop sedation in simulated ICU patients. Master's thesis, Texas Tech University, 2003.

B L Moore, E D Sinzinger, T M Quasny, and L D Pyeatt. Intelligent control of closed-loop sedation in simulated ICU patients. In *FLAIRS 2004*. AAAI Press, 2004.

B L Moore, L D Pyeatt, and A G Doufas. Fuzzy control for closed-loop, patient-specific hypnosis in intraoperative patients: A simulation study. In *Conf Proc IEEE Eng Med Biol Soc*, volume 1, 2009.

B L Moore, A G Doufas, and L D Pyeatt. Reinforcement learning: A novel method for optimal control of propofol-induced hypnosis. *Anesth Analg*, 112(2):360–367, Feb 2011a.

B L Moore, T M Quasny, and A G Doufas. Reinforcement learning versus proportional-integral-derivative control of hypnosis in a simulated intraoperative patient. *Anesth Analg*, 112(2):350–359, Feb 2011b.

E P Mortier, M M R F Struys, T De Smet, Y D I Versichelen, and G Rolly. Closed-loop controlled administration of propofol using bispectral analysis. *Anaesthesia*, 53(8): 749–754, Aug 1998.

P Myles, K Leslie, J McNeil, A Forbes, and M Chan. Bispectral index monitoring to prevent awareness during anaesthesia: the B-Aware randomised controlled trial. *Lancet*, 363(9423):1757–1763, 2000.

D Nieuwenhuijs, E L Coleman, N J Douglas, G B Drummond, and A Dahan. Bispectral index values and spectral edge frequency at different stages of physiologic sleep. *Anesth Analg*, 94(1):125–129, Jan 2002.

D A O'Hara, D K Bogen, and A Noordergraaf. The use of computers for controlling the delivery of anesthesia. *Anesthesiology*, 77(3):563–81, Sep 1992.

K T Olkkola, H Schwilden, and C Apffelstaedt. Model-based adaptive closed-loop feedback control of atracurium-induced neuromuscular blockade. *Acta Anaesth Scand*, 35(5):420–3, Jul 1991.

S Omatu, M Khalid, and R Yusof. *Neuro-control and its Applications*, chapter 4, pages 152–160. Advances in Industrial Control. Springer, 1996.

S Pilge, R Zanner, G Schneider, J Blum, M Kreuzer, and E F Kochs. Analysis of cerebral state, bispectral, and narcotrend indices. *Anesthesiology*, 104(3):488–494, Mar 2006.

I J Rampil. A primer for EEG signal processing in anesthesia. *Anesthesiology*, 89(4): 980–1002, Oct 1997.

M Renna, T Wigmore, A Mofeez, and C Gillbe. Biasing effect of the electromyogram on BIS: A controlled study during high-dose fentanyl induction. *J Clin Monit Comput*, 17 (6):377–81, Aug 2002.

A E Rigby-Jones and J R Sneyd. Pharmacokinetics and pharmacodynamics: Is there anything new? *Anaesthesia*, 67(1):5–11, Jan 2012.

H Röpcke, M Knen-Bergmann, M Cuhls, T Bouillon, and A Hoeft. Propofol and remifentanil pharmacodynamic interaction during orthopedic surgical procedures as measured by effects on bispectral index. *J Clin Anesth*, 13(3):198–207, May 2001a.

H Röpcke, B Rehberg, M Koenen-Bergmann, T Bouillon, J Bruhn, and A Hoeft. Surgical stimulation shifts EEG concentration-response relationship of desflurane. *Anesthesiology*, 94(3):255–113, Mar 2001b.

S Russel and P Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2002.

F Sahba, H R Tizhoosh, and M M A Salama. Application of reinforcement learning for segmentation of transrectal ultrasound images. *BMC Med Imaging*, 8(8), 2008.

T Sakai, A Matsuki, P F White, and A H Giesecke. Use of an EEG-bispectral closed-loop delivery system for administering propofol. *Acta Anesth Scand*, 44:1007–1010, 2000.

R H Sandin, G Enlund, P Samuelsson, and C Lennmarken. Awareness during anaesthesia: A prospective case study. *Lancet*, 355(9205):707–711, 2000.

J Schaublin, M Derighetti, P Feigenwinter, S Petersen-Felix, and A M Zbinden. Fuzzy logic control of mechanical ventilation during anaesthesia. *Brit J Anaesth*, 77(5):636–41, Nov 1996.

T Schnider, C F Minto, P L Gambus, C Andresen, D B Goodale, S L Shafer, and E J Youngs. The influence of method of administration and covariates on the pharmacokinetics of propofol in adult volunteers. *Anesthesiology*, 88(5):1170–1182, May 1998.

T W Schnider, C F Minto, S L Shafer, P L Gambus, C Andresen, D B Goodale, and E J Youngs. The influence of age on propofol pharmacodynamics. *Anesthesiology*, 90(6): 1502–16, Jun 1999.

H Schwilden, J Schüttler, and H Stoeckel. Closed-loop feedback control of methohexital anesthesia by quantitative EEG analysis in humans. *Anesthesiology*, 67(3):341–7, Sep 1987.

P S Sebel, T A Bowdle, M M Ghoneim, I J Rampil, R E Padilla, T J Gan, and K B Domino. The incidence of awareness during anesthesia: A multicenter United States study. *Anesth Analg*, 99:833–839, 2004.

F S Servin. TCI compared with manually controlled infusion of propofol: A multicentre study. *Anaesthesia*, 53(Suppl 1):82–86, Apr 1998.

M M Shanechi, J J Chemali, M Liberman M, K Solt, and E N Brown. A brain-machine interface for control of medically-induced coma. *PLOS Compu Biol*, 9(10):1–17, Oct 2013.

J C Sigl and N G Chamoun. An introduction to bispectral analyis for the electroencephalogram. *J Clin Monitor*, 10(6):392–404, November 1994.

J W Sleigh, J Andrzejowski, A Steyn-Ross, and M Steyn-Ross. The bispectral index: A measure of depth of sleep? *Anesth Analg*, 88(3):659–661, Mar 1999.

M M R F Struys, T De Smet, S D Greenwald, A R Abasalom, S Bingé, and E P Mortier. Closed-loop controlled administration of propofol using bispectral analysis. *Anesthesiology*, 95(1):6–17, Jul 2001.

M M R F Struys, T De Smet, S D Greenwald, A R Absalom, S Bingé, and E P Mortier. Performance evaluation of two published closed-loop control systems using bispectral index monitoring: A simulation study. *Anesthesiology*, 100(3):640–7, Mar 2004.

M M R F Struys, M J Coppens, N De Neve, E P Mortier, A G Doufas, J F P Van Bocxlaer, and S L Shafer. Influence of administration rate on propofol plasma-effect site equilibration. *Anesthesiology*, 07(3):386–396, Sept 2007.

R Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In Touretzky, Mozer, and Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 1038–1044. The MIT Press, 1996.

R S Sutton and A G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

G Tesauro. Temporal difference learning of backgammon strategy. In *Proceedings of the International Conference on Machine Learning*, pages 451–457. Morgan Kaufmann, 1992.

D R Theil, T E Stanley, 3rd, W D White, D Goodman, P S Glass, S A Bai, J R Jacobs, and J G Reves. Midazolam and fentanyl continuous infusion anesthesia for cardiac surgery: A comparison of computer-assisted versus manual infusion systems. *J Cardiothorac Vasc Anesth*, 7(3):300–6, Jun 1993.

J N Tsitsiklas and B Van Roy. An analysis of temporal difference learning with function approximation. Technical Report LIDS-P-2322, Massachusetts Institute of Technology, 1996.

C Vanlersberghe and F Camu. *Modern Anesthetics (Handbook of Experimental Pharmacology)*, volume 182, chapter Propofol, pages 227–252. Springer, 2008.

J R Varvel, D L Donoho, and S L Shafer. Measuring the predictive performance of computer-controlled infusion pumps. *J Pharmacokinet Biopharm*, 20:63–94, Feb 1992.

C J C H Watkins. *Learning from Delayed Rewards*. PhD dissertation, Cambridge University, Computer Science Department, 1989.

W Wood. Variability of human drug response. *Anesthesiology*, 71(4):631–634, Nov 1989.

L Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

Y Zhao, M R Kosorok, and D Zeng. Reinforcement learning design for cancer clinical trials. *Stat Med*, 28(26):3294–315, Nov 2009.

Journal of Machine Learning Research 15 (2014) 697-747 Submitted 10/12; Revised 6/13; Published 2/14

Clustering Hidden Markov Models with Variational HEM

Emanuele Coviello ECOVIELL@UCSD.EDU
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093, USA

Antoni B. Chan ABCHAN@CITYU.EDU.HK
Department of Computer Science
City University of Hong Kong
Kowloon Tong, Hong Kong

Gert R.G. Lanckriet GERT@ECE.UCSD.EDU
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093, USA

Editor: Tony Jebara

Abstract

The hidden Markov model (HMM) is a widely-used generative model that copes with sequential data, assuming that each observation is conditioned on the state of a hidden Markov chain. In this paper, we derive a novel algorithm to cluster HMMs based on the hierarchical EM (HEM) algorithm. The proposed algorithm i) clusters a given collection of HMMs into groups of HMMs that are similar, in terms of the distributions they represent, and ii) characterizes each group by a "cluster center", that is, a novel HMM that is representative for the group, in a manner that is consistent with the underlying generative model of the HMM. To cope with intractable inference in the E-step, the HEM algorithm is formulated as a variational optimization problem, and efficiently solved for the HMM case by leveraging an appropriate variational approximation. The benefits of the proposed algorithm, which we call variational HEM (VHEM), are demonstrated on several tasks involving time-series data, such as hierarchical clustering of motion capture sequences, and automatic annotation and retrieval of music and of online hand-writing data, showing improvements over current methods. In particular, our variational HEM algorithm effectively leverages large amounts of data when learning annotation models by using an efficient hierarchical estimation procedure, which reduces learning times and memory requirements, while improving model robustness through better regularization.

Keywords: Hierarchical EM algorithm, clustering, hidden Markov model, hidden Markov mixture model, variational approximation, time-series classification

1. Introduction

The hidden Markov model (HMM) (Rabiner and Juang, 1993) is a probabilistic model that assumes a signal is generated by a double embedded stochastic process. A discrete-time hidden state process, which evolves as a Markov chain, encodes the dynamics of the signal, while an observation process encodes the appearance of the signal at each time,

conditioned on the current state. HMMs have been successfully applied to a variety of fields, including speech recognition (Rabiner and Juang, 1993), music analysis (Qi et al., 2007) and identification (Batlle et al., 2002), online hand-writing recognition (Nag et al., 1986), analysis of biological sequences (Krogh et al., 1994), and clustering of time series data (Jebara et al., 2007; Smyth, 1997; Alon et al., 2003).

This paper is about clustering HMMs. More precisely, we are interested in an algorithm that, given a collection of HMMs, partitions them into K clusters of "similar" HMMs, while also learning a representative HMM "cluster center" that concisely and appropriately represents each cluster. This is similar to standard k-means clustering, except that the data points are HMMs now instead of vectors in $\mathbb{R}^d$.

Various applications motivate the design of HMM clustering algorithms, ranging from hierarchical clustering of sequential data (e.g., speech or motion sequences modeled by HMMs as by Jebara et al. 2007), to hierarchical indexing for fast retrieval, to reducing the computational complexity of estimating mixtures of HMMs from large data sets (e.g., semantic annotation models for music and video)—by clustering HMMs, efficiently estimated from many small subsets of the data, into a more compact mixture model of all data. However, there has been little work on HMM clustering and, therefore, its applications.

Existing approaches to clustering HMMs operate directly on the HMM *parameter* space, by grouping HMMs according to a suitable pairwise distance defined in terms of the HMM parameters. However, as HMM parameters lie on a non-linear manifold, a simple application of the k-means algorithm will not succeed in the task, since it assumes real vectors in a Euclidean space. In addition, such an approach would have the additional complication that HMM parameters for a particular generative model are not unique, that is, a permutation of the states leads to the same generative model. One solution, proposed by Jebara et al. (2007), first constructs an appropriate similarity matrix between all HMMs that are to be clustered (e.g., based on the Bhattacharya affinity, which depends non-linearly on the HMM parameters Jebara et al. 2004; Hershey and Olsen 2008), and then applies spectral clustering. While this approach has proven successful to group HMMs into similar clusters (Jebara et al., 2007), it does not directly address the issue of generating HMM cluster centers. Each cluster can still be represented by choosing one of the given HMMs, for example, the HMM which the spectral clustering procedure maps the closest to each spectral clustering center. However, this may be suboptimal for some applications of HMM clustering, for example in hierarchical estimation of annotation models. Another distance between HMM distributions suitable for spectral clustering is the KL divergence, which in practice has been approximated by sampling sequences from one model and computing their log-likelihood under the other (Juang and Rabiner, 1985; Zhong and Ghosh, 2003; Yin and Yang, 2005).

Instead, in this paper we propose to cluster HMMs *directly* with respect to the *probability distributions* they represent. The probability distributions of the HMMs are used throughout the whole clustering algorithm, and not only to construct an initial embedding as Jebara et al. (2007). By clustering the output distributions of the HMMs, marginalized over the hidden-state distributions, we avoid the issue of multiple equivalent parameterizations of the hidden states. We derive a hierarchical expectation maximization (HEM) algorithm that, starting from a collection of input HMMs, estimates a smaller mixture model of HMMs that concisely represents and clusters the input HMMs (i.e., the input HMM distributions guide the estimation of the output mixture distribution).

The HEM algorithm is a *generalization* of the EM algorithm—the EM algorithm can be considered as a special case of HEM for a mixture of delta functions as input. The main difference between HEM and EM is in the E-step. While the EM algorithm computes the sufficient statistics given the observed data, the HEM algorithm calculates the expected sufficient statistics averaged over all possible observations generated by the input probability models. Historically, the first HEM algorithm was designed to cluster *Gaussian* probability distributions (Vasconcelos and Lippman, 1998). This algorithm starts from a Gaussian mixture model (GMM) with $K^{(b)}$ components and reduces it to another GMM with fewer components, where each of the mixture components of the reduced GMM represents, that is, *clusters*, a group of the original Gaussian mixture components. More recently, Chan et al. (2010b) derived an HEM algorithm to cluster *dynamic texture* (DT) models (i.e., linear dynamical systems, LDSs) through their probability distributions. HEM has been applied successfully to construct GMM hierarchies for efficient image indexing (Vasconcelos, 2001), to cluster video represented by DTs (Chan et al., 2010a), and to estimate GMMs or DT mixtures (DTMs, that is, LDS mixtures) from large data sets for semantic annotation of images (Carneiro et al., 2007), video (Chan et al., 2010a) and music (Turnbull et al., 2008; Coviello et al., 2011). Note that HMMs cannot be clustered by using the original HEM by Vasconcelos and Lippman (1998). Specifically, the original formulation of HEM was designed for clustering data points represented by *individual* Gaussian models. When clustering HMMs, we are interested in assigning every HMM as a whole to a cluster, and do not want to treat their individual Gaussian states independently. Even with GMMs (as opposed to single Gaussians) this is not possible in closed form, since it would need the expected log likelihood of a mixture.

To extend the HEM framework from clustering Gaussians to clustering HMMs, additional marginalization over the hidden-state processes is required, as with DTs. However, while Gaussians and DTs allow tractable inference in the E-step of HEM, this is no longer the case for HMMs. Therefore, in this work, we derive a variational formulation of the HEM algorithm (VHEM), and then leverage a variational *approximation* derived by Hershey et al. (2007) (which has not been used in a learning context so far) to make the inference in the E-step tractable. The resulting algorithm not only clusters HMMs, but also learns novel HMMs that are representative centers of each cluster. The resulting VHEM algorithm can be generalized to handle other classes of graphical models, for which exact computation of the E-step in the standard HEM would be intractable, by leveraging similar variational approximations—for example, any mixtures of continuous exponential family distributions (e.g., Gaussian) the more general case of HMMs with emission probabilities that are (mixtures of) continuous exponential family distributions.

Compared to the spectral clustering algorithm of Jebara et al. (2007), the VHEM algorithm has several advantages that make it suitable for a variety of applications. First, the VHEM algorithm is capable of both clustering, as well as learning *novel* cluster centers, in a manner that is consistent with the underlying generative probabilistic framework. In addition, since it does not require sampling steps, it is also scalable with low memory requirements. As a consequence, VHEM for HMMs allows for efficient estimation of HMM mixtures from large data sets using a hierarchical estimation procedure. In particular, intermediate HMM mixtures are first estimated in *parallel* by running the EM algorithm on small independent portions of the data set. The final model is then estimated from the

intermediate models using the VHEM algorithm. Because VHEM is based on maximum-likelihood principles, it drives model estimation towards similar optimal parameter values as performing maximum-likelihood estimation on the full data set. In addition, by averaging over all possible observations compatible with the input models in the E-step, VHEM provides an implicit form of regularization that prevents over-fitting and improves robustness of the learned models, compared to a direct application of the EM algorithm on the full data set. Note that, in contrast to Jebara et al. (2007), VHEM does not construct a kernel embedding, and is therefore expected to be more efficient, especially for large $K^{(b)}$.

In summary, the contributions of this paper are three-fold: i) we derive a variational formulation of the HEM algorithm for clustering HMMs, which generates novel HMM centers representative of each cluster; ii) we evaluate VHEM on a variety of clustering, annotation, and retrieval problems involving time-series data, showing improvement over current clustering methods; iii) we demonstrate in experiments that VHEM can effectively learn HMMs from large sets of data, more efficiently than standard EM, while improving model robustness through better regularization. With respect to our previous work, the VHEM algorithm for HMMs was originally proposed by Coviello et al. (2012a)

The remainder of the paper is organized as follows. We review the hidden Markov model (HMM) and the hidden Markov mixture model (H3M) in Section 2. We present the derivation of the VHEM-H3M algorithm in Section 3, followed by a discussion in Section 4.boldsymbol Finally, we present experimental results in Sections 5 and 6.

2. The Hidden Markov (Mixture) Model

A hidden Markov model (HMM) $\mathcal{M}$ assumes a sequence of τ observations $\mathbf{y} = \{y_1, \ldots, y_\tau\}$ is generated by a double embedded stochastic process, where each observation (or emission) y_t at time t depends on the state of a discrete hidden variable x_t, and the sequence of hidden states $\mathbf{x} = \{x_1, \ldots, x_\tau\}$ evolves as a first-order Markov chain. The hidden variables can take one of S values, $\{1, \ldots, S\}$, and the evolution of the hidden process is encoded in a state transition matrix $A = [a_{\beta,\beta'}]_{\beta,\beta'=1,\ldots,S}$, where each entry, $a_{\beta,\beta'} = p(x_{t+1} = \beta' | x_t = \beta, \mathcal{M})$, is the probability of transitioning from state β to state β', and an initial state distribution $\pi = [\pi_1, \ldots, \pi_S]$, where $\pi_\beta = p(x_1 = \beta | \mathcal{M})$.

Each state β generates observations according to an emission probability density function, $p(y_t | x_t = \beta, \mathcal{M})$. Here, we assume the emission density is *time-invariant*, and modeled as a Gaussian mixture model (GMM) with M components:

$$p(y | x = \beta, \mathcal{M}) = \sum_{m=1}^{M} c_{\beta,m} p(y | \zeta = m, x = \beta, \mathcal{M}), \tag{1}$$

where $\zeta \sim \text{multinomial}(c_{\beta,1}, \ldots, c_{\beta,M})$ is the hidden assignment variable that selects the mixture component, with $c_{\beta,m}$ as the mixture weight of the mth component, and each component is a multivariate Gaussian distribution,

$$p(y | \zeta = m, x = \beta, \mathcal{M}) = \mathcal{N}(y; \mu_{\beta,m}, \Sigma_{\beta,m}),$$

with mean $\mu_{\beta,m}$ and covariance matrix $\Sigma_{\beta,m}$. The HMM is specified by the parameters

$$\mathcal{M} = \{\pi, A, \{\{c_{\beta,m}, \mu_{\beta,m}, \Sigma_{\beta,m}\}_{m=1}^{M}\}_{\beta=1}^{S}\},$$

which can be efficiently learned from an observation sequence $\mathbf{y}$ with the Baum-Welch algorithm (Rabiner and Juang, 1993), which is based on maximum likelihood estimation.

The probability distribution of a state sequence $\mathbf{x}$ generated by an HMM $\mathcal{M}$ is

$$p(\mathbf{x}|\mathcal{M}) = p(x_1|\mathcal{M}) \prod_{t=2}^{\tau} p(x_t|x_{t-1}, \mathcal{M}) = \pi_{x_1} \prod_{t=2}^{\tau} a_{x_{t-1},x_t},$$

while the joint likelihood of an observation sequence $\mathbf{y}$ and a state sequence $\mathbf{x}$ is

$$p(\mathbf{y}, \mathbf{x}|\mathcal{M}) = p(\mathbf{y}|\mathbf{x}, \mathcal{M}) p(\mathbf{x}|\mathcal{M}) = p(x_1|\mathcal{M}) \prod_{t=2}^{\tau} p(x_t|x_{t-1}, \mathcal{M}) \prod_{t=1}^{\tau} p(y_t|x_t, \mathcal{M}).$$

Finally, the observation likelihood of $\mathbf{y}$ is obtained by marginalizing out the state sequence from the joint likelihood,

$$p(\mathbf{y}|\mathcal{M}) = \sum_{\mathbf{x}} p(\mathbf{y}, \mathbf{x}|\mathcal{M}) = \sum_{\mathbf{x}} p(\mathbf{y}|\mathbf{x}, \mathcal{M}) p(\mathbf{x}|\mathcal{M}), \tag{2}$$

where the summation is over all state sequences of length τ, and can be performed efficiently using the *forward algorithm* (Rabiner and Juang, 1993).

A hidden Markov mixture model (H3M) (Smyth, 1997) models a set of observation sequences as samples from a group of K hidden Markov models, each associated to a specific sub-behavior. For a given sequence, an assignment variable $z \sim \text{multinomial}(\omega_1, \cdots, \omega_K)$ selects the parameters of one of the K HMMs, where the kth HMM is selected with probability ω_k. Each mixture component is parametrized by

$$\mathcal{M}_z = \{\pi^z, A^z, \{\{c^z_{\beta,m}, \mu^z_{\beta,m}, \Sigma^z_{\beta,m}\}_{m=1}^{M}\}_{\beta=1}^{S}\},$$

and the H3M is parametrized by $\mathcal{M} = \{\omega_z, \mathcal{M}_z\}_{z=1}^{K}$, which can be estimated from a collection of observation sequences using the EM algorithm (Smyth, 1997; Alon et al., 2003).

To reduce clutter, here we assume that all the HMMs have the same number S of hidden states and that all emission probabilities have M mixture components. Our derivation could be easily extended to the more general case though.

3. Clustering Hidden Markov Models

Algorithms for clustering HMMs can serve a wide range of applications, from hierarchical clustering of sequential data (e.g., speech or motion sequences modeled by HMMs (Jebara et al., 2007)), to hierarchical indexing for fast retrieval, to reducing the computational complexity of estimating mixtures of HMMs from large weakly-annotated data sets—by clustering HMMs, efficiently estimated from many small subsets of the data, into a more compact mixture model of all data.

In this work we derive a hierarchical EM algorithm for clustering HMMs (HEM-H3M) with respect to their probability distributions. We approach the problem of clustering HMMs as reducing an input HMM mixture with a large number of components to a new mixture with fewer components. Note that different HMMs in the input mixture are allowed to have different weights (i.e., the mixture weights $\{\omega_z\}_{z=1}^{K}$ are not necessarily all equal).

One method for estimating the reduced mixture model is to generate samples from the input mixture, and then perform maximum likelihood estimation, that is, maximize the log-likelihood of these samples. However, to avoid explicitly generating these samples, we instead maximize the *expectation* of the log-likelihood with respect to the input mixture model, thus averaging over all possible samples from the input mixture model. In this way, the dependency on the samples is replaced by a marginalization with respect to the input mixture model. While such marginalization is tractable for Gaussians and DTs, this is no longer the case for HMMs. Therefore, in this work, we i) derive a variational formulation of the HEM algorithm (VHEM), and ii) specialize it to the HMM case by leveraging a variational approximation proposed by Hershey et al. (2007). Note that the work of Hershey et al. (2007) was proposed as an alternative to MCMC sampling for the computation of the KL divergence between two HMMs, and has not been used in a learning context so far.

We present the problem formulation in Section 3.1, and derive the algorithm in Sections 3.2, 3.3 and 3.4.

3.1 Formulation

Let $\mathcal{M}^{(b)}$ be a base hidden Markov mixture model with $K^{(b)}$ components. The goal of the VHEM algorithm is to find a reduced hidden Markov mixture model $\mathcal{M}^{(r)}$ with $K^{(r)} < K^{(b)}$ (i.e., fewer) components that represents $\mathcal{M}^{(b)}$ well. The likelihood of a random sequence $\mathbf{y} \sim \mathcal{M}^{(b)}$ is given by

$$p(\mathbf{y}|\mathcal{M}^{(b)}) = \sum_{i=1}^{K^{(b)}} \omega_i^{(b)} p(\mathbf{y}|z^{(b)} = i, \mathcal{M}^{(b)}), \tag{3}$$

where $z^{(b)} \sim \text{multinomial}(\omega_1^{(b)}, \cdots \omega_{K^{(b)}}^{(b)})$ is the hidden variable that indexes the mixture components. $p(\mathbf{y}|z = i, \mathcal{M}^{(b)})$ is the likelihood of $\mathbf{y}$ under the ith mixture component, as in (2), and $\omega_i^{(b)}$ is the mixture weight for the ith component. Likewise, the likelihood of the random sequence $\mathbf{y} \sim \mathcal{M}^{(r)}$ is

$$p(\mathbf{y}|\mathcal{M}^{(r)}) = \sum_{j=1}^{K^{(r)}} \omega_j^{(r)} p(\mathbf{y}|z^{(r)} = j, \mathcal{M}^{(r)}), \tag{4}$$

where $z^{(r)} \sim \text{multinomial}(\omega_1^{(r)}, \cdots, \omega_{K^{(r)}}^{(r)})$ is the hidden variable for indexing components in $\mathcal{M}^{(r)}$.

At a high level, the VHEM-H3M algorithm estimates the reduced H3M model $\mathcal{M}^{(r)}$ in (4) from *virtual* sequences distributed according to the base H3M model $\mathcal{M}^{(b)}$ in (3). From this estimation procedure, the VHEM algorithm provides:

1. a soft clustering of the original $K^{(b)}$ components into $K^{(r)}$ groups, where cluster membership is encoded in assignment variables that represent the *responsibility* of each reduced mixture component for each base mixture component, that is, $\hat{z}_{i,j} = p(z^{(r)} = j|z^{(b)} = i)$, for $i = 1, \ldots, K^{(b)}$ and $j = 1, \ldots, K^{(r)}$;

variables	*base model (b)*	*reduced model (r)*
index for HMM components	i	j
number of HMM components	$K^{(b)}$	$K^{(r)}$
HMM states	β	ρ
number of HMM states	S	S
HMM state sequence	$\boldsymbol{\beta} = \{\beta_1, \cdots, \beta_\tau\}$	$\boldsymbol{\rho} = \{\rho_1, \cdots, \rho_\tau\}$
index for component of GMM	m	ℓ
number of Gaussian components	M	M
models		
H3M	$\mathcal{M}^{(b)}$	$\mathcal{M}^{(r)}$
HMM component (of H3M)	$\mathcal{M}_i^{(b)}$	$\mathcal{M}_j^{(r)}$
GMM emission	$\mathcal{M}_{i,\beta}^{(b)}$	$\mathcal{M}_{j,\rho}^{(r)}$
Gaussian component (of GMM)	$\mathcal{M}_{i,\beta,m}^{(b)}$	$\mathcal{M}_{j,\rho,\ell}^{(r)}$
parameters		
H3M component weight	$\omega_i^{(b)}$	$\omega_j^{(r)}$
HMM initial state	$\pi^{(b),i}$	$\pi^{(r),j}$
HMM state transition matrix	$A^{(b),i}$	$A^{(r),j}$
GMM emission	$\{c_{\beta,m}^{(b),i}, \mu_{\beta,m}^{(b),i}, \Sigma_{\beta,m}^{(b),i}\}_{m=1}^M$	$\{c_{\rho,\ell}^{(r),j}, \mu_{\rho,\ell}^{(r),j}, \Sigma_{\rho,\ell}^{(r),j}\}_{\ell=1}^M$
probability distributions	*notation*	*short-hand*
HMM state sequence (b)	$p(\mathbf{x} = \boldsymbol{\beta} \vert z^{(b)} = i, \mathcal{M}^{(b)})$	$p(\boldsymbol{\beta} \vert \mathcal{M}_i^{(b)}) = \pi_{\boldsymbol{\beta}}^{(b),i}$
HMM state sequence (r)	$p(\mathbf{x} = \boldsymbol{\rho} \vert z^{(r)} = j, \mathcal{M}^{(r)})$	$p(\boldsymbol{\rho} \vert \mathcal{M}_j^{(r)}) = \pi_{\boldsymbol{\rho}}^{(r),j}$
HMM observation likelihood (r)	$p(\mathbf{y} \vert z^{(r)} = j, \mathcal{M}^{(r)})$	$p(\mathbf{y} \vert \mathcal{M}_j^{(r)})$
GMM emission likelihood (r)	$p(y_t \vert x_t = \rho, \mathcal{M}_j^{(r)})$	$p(y_t \vert \mathcal{M}_{j,\rho}^{(r)})$
Gaussian component likelihood (r)	$p(y_t \vert \zeta_t = \ell, x_t = \rho, \mathcal{M}_j^{(r)})$	$p(y_t \vert \mathcal{M}_{j,\rho,\ell}^{(r)})$
expectations		
HMM observation sequence (b)	$\mathrm{E}_{\mathbf{y} \vert z^{(b)}=i, \mathcal{M}^{(b)}}[\cdot]$	$\mathrm{E}_{\mathcal{M}_i^{(b)}}[\cdot]$
GMM emission (b)	$\mathrm{E}_{y_t \vert x_t=\beta, \mathcal{M}_i^{(b)}}[\cdot]$	$\mathrm{E}_{\mathcal{M}_{i,\beta}^{(b)}}[\cdot]$
Gaussian component (b)	$\mathrm{E}_{y_t \vert \zeta_t=m, x_t=\beta, \mathcal{M}_i^{(b)}}[\cdot]$	$\mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\cdot]$
expected log-likelihood	*lower bound*	*variational distribution*
$\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(Y_i \vert \mathcal{M}^{(r)})]$	$\mathcal{L}_{H3M}^i$	$q_i(z_i = j) = z_{ij}$
$\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y} \vert \mathcal{M}_j^{(r)})]$	$\mathcal{L}_{HMM}^{i,j}$	$q^{i,j}(\boldsymbol{\rho} \vert \boldsymbol{\beta}) = \phi_{\boldsymbol{\rho} \vert \boldsymbol{\beta}}^{i,j}$ $= \phi_1^{i,j}(\rho_1 \vert \beta_1) \prod_{t=2}^{\tau} \phi_t^{i,j}(\rho_t \vert \rho_{t-1}, \beta_t)$
$\mathrm{E}_{\mathcal{M}_{i,\beta}^{(b)}}[\log p(y \vert \mathcal{M}_{j,\rho}^{(r)})]$	$\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$	$q_{\beta,\rho}^{i,j}(\zeta = \ell \vert m) = \eta_{\ell \vert m}^{(i,\beta),(j,\rho)}$

Table 1: Notation used in the derivation of the VHEM-H3M algorithm.

2. novel cluster centers represented by the individual mixture components of the reduced model in (4), that is, $p(\mathbf{y} \vert z^{(r)} = j, \mathcal{M}^{(r)})$ for $j = 1, \ldots, K^{(r)}$.

Finally, because we take the expectation over the virtual samples, the estimation is carried out in an efficient manner that requires only knowledge of the parameters of the base model, without the need of generating actual virtual samples.

3.1.1 Notation

We will always use i and j to index the components of the base model $\mathcal{M}^{(b)}$ and the reduced model $\mathcal{M}^{(r)}$, respectively. To reduce clutter, we will also use the short-hand notation $\mathcal{M}_i^{(b)}$ and $\mathcal{M}_j^{(r)}$ to denote the ith component of $\mathcal{M}^{(b)}$ and the jth component of $\mathcal{M}^{(r)}$, respectively. Hidden states of the HMMs are denoted with β for the base model $\mathcal{M}_i^{(b)}$, and with ρ for the reduced model $\mathcal{M}_j^{(r)}$.

The GMM emission models for each hidden state are denoted as $\mathcal{M}_{i,\beta}^{(b)}$ and $\mathcal{M}_{j,\rho}^{(r)}$. We will always use m and ℓ for indexing the individual Gaussian components of the GMM emissions of the base and reduced models, respectively. The individual Gaussian components are denoted as $\mathcal{M}_{i,\beta,m}^{(b)}$ for the base model, and $\mathcal{M}_{j,\rho,\ell}^{(r)}$ for the reduced model. Finally, we denote the parameters of ith HMM component of the base mixture model as $\mathcal{M}_i^{(b)} = \{\pi^{(b),i}, A^{(b),i}, \{\{c_{\beta,m}^{(b),i}, \mu_{\beta,m}^{(b),i}, \Sigma_{\beta,m}^{(b),i}\}_{m=1}^{M}\}_{\beta=1}^{S}\}$, and for the jth HMM in the reduced mixture as $\mathcal{M}_j^{(r)} = \{\pi^{(r),j}, A^{(r),j}, \{\{c_{\rho,\ell}^{(r),j}, \mu_{\rho,\ell}^{(r),j}, \Sigma_{\rho,\ell}^{(r),j}\}_{\ell=1}^{M}\}_{\rho=1}^{S}\}$.

When appearing in a probability distribution, the short-hand model notation (e.g., $\mathcal{M}_i^{(b)}$) always implies *conditioning* on the model being active. For example, we will use $p(\mathbf{y}|\mathcal{M}_i^{(b)})$ as short-hand for $p(\mathbf{y}|z^{(b)} = i, \mathcal{M}^{(b)})$, or $p(y_t|\mathcal{M}_{i,\beta}^{(b)})$ as short-hand for $p(y_t|x_t = \beta, z^{(b)} = i, \mathcal{M}^{(b)})$. Furthermore, we will use $\pi_{\boldsymbol{\beta}}^{(b),i}$ as short-hand for the probability of the state sequence $\boldsymbol{\beta}$ according to the base HMM component $\mathcal{M}_i^{(b)}$, that is, $p(\boldsymbol{\beta}|\mathcal{M}_i^{(b)})$, and likewise $\mathcal{M}_{\boldsymbol{\rho}}^{(r),j}$ for the reduced HMM component.

Expectations will also use the short-hand model notation to imply conditioning on the model. In addition, expectations are assumed to be taken with respect to the output variable ($\mathbf{y}$ or y_t), unless otherwise specified. For example, we will use $\mathrm{E}_{\mathcal{M}_i^{(b)}}[\cdot]$ as short-hand for $\mathrm{E}_{\mathbf{y}|z^{(b)}=i,\mathcal{M}^{(b)}}[\cdot]$.

Table 1 summarizes the notation used in the derivation, including the variable names, model parameters, and short-hand notations for probability distributions and expectations. The bottom of Table 1 also summarizes the variational lower bound and variational distributions, which will be introduced subsequently.

3.2 Variational HEM Algorithm

To learn the reduced model in (4), we consider a set of N *virtual* samples, distributed according to the base model $\mathcal{M}^{(b)}$ in (3), such that $N_i = N\omega_i^{(b)}$ samples are drawn from the ith component. We denote the set of N_i virtual samples for the ith component as $Y_i = \{\mathbf{y}^{(i,m)}\}_{m=1}^{N_i}$, where $\mathbf{y}^{(i,m)} \sim \mathcal{M}_i^{(b)}$, and the entire set of N samples as $Y = \{Y_i\}_{i=1}^{K^{(b)}}$. Note that, in this formulation, we are not considering virtual samples $\{\mathbf{x}^{(i,m)}, \mathbf{y}^{(i,m)}\}$ for each base component, according to its joint distribution $p(\mathbf{x}, \mathbf{y}|\mathcal{M}_i^{(b)})$. The reason is that the hidden-state space of each base mixture component $\mathcal{M}_i^{(b)}$ may have a different representation (e.g., the numbering of the hidden states may be permuted between the components). This mismatch will cause problems when the parameters of $\mathcal{M}_j^{(r)}$ are computed from virtual samples of the hidden states of $\{\mathcal{M}_i^{(b)}\}_{i=1}^{K^{(b)}}$. Instead, we treat $X_i = \{\mathbf{x}^{(i,m)}\}_{m=1}^{N_i}$ as "missing"

information, and estimate them in the E-step. The log-likelihood of the virtual samples is

$$\log p(Y|\mathcal{M}^{(r)}) = \sum_{i=1}^{K^{(b)}} \log p(Y_i|\mathcal{M}^{(r)}), \tag{5}$$

where, in order to obtain a consistent clustering, we assume the entirety of samples Y_i is assigned to the same component of the reduced model (Vasconcelos and Lippman, 1998).

The original formulation of HEM (Vasconcelos and Lippman, 1998) maximizes (5) with respect to $\mathcal{M}^{(r)}$, and uses the law of large numbers to turn the virtual samples Y_i into an expectation over the base model components $\mathcal{M}_i^{(b)}$. In this paper, we will start with a different objective function to derive the VHEM algorithm. To estimate $\mathcal{M}^{(r)}$, we will maximize the average log-likelihood of all possible virtual samples, weighted by their likelihood of being generated by $\mathcal{M}_i^{(b)}$, that is, the *expected* log-likelihood of the virtual samples,

$$\mathcal{J}(\mathcal{M}^{(r)}) = \mathrm{E}_{\mathcal{M}^{(b)}}\left[\log p(Y|\mathcal{M}^{(r)})\right] = \sum_{i=1}^{K^{(b)}} \mathrm{E}_{\mathcal{M}_i^{(b)}}\left[\log p(Y_i|\mathcal{M}^{(r)})\right], \tag{6}$$

where the expectation is over the base model components $\mathcal{M}_i^{(b)}$. Maximizing (6) will eventually lead to the same estimate as maximizing (5), but allows us to strictly preserve the variational lower bound, which would otherwise be ruined when applying the law of large numbers to (5).

A general approach to deal with maximum likelihood estimation in the presence of hidden variables (which is the case for H3Ms) is the EM algorithm (Dempster et al., 1977). In the traditional formulation the EM algorithm is presented as an alternation between an expectation step (E-step) and a maximization step (M-step). In this work, we take a variational perspective (Neal and Hinton, 1998; Wainwright and Jordan, 2008; Csiszár and Tusnády, 1984), which views each step as a maximization step. The variational E-step first obtains a family of lower bounds to the (expected) log-likelihood (i.e., to Equation 6), indexed by variational parameters, and then optimizes over the variational parameters to find the tightest bound. The corresponding M-step then maximizes the lower bound (with the variational parameters fixed) with respect to the model parameters. One advantage of the variational formulation is that it readily allows for useful extensions to the EM algorithm, such as replacing a difficult inference in the E-step with a variational approximation. In practice, this is achieved by restricting the maximization in the variational E-step to a smaller domain for which the lower bound is tractable.

The EM algorithm with variational E-step is guaranteed to converge (Gunawardana and Byrne, 2005). Despite the approximation prevents convergence to local maxima of the data log-likelihood (Gunawardana and Byrne, 2005), the algorithm still performs well empirically, as shown in Section 5 and Section 6.

3.2.1 Lower Bound to an Expected Log-likelihood

Before proceeding with the derivation of VHEM for H3Ms, we first need to derive a lower-bound to an expected log-likelihood term, for example, (6). Our derivation starts from a variational lower-bound to a log-likelihood (as opposed to an *expected* log-likelihood), a

standard tool in machine learning (Jordan et al., 1999; Jaakkola, 2000), which we briefly review next. In all generality, let $\{O, H\}$ be the observation and hidden variables of a probabilistic model, respectively, where $p(H)$ is the distribution of the hidden variables, $p(O|H)$ is the conditional likelihood of the observations, and $p(O) = \sum_H p(O|H)p(H)$ is the observation likelihood. We can define a *variational lower bound* to the observation log-likelihood (Jordan et al., 1999; Jaakkola, 2000):

$$\begin{aligned}\log p(O) &\geq \log p(O) - D(q(H)||p(H|O)) \\ &= \sum_H q(H) \log \frac{p(H)p(O|H)}{q(H)},\end{aligned}$$

where $p(H|O)$ is the posterior distribution of H given observation O, and $D(p\|q) = \int p(y) \log \frac{p(y)}{q(y)} dy$ is the Kullback-Leibler (KL) divergence between two distributions, p and q. We introduce a variational distribution $q(H)$, which approximates the posterior distribution, where $\sum_H q(H) = 1$ and $q(H) \geq 0$. When the variational distribution equals the true posterior, $q(H) = P(H|O)$, then the KL divergence is zero, and hence the lower-bound reaches $\log p(O)$. When the true posterior cannot be computed, then typically q is restricted to some set of approximate posterior distributions $\mathcal{Q}$ that are tractable, and the best lower-bound is obtained by maximizing over $q \in \mathcal{Q}$,

$$\log p(O) \geq \max_{q \in \mathcal{Q}} \sum_H q(H) \log \frac{p(H)p(O|H)}{q(H)}. \tag{7}$$

From the standard lower bound in (7), we can now derive a lower bound to an expected log-likelihood expression. Let $\mathrm{E}_b[\cdot]$ be the expectation with respect to O with some distribution $p_b(O)$. Since $p_b(O)$ is non-negative, taking the expectation on both sides of (7) yields,

$$\mathrm{E}_b\left[\log p(O)\right] \geq \mathrm{E}_b\left[\max_{q \in Q} \sum_H q(H) \log \frac{p(H)p(O|H)}{q(H)}\right] \tag{8}$$

$$\geq \max_{q \in Q} \mathrm{E}_b\left[\sum_H q(H) \log \frac{p(H)p(O|H)}{q(H)}\right] \tag{9}$$

$$= \max_{q \in Q} \sum_H q(H) \left\{\log \frac{p(H)}{q(H)} + \mathrm{E}_b\left[\log p(O|H)\right]\right\}, \tag{10}$$

where (9) follows from Jensen's inequality (i.e., $f(\mathrm{E}[x]) \leq \mathrm{E}[f(x)]$ when f is convex), and the convexity of the max function. Hence, (10) is a variational lower bound on the expected log-likelihood, which depends on the family of variational distributions Q.

In (8) we are computing the best lower-bound (7) to $\log p(O)$ *individually* for each value of the observation variable O, which in general corresponds to different optimal $q^* \in \mathcal{Q}$ for different values of O. Note that the expectation in (8) is not analytically tractable when $p(O)$ is a mixture model (i.e., it is the expected log-likelihood of a mixture). Hence, we treat the ensemble of observations $O \sim p_b$ as a whole, and in (9) find a single $q^* \in \mathcal{Q}$ for which the lower bound is best on average. Mathematically, this correspond to using Jensen inequality to pass from (8) to (9), which shows that the additional approximation makes the lower-bound looser.

3.2.2 Variational Lower Bound

We now derive a lower bound to the expected log-likelihood cost function in (6). The derivation will proceed by successively applying the lower bound from (10) to each expected log-likelihood term that arises. This will result in a set of nested lower bounds.

A variational lower bound to the expected log-likelihood of the virtual samples in (6) is obtained by lower bounding each of the expectation terms $\mathrm{E}_{\mathcal{M}_i^{(b)}}$ in the sum,

$$\mathcal{J}(\mathcal{M}^{(r)}) = \sum_{i=1}^{K^{(b)}} \mathrm{E}_{\mathcal{M}_i^{(b)}} \left[\log p(Y_i|\mathcal{M}^{(r)})\right] \geq \sum_{i=1}^{K^{(b)}} \mathcal{L}^i_{H3M}, \tag{11}$$

where we define three nested lower bounds, corresponding to different model elements (the H3M, the component HMMs, and the emission GMMs):

$$\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(Y_i|\mathcal{M}^{(r)})] \geq \mathcal{L}^i_{H3M}, \tag{12}$$

$$\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})] \geq \mathcal{L}^{i,j}_{HMM}, \tag{13}$$

$$\mathrm{E}_{\mathcal{M}_{i,\beta}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho}^{(r)})] \geq \mathcal{L}^{(i,\beta),(j,\rho)}_{GMM}. \tag{14}$$

In (12), the first lower bound, $\mathcal{L}^i_{H3M}$, is on the expected log-likelihood of an H3M $\mathcal{M}^{(r)}$ with respect to an HMM $\mathcal{M}_i^{(b)}$. Because $p(Y_i|\mathcal{M}^{(r)})$ is the likelihood under a mixture of HMMs, as in (4), where the observation variable is Y_i and the hidden variable is z_i (the assignment of Y_i to a component of $\mathcal{M}^{(r)}$), its expectation cannot be calculated directly. Hence, we introduce the variational distribution $q_i(z_i)$ and apply (10) to (12), yielding the lower bound (see Appendix A),

$$\mathcal{L}^i_{H3M} = \max_{q_i} \sum_j q_i(z_i = j) \left\{ \log \frac{p(z_i = j|\mathcal{M}^{(r)})}{q_i(z_i = j)} + N_i \mathcal{L}^{i,j}_{HMM} \right\}. \tag{15}$$

The lower bound in (15) depends on the second lower bound (Eq. 13), $\mathcal{L}^{i,j}_{HMM}$, which is on the expected log-likelihood of an HMM $\mathcal{M}_j^{(r)}$, averaged over observation sequences from a *different* HMM $\mathcal{M}_i^{(b)}$. Although the data log-likelihood $\log p(\mathbf{y}|\mathcal{M}_j^{(r)})$ can be computed exactly using the forward algorithm (Rabiner and Juang, 1993), calculating its expectation is not analytically tractable since an observation sequence $\mathbf{y}$ from a HMM $\mathcal{M}_j^{(r)}$ is essentially an observation from a mixture model.[1]

To calculate the lower bound $\mathcal{L}^{i,j}_{HMM}$ in (13), we first rewrite the expectation $\mathrm{E}_{\mathcal{M}_i^{(b)}}$ in (13) to explicitly marginalize over the state sequence $\boldsymbol{\beta}$ of $\mathcal{M}_i^{(b)}$, and then apply (10) where the hidden variable is the state sequence $\boldsymbol{\rho}$ of $\mathcal{M}_j^{(r)}$, yielding (see Appendix A)

$$\mathcal{L}^{i,j}_{HMM} = \sum_{\boldsymbol{\beta}} \pi^{(b),i}_{\boldsymbol{\beta}} \max_{q^{i,j}} \sum_{\boldsymbol{\rho}} q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left\{ \log \frac{p(\boldsymbol{\rho}|\mathcal{M}_j^{(r)})}{q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \sum_t \mathcal{L}^{(i,\beta_t),(j,\rho_t)}_{GMM} \right\}, \tag{16}$$

1. For an observation sequence of length τ, an HMM with S states can be considered as a mixture model with $O(S^\tau)$ components.

where we introduce a variational distribution $q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})$ on the state sequence $\boldsymbol{\rho}$, which depends on a particular sequence $\boldsymbol{\beta}$ from $\mathcal{M}_i^{(b)}$. As before, (16) depends on another nested lower bound, $\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$ in (14), which is on the expected log-likelihood of a GMM emission density $\mathcal{M}_{j,\rho}^{(r)}$ with respect to another GMM $\mathcal{M}_{i,\beta}^{(b)}$. This lower bound does not depend on time, as we have assumed that the emission densities are time-invariant.

Finally, we obtain the lower bound $\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$ for (14), by explicitly marginalizing over the GMM hidden assignment variable in $\mathcal{M}_{i,\beta}^{(b)}$ and then applying (10) to the expectation of the GMM emission distribution $p(y|\mathcal{M}_{j,\rho}^{(r)})$, yielding (see Appendix A),

$$\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)} = \sum_{m=1}^{M} c_{\beta,m}^{(b),i} \max_{q_{\beta,\rho}^{i,j}} \sum_{\zeta=1}^{M} q_{\beta,\rho}^{i,j}(\zeta|m) \left\{ \log \frac{p(\zeta|\mathcal{M}_{j,\rho}^{(r)})}{q_{\beta,\rho}^{i,j}(\zeta|m)} + \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho,\zeta}^{(r)})] \right\}, \quad (17)$$

where we introduce the variational distribution $q_{\beta,\rho}^{i,j}(\zeta|m)$, which is conditioned on the observation y arising from the mth component in $\mathcal{M}_{i,\beta}^{(b)}$. In (17), the term $\mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})]$ is the expected log-likelihood of the Gaussian distribution $\mathcal{M}_{j,\rho,\ell}^{(r)}$ with respect to the Gaussian $\mathcal{M}_{i,\beta,m}^{(b)}$, which has a closed-form solution (see Section 3.3.1).

In summary, we have derived a variational lower bound to the expected log-likelihood of the virtual samples, which is given by (11). This lower bound is composed of three nested lower bounds in (15), (16), and (17), corresponding to different model elements (the H3M, the component HMMs, and the emission GMMs), where $q_i(z_i)$, $q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})$, and $q_{\beta,\rho}^{i,j}(\zeta|m)$ are the corresponding variational distributions. Finally, the variational HEM algorithm for HMMs consists of two alternating steps:

- (variational E-step) given $\mathcal{M}^{(r)}$, calculate the variational distributions $q_{\beta,\rho}^{i,j}(\zeta|m)$, $q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})$, and $q_i(z_i)$ for the lower bounds in (17), (16), and (15);
- (M-step) update the model parameters via $\mathcal{M}^{(r)*} = \mathrm{argmax}_{\mathcal{M}^{(r)}} \sum_{i=1}^{K^{(b)}} \mathcal{L}_{H3M}^{i}$.

In the following subsections, we derive the E- and M-steps of the algorithm. The entire procedure is summarized in Algorithm 1.

3.3 Variational E-Step

The variational E-step consists of finding the variational distributions that maximize the lower bounds in (17), (16), and (15). In particular, given the nesting of the lower bounds, we proceed by first maximizing the GMM lower bound $\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$ for each pair of emission GMMs in the base and reduced models. Next, the HMM lower bound $\mathcal{L}_{HMM}^{i,j}$ is maximized for each pair of HMMs in the base and reduced models, followed by maximizing the H3M lower bound $\mathcal{L}_{H3M}^{i}$ for each base HMM. Finally, a set of summary statistics are calculated, which will be used in the M-step.

3.3.1 Variational Distributions

We first consider the forms of the three variational distributions, as well as the optimal parameters to maximize the corresponding lower bounds.

Algorithm 1 VHEM algorithm for H3Ms

Input: base H3M $\mathcal{M}^{(b)} = \{\omega_i^{(b)}, \mathcal{M}_i^{(b)}\}_{i=1}^{K^{(b)}}$, number of virtual samples N.
Initialize reduced H3M $\mathcal{M}^{(r)} = \{\omega_j^{(r)}, \mathcal{M}_j^{(r)}\}_{j=1}^{K^{(r)}}$.
repeat
  {Variational E-step}
  Compute optimal variational distributions and variational lower bounds:
  for each pair of HMMs $\mathcal{M}_i^{(s)}$ and $\mathcal{M}_j^{(r)}$
    for each pair of emission GMMs for state β of $\mathcal{M}_i^{(s)}$ and ρ of $\mathcal{M}_j^{(r)}$:
      Compute optimal variational distributions $\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)}$ as in (18)
      Compute optimal lower bound $\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$ to expected log-likelihood as in (22)
    Compute optimal variational distributions for HMMs as in Appendin B
      $\hat{\phi}_1^{i,j}(\rho_1|\beta_1), \quad \hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1}, \beta_t)$ for $t = \tau, \ldots, 2$
    Compute optimal lower bound $\mathcal{L}_{HMM}^{i,j}$ to expected log-likelihood as in (21)
  Compute optimal assignment probabilities:

$$\hat{z}_{ij} = \frac{\omega_j^{(r)} \exp(N\omega_i^{(b)} \mathcal{L}_{HMM}^{i,j})}{\sum_{j'} \omega_{j'}^{(r)} \exp(N\omega_i^{(b)} \mathcal{L}_{HMM}^{i,j'})}$$

  Compute aggregate summary statistics for each pair of HMMs $\mathcal{M}_i^{(s)}$ and $\mathcal{M}_j^{(r)}$ as in Section 3.3.3:

$$\hat{\nu}_1^{i,j}(\rho) = \sum_{\beta=1}^{S} \nu_1^{i,j}(\rho, \beta), \quad \hat{\nu}^{i,j}(\rho, \beta) = \sum_{t=1}^{\tau} \nu_t^{i,j}(\rho, \beta), \quad \hat{\xi}^{i,j}(\rho, \rho') = \sum_{t=2}^{\tau} \sum_{\beta=1}^{S} \xi_t^{i,j}(\rho, \rho', \beta)$$

  {M-step}
  For each component $\mathcal{M}_j^{(r)}$, recompute parameters using (24)-(28).
until convergence
Output: reduced H3M $\{\omega_j^{(r)}, \mathcal{M}_j^{(a)}\}_{j=1}^{K^{(r)}}$.

GMM: For the GMM lower bound $\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$, we assume each variational distribution has the form (Hershey et al., 2007)

$$q_{\beta,\rho}^{i,j}(\zeta = l|m) = \eta_{\ell|m}^{(i,\beta),(j,\rho)},$$

where $\sum_{\ell=1}^{M} \eta_{\ell|m}^{(i,\beta),(j,\rho)} = 1$, and $\eta_{\ell|m}^{(i,\beta),(j,\rho)} \geq 0$, $\forall \ell$. Intuitively, $\boldsymbol{\eta}^{(i,\beta),(j,\rho)}$ is the responsibility matrix between each pair of Gaussian components in the GMMs $\mathcal{M}_{i,\beta}^{(b)}$ and $\mathcal{M}_{j,\rho}^{(r)}$, where $\eta_{\ell|m}^{(i,\beta),(j,\rho)}$ represents the probability that an observation from component m of $\mathcal{M}_{i,\beta}^{(b)}$ corresponds to component ℓ of $\mathcal{M}_{j,\rho}^{(r)}$. Substituting into (17) and maximizing the variational

parameters yields (see Appendix B)

$$\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)} = \frac{c_{\rho,\ell}^{(r),j} \exp\left\{ \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})] \right\}}{\sum_{\ell'} c_{\rho,\ell'}^{(r),j} \exp\left\{ \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho,\ell'}^{(r)})] \right\}}, \tag{18}$$

where the expected log-likelihood of a Gaussian $\mathcal{M}_{j,\rho,\ell}^{(r)}$ with respect to another Gaussian $\mathcal{M}_{i,\beta,m}^{(b)}$ is computable in closed-form (Penny and Roberts, 2000),

$$\begin{aligned}\mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})] = &-\frac{d}{2}\log 2\pi - \frac{1}{2}\log\left|\Sigma_{\rho,\ell}^{(r),j}\right| - \frac{1}{2}\mathrm{tr}\left((\Sigma_{\rho,\ell}^{(r),j})^{-1}\Sigma_{\beta,m}^{(b),i}\right) \\ &- \frac{1}{2}(\mu_{\rho,\ell}^{(r),j} - \mu_{\beta,m}^{(b),i})^T(\Sigma_{\rho,\ell}^{(r),j})^{-1}(\mu_{\rho,\ell}^{(r),j} - \mu_{\beta,m}^{(b),i}).\end{aligned}$$

HMM: For the HMM lower bound $\mathcal{L}_{HMM}^{i,j}$, we assume each variational distribution takes the form of a Markov chain,

$$q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) = \phi^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) = \phi_1^{i,j}(\rho_1|\beta_1)\prod_{t=2}^{\tau}\phi_t^{i,j}(\rho_t|\rho_{t-1},\beta_t),$$

where $\sum_{\rho_1=1}^{S}\phi_1^{i,j}(\rho_1|\beta_1) = 1$, and $\sum_{\rho_t=1}^{S}\phi_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) = 1$, and all the factors are non-negative. The variational distribution $q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})$ represents the probability of the state sequence $\boldsymbol{\rho}$ in HMM $\mathcal{M}_j^{(r)}$, when $\mathcal{M}_j^{(r)}$ is used to explain the *observation* sequence generated by $\mathcal{M}_i^{(b)}$ that evolved through state sequence $\boldsymbol{\beta}$.

Substituting $\phi^{i,j}$ into (16), the maximization with respect to $\phi_t^{i,j}(\rho_t|\rho_{t-1},\beta_t)$ and $\phi_1^{i,j}(\rho_1|\beta_1)$ is carried out independently for each pair (i,j), and follows (Hershey et al., 2007). This is further detailed in Appendix B. By separating terms and breaking up the summation over $\boldsymbol{\beta}$ and $\boldsymbol{\rho}$, the optimal $\hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t)$ and $\hat{\phi}_1^{i,j}(\rho_1|\beta_1)$ can be obtained using an efficient recursive iteration (similar to the forward algorithm).

H3M: For the H3M lower bound $\mathcal{L}_{H3M}^{i}$, we assume variational distributions of the form $q_i(z_i = j) = z_{ij}$, where $\sum_{j=1}^{K^{(r)}} z_{ij} = 1$, and $z_{ij} \geq 0$. Substituting z_{ij} into (15), and maximizing variational parameters are obtained as (see Appendix B)

$$\hat{z}_{ij} = \frac{\omega_j^{(r)} \exp(N_i \mathcal{L}_{HMM}^{i,j})}{\sum_{j'} \omega_{j'}^{(r)} \exp(N_i \mathcal{L}_{HMM}^{i,j'})}. \tag{19}$$

Note that in the standard HEM algorithm (Vasconcelos and Lippman, 1998; Chan et al., 2010a), the assignment probabilities z_{ij} are based on the expected log-likelihoods of the components, (e.g., $\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})]$ for H3Ms). For the variational HEM algorithm, these expectations are now replaced with their lower bounds (in our case, $\mathcal{L}_{HMM}^{i,j}$).

3.3.2 Lower Bound

Substituting the optimal variational distributions into (15), (16), and (17) gives the lower bounds,

$$\mathcal{L}^{i}_{H3M} = \sum_{j} \hat{z}_{ij} \left\{ \log \frac{\omega^{(r)}_{j}}{\hat{z}_{ij}} + N_i \mathcal{L}^{i,j}_{HMM} \right\}, \tag{20}$$

$$\mathcal{L}^{i,j}_{HMM} = \sum_{\boldsymbol{\beta}} \pi^{(b),i}_{\boldsymbol{\beta}} \sum_{\boldsymbol{\rho}} \hat{\phi}^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left\{ \log \frac{\pi^{(r),j}_{\boldsymbol{\rho}}}{\hat{\phi}^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \sum_{t} \mathcal{L}^{(i,\beta_t),(j,\rho_t)}_{GMM} \right\}, \tag{21}$$

$$\mathcal{L}^{(i,\beta),(j,\rho)}_{GMM} = \sum_{m=1}^{M} c^{(b),i}_{\beta,m} \sum_{\ell=1}^{M} \hat{\eta}^{(i,\beta),(j,\rho)}_{\ell|m} \left\{ \log \frac{c^{(r),j}_{\rho,\ell}}{\hat{\eta}^{(i,\beta),(j,\rho)}_{\ell|m}} + \mathrm{E}_{\mathcal{M}^{(b)}_{i,\beta,m}} [\log p(y|\mathcal{M}^{(r)}_{j,\rho,\ell})] \right\}. \tag{22}$$

The lower bound $\mathcal{L}^{i,j}_{HMM}$ requires summing over all sequences $\boldsymbol{\beta}$ and $\boldsymbol{\rho}$. This summation can be computed efficiently along with $\hat{\phi}^{i,j}_t(\rho_t|\rho_{t-1},\beta_t)$ and $\hat{\phi}^{i,j}_1(\rho_1|\beta_1)$ using a recursive algorithm from Hershey et al. (2007). This is described in Appendix B.

3.3.3 Summary Statistics

After calculating the optimal variational distributions, we calculate the following summary statistics, which are necessary for the M-step:

$$\nu^{i,j}_1(\rho_1,\beta_1) = \pi^{(b),i}_{\beta_1} \hat{\phi}^{i,j}_1(\rho_1|\beta_1),$$

$$\xi^{i,j}_t(\rho_{t-1},\rho_t,\beta_t) = \left(\sum_{\beta_{t-1}=1}^{S} \nu^{i,j}_{t-1}(\rho_{t-1},\beta_{t-1})\, a^{(b),i}_{\beta_{t-1},\beta_t} \right) \hat{\phi}^{i,j}_t(\rho_t|\rho_{t-1},\beta_t), \text{ for } t = 2,\ldots,\tau,$$

$$\nu^{i,j}_t(\rho_t,\beta_t) = \sum_{\rho_{t-1}=1}^{S} \xi^{i,j}_t(\rho_{t-1},\rho_t,\beta_t), \text{ for } t = 2,\ldots,\tau,$$

and the aggregate statistics

$$\hat{\nu}^{i,j}_1(\rho) = \sum_{\beta=1}^{S} \nu^{i,j}_1(\rho,\beta), \tag{23}$$

$$\hat{\nu}^{i,j}(\rho,\beta) = \sum_{t=1}^{\tau} \nu^{i,j}_t(\rho,\beta),$$

$$\hat{\xi}^{i,j}(\rho,\rho') = \sum_{t=2}^{\tau} \sum_{\beta=1}^{S} \xi^{i,j}_t(\rho,\rho',\beta).$$

The statistic $\hat{\nu}^{i,j}_1(\rho)$ is the expected number of times that the HMM $\mathcal{M}^{(r)}_j$ starts from state ρ, when modeling sequences generated by $\mathcal{M}^{(b)}_i$. The quantity $\hat{\nu}^{i,j}(\rho,\beta)$ is the expected number of times that the HMM $\mathcal{M}^{(r)}_j$ is in state ρ when the HMM $\mathcal{M}^{(b)}_i$ is in state β, when both HMMs are modeling sequences generated by $\mathcal{M}^{(b)}_i$. Similarly, the quantity $\hat{\xi}^{i,j}(\rho,\rho')$

is the expected number of transitions from state ρ to state ρ' of the HMM $\mathcal{M}_j^{(r)}$, when modeling sequences generated by $\mathcal{M}_i^{(b)}$.

3.4 M-Step

In the M-step, the lower bound in (11) is maximized with respect to the parameters $\mathcal{M}^{(r)}$,

$$\mathcal{M}^{(r)^*} = \operatorname*{argmax}_{\mathcal{M}^{(r)}} \sum_{i=1}^{K^{(b)}} \mathcal{L}_{H3M}^{i}.$$

The derivation of the maximization is presented in Appendix C. Each mixture component of $\mathcal{M}^{(r)}$ is updated independently according to

$$\omega_j^{(r)^*} = \frac{\sum_{i=1}^{K^{(b)}} \hat{z}_{i,j}}{K^{(b)}}, \tag{24}$$

$$\pi_\rho^{(r),j^*} = \frac{\sum_{i=1}^{K^{(b)}} \hat{z}_{i,j}\omega_i^{(b)}\hat{\nu}_1^{i,j}(\rho)}{\sum_{\rho'=1}^{S}\sum_{i=1}^{K^{(b)}} \hat{z}_{i,j}\omega_i^{(b)}\hat{\nu}_1^{i,j}(\rho'))}, \quad a_{\rho,\rho'}^{(r),j^*} = \frac{\sum_{i=1}^{K^{(b)}} \hat{z}_{i,j}\omega_i^{(b)}\hat{\xi}^{i,j}(\rho,\rho')}{\sum_{\sigma=1}^{S}\sum_{i=1}^{K^{(b)}} \hat{z}_{i,j}\omega_i^{(b)}\hat{\xi}^{i,j}(\rho,\sigma)}, \tag{25}$$

$$c_{\rho,\ell}^{(r),j^*} = \frac{\Omega_{j,\rho}\left(\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)}\right)}{\sum_{\ell'=1}^{M}\Omega_{j,\rho}\left(\hat{\eta}_{\ell'|m}^{(i,\beta),(j,\rho)}\right)}, \qquad \mu_{\rho,\ell}^{(r),j^*} = \frac{\Omega_{j,\rho}\left(\eta_{\ell|m}^{(i,\beta),(j,\rho)}\ \mu_{\beta,m}^{(b),i}\right)}{\Omega_{j,\rho}\left(\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)}\right)}, \tag{26}$$

$$\Sigma_{\rho,\ell}^{(r),j^*} = \frac{\Omega_{j,\rho}\left(\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)}\left[\Sigma_{\beta,m}^{(b),i} + (\mu_{\beta,m}^{(b),i} - \mu_{\rho,\ell}^{(r),j})(\mu_{\beta,m}^{(b),i} - \mu_{\rho,\ell}^{(r),j})^T\right]\right)}{\Omega_{j,\rho}\left(\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)}\right)}, \tag{27}$$

where $\Omega_{j,\rho}(\cdot)$ is the weighted sum operator over all base models, HMM states, and GMM components (i.e., over all tuples (i,β,m)),

$$\Omega_{j,\rho}(f(i,\beta,m)) = \sum_{i=1}^{K^{(b)}} \hat{z}_{i,j}\omega_i^{(b)} \sum_{\beta=1}^{S} \hat{\nu}^{i,j}(\rho,\beta) \sum_{m=1}^{M} c_{\beta,m}^{(b),i}\, f(i,\beta,m). \tag{28}$$

The terms $\pi_\rho^{(r),j}$ and $a_{\rho,\rho'}^{(r),j}$ are elements of the initial state prior and transition matrix, $\pi^{(r),j}$ and $A^{(r),j}$. Note that the covariance matrices of the reduced models in (27) include an additional outer-product term, which acts to regularize the covariances of the base models. This regularization effect derives from the E-step, which averages all possible observations from the base model.

4. Applications and Related Work

In the previous section, we derived the VHEM-H3M algorithm to cluster HMMs. We now discuss various applications of the algorithm (Section 4.1), and then present some literature that is related to HMM clustering (Section 4.2).

4.1 Applications of the VHEM-H3M Algorithm

The proposed VHEM-H3M algorithm clusters HMMs *directly* through the distributions they represent, and learns *novel* HMM cluster centers that compactly represent the structure of each cluster.

An application of the VHEM-H3M algorithm is in *hierarchical clustering* of HMMs. In particular, the VHEM-H3M algorithm is used recursively on the HMM cluster centers, to produce a bottom-up hierarchy of the input HMMs. Since the cluster centers condense the structure of the clusters they represent, the VHEM-H3M algorithm can implicitly leverage rich information on the underlying structure of the clusters, which is expected to impact positively the quality of the resulting hierarchical clustering.

Another application of VHEM is for efficient estimation of H3Ms from data, by using a *hierarchical estimation procedure* to break the learning problem into smaller pieces. First, a data set is split into small (non-overlapping) portions and intermediate HMMs are learned for each portion, via standard EM. Then, the final model is estimated from the intermediate models using the VHEM-H3M algorithm. Because VHEM and standard EM are based on similar maximum-likelihood principles, it drives model estimation towards similar optimal parameter values as performing EM estimation directly on the full data set. However, compared to direct EM estimation, VHEM-H3M is more memory- and time-efficient. First, it no longer requires storing in memory the entire data set during parameter estimation. Second, it does not need to evaluate the likelihood of all the samples at each iteration, and converges to effective estimates in shorter times. Note that even if a parallel implementation of EM could effectively handle the high memory requirements, a parallel-VHEM will still require fewer resources than a parallel-EM.

In addition, for the hierarchical procedure, the estimation of the intermediate models can be easily parallelized, since they are learned independently of each other. Finally, hierarchical estimation allows for efficient model updating when adding new data. Assuming that the previous intermediate models have been saved, re-estimating the H3M requires learning the intermediate models of only the new data, followed by running VHEM again. Since estimation of the intermediate models is typically as computationally intensive as the VHEM stage, reusing the previous intermediate models will lead to considerable computational savings when re-estimating the H3M.

In hierarchical estimation (EM on each time-series, VHEM on intermediate models), VHEM implicitly averages over all possible observations (virtual variations of each time-series) compatible with the intermediate models. We expect this to regularize estimation, which may result in models that generalize better (compared to estimating models with direct EM). Lastly, the "virtual" samples (i.e., sequences), which VHEM implicitly generates for maximum-likelihood estimation, need not be of the same length as the actual input data for estimating the intermediate models. Making the virtual sequences relatively short will positively impact the run time of each VHEM iteration. This may be achieved without loss of modeling accuracy, as show in Section 6.3.

4.2 Related Work

Jebara et al. (2007)'s approach to clustering HMMs consists of applying spectral clustering to a probability product kernel (PPK) matrix between HMMs—we will refer to it as PPK-

SC. In particular, the PPK similarity between two HMMs, $\mathcal{M}^{(a)}$ and $\mathcal{M}^{(b)}$, is defined as

$$k(a,b) = \int p(\mathbf{y}|\mathcal{M}^{(a)})^{\lambda}\, p(\mathbf{y}|\mathcal{M}^{(b)})^{\lambda} d\mathbf{y}, \tag{29}$$

where λ is a scalar, and τ is the length of "virtual" sequences. The case $\lambda = \frac{1}{2}$ corresponds to the Bhattacharyya affinity. While this approach indirectly leverages the probability distributions represented by the HMMs (i.e., the PPK affinity is computed from the probability distributions of the HMMs) and has proven successful in grouping HMMs into similar clusters (Jebara et al., 2007), it has several limitations. First, the spectral clustering algorithm cannot produce novel HMM cluster centers to represent the clusters, which is suboptimal for several applications of HMM clustering. For example, when implementing hierarchical clustering in the spectral embedding space (e.g., using hierarchical k-means clustering), clusters are represented by *single* points in the embedding space. This may fail to capture information on the local structure of the clusters that, when using VHEM-H3M, would be encoded by the novel HMM cluster centers. Hence, we expect VHEM-H3M to produce better hierarchical clustering than the spectral clustering algorithm, especially at higher levels of the hierarchy. This is because, when building a new level, VHEM can leverage more information from the lower levels, as encoded in the HMM cluster centers.

One simple extension of PPK-SC to obtain a HMM cluster center is to select the input HMM that the spectral clustering algorithm maps closest to the spectral clustering center. However, with this method, the HMM cluster centers are limited to be one of the *existing* input HMMs (i.e., similar to the k-medoids algorithm by Kaufman and Rousseeuw 1987), instead of the HMMs that optimally condense the structure of the clusters. Therefore, we expect the *novel* HMM cluster centers learned by VHEM-H3M to better represent the clusters. A more involved, "hybrid" solution is to learn the HMM cluster centers with VHEM-H3M *after* obtaining clusters with PPK-SC—using the VHEM-H3M algorithm to summarize all the HMMs within each PPK-SC cluster into a single HMM. However, we expect our VHEM-H3M algorithm to learn more accurate clustering models, since it jointly learns the clustering and the HMM centers, by optimizing a single objective function (i.e., the lower bound to the expected log-likelihood in Equation 11).

A second drawback of the spectral clustering algorithm is that the construction and the inversion of the similarity matrix between the input HMMs is a costly operation when their number is large[2] (e.g., see the experiment on H3M density estimation on the music data in Section 6.1). Therefore, we expect VHEM-H3M to be computationally more efficient than the spectral clustering algorithm since, by *directly* operating on the probability distributions of the HMMs, it does not require the construction of an initial embedding or any costly matrix operation on large kernel matrices.

Finally, as Jebara et al. (2004) note, the exact computation of (29) cannot be carried out efficiently, unless $\lambda = 1$. For different values of λ,[3] Jebara et al. (2004) propose to *approximate* $k(a,b)$ with an alternative kernel function that can be efficiently computed;

2. The computational complexity of large spectral clustering problems can be alleviated by means of numerical techniques for the solutions of eigenfunction problems such as the Nyström method (Nyström, 1930; Fowlkes et al., 2004), or by sampling only part of the similarity matrix and using a sparse eigensonver (Achlioptas et al., 2002).
3. The experimental results in Jebara et al. (2004) and Jebara et al. (2007) suggest to use $\lambda < 1$.

this alternative kernel function, however, is not guaranteed to be invariant to different but equivalent representations of the hidden state process (Jebara et al., 2004).[4] Alternative approximations for the Bhattacharyya setting (i.e., $\lambda = \frac{1}{2}$) have been proposed by Hershey and Olsen (2008).

Note that spectral clustering algorithms (similar to the one by Jebara et al. 2007) can be applied to kernel (similarity) matrices that are based on other affinity scores between HMM distributions than the PPK similarity of Jebara et al. (2004). Examples can be found in earlier work on HMM-based clustering of time-series, such as by Juang and Rabiner (1985), Lyngso et al. (1999), Bahlmann and Burkhardt (2001), Panuccio et al. (2002). In particular, Juang and Rabiner (1985) propose to approximate the (symmetrised) log-likelihood between two HMM distributions by computing the log-likelihood of real samples generated by one model under the other.[5] Extensions of the work of Juang and Rabiner (1985) have been proposed by Zhong and Ghosh (2003) and Yin and Yang (2005). In this work we do not pursue a comparison of the various similarity functions, but implement spectral clustering only based on PPK similarity (which Jebara et al. 2007 showed to be superior).

HMMs can also be clustered by sampling a number of time-series from each of the HMMs in the base mixture, and then applying the EM algorithm for H3Ms (Smyth, 1997), to cluster the time-series. Despite its simplicity, this approach would suffer from high memory and time requirements, especially when dealing with a large number of input HMMs. First, all generated samples need to be stored in memory. Second, evaluating the likelihood of the generated samples at each iteration is computationally intensive, and prevents the EM algorithm from converging to effective estimates in acceptable times.[6] On the contrary, VHEM-H3M is more efficient in computation and memory usage, as it replaces a costly sampling step (along with the associated likelihood computations at each iteration) with an expectation. An additional problem of EM with sampling is that, with a simple application of the EM algorithm, time-series generated from the same input HMM can be assigned to different clusters of the output H3M. As a consequence, the resulting clustering is not necessary *consistent*, since in this case the corresponding input HMM may not be clearly assigned to any single cluster. In our experiments, we circumvent this problem by defining appropriate constrains on the assignment variables.

The VHEM algorithm is similar in spirit to Bregman-clustering by Banerjee et al. (2005). Both algorithms base clustering on KL-divergence—the KL divergence and the expected

4. The kernel in (29) is computed by marginalizing out the hidden state variables, that is, $\int \left(\sum_{\mathbf{x}} p(\mathbf{y},\mathbf{x}|\mathcal{M}^{(a)})\right)^{\lambda} \left(\sum_{\mathbf{x}} p(\mathbf{y},\mathbf{x}|\mathcal{M}^{(b)})\right)^{\lambda} d\mathbf{y}$. This can be efficiently solved with the junction tree algorithm only when $\lambda = 1$. For $\lambda \neq 1$, Jebara et al. (2004) propose to use an alternative kernel $\tilde{k}$ that applies the power operation to the terms of the sum rather than the entire sum, where the terms are joint probabilities $p(\mathbf{y},\mathbf{x})$. I.e., $\tilde{k}(a,b) = \int \sum_{\mathbf{x}} \left(p(\mathbf{y},\mathbf{x}|\mathcal{M}^{(a)})\right)^{\lambda} \sum_{\mathbf{x}} \left(p(\mathbf{y},\mathbf{x}|\mathcal{M}^{(b)})\right)^{\lambda} d\mathbf{y}$.
5. For two HMM distributions, $\mathcal{M}^{(a)}$ and $\mathcal{M}^{(b)}$, Juang and Rabiner (1985) consider the affinity $L(a,b) = \frac{1}{2}\left[\log p(Y_b|\mathcal{M}^{(a)}) + p(Y_a|\mathcal{M}^{(b)})\right]$, where Y_a and Y_b are sets of observation sequences generated from $\mathcal{M}^{(a)}$ and $\mathcal{M}^{(b)}$, respectively.
6. In our experiments, EM on generated samples took two orders of magnitude more time than VHEM.

log-likelihood differ only for an entropy term that does not affect the clustering.[7] The main differences are: 1) in our setting, the expected log-likelihood (and KL divergence) is not computable in closed form, and hence VHEM uses an approximation; 2) VHEM-H3M clusters random *processes* (i.e., time series models), whereas Bregman-clustering (Banerjee et al., 2005) is limited to single random variables. Note that the number of virtual observations N allows to control the *peakiness* of the assignments $\hat{z}_{i,j}$. Limiting cases for $N \to \infty$ and $N = 1$ are similar to Bregman *hard* and *soft* cluttering, respectively (Goldberger and Roweis, 2004; Banerjee et al., 2005; Dhillon, 2007).

In the next two sections, we validate the points raised in this discussion through experimental evaluation using the VHEM-H3M algorithm. In particular, we consider clustering experiments in Section 5, and H3M density estimation for automatic annotation and retrieval in Section 6. Each application exploits some of the benefits of VHEM. First, we show that VHEM-H3M is more accurate in clustering than PPK-SC, in particular at higher levels of a hierarchical clustering (Section 5.2), and in an experiment with synthetic data (Section 5.3). Similarly, the annotation and retrieval results in Section 6 favor VHEM-H3M over PPK-SC and over standard EM, suggesting that VHEM-H3M is more robust and effective for H3M density estimation. Finally, in all the experiments, the running time of VHEM-H3M compares favorably with the other HMM clustering algorithms; PPK-SC suffers long delays when the number of input HMMs is large and the standard EM algorithm is considerably slower. This demonstrates that VHEM-H3M is most efficient for clustering HMMs.

5. Clustering Experiments

In this section, we present an empirical study of the VHEM-H3M algorithm for clustering and hierarchical clustering of HMMs. Clustering HMMs consists in partitioning K_1 input HMMs into $K_2 < K_1$ groups of similar HMMs. Hierarchical clustering involves organizing the input HMMs in a multi-level hierarchy with h levels, by applying clustering in a recursive manner. Each level ℓ of the hierarchy has K_ℓ groups (with $K_1 > K_2 > \cdots > K_{h-1} > K_h$), and the first level consists of the K_1 input HMMs.

We begin with an experiment on hierarchical clustering, where each of the input HMMs to be clustered is estimated on a sequence of motion capture data (Section 5.2). Then, we present a simulation study on clustering synthetic HMMs (Section 5.3). First, we provide an overview of the different algorithms used in this study.

5.1 Clustering Methods

In the clustering experiments, we will compare our VHEM-H3M algorithm with several other clustering algorithms. The various algorithms are summarized here.

- **VHEM-H3M**: We cluster K_1 input HMMs into K_2 clusters by using the VHEM-H3M algorithm (on the input HMMs) to learn a H3M with K_2 components (as explained

7. We can show that the VHEM algorithm performs clustering based on KL divergence. Letting $\mathcal{D}^{i,j}_{HMM} = \mathcal{L}^{i,i}_{HMM} - \mathcal{L}^{i,j}_{HMM} \approx D(\mathcal{M}^{(b)}_i || \mathcal{M}^{(r)}_j)$ be an approximation to the KL using (21), we can rewrite the E-step as $\hat{z}_{ij} \propto \omega^{(r)}_j e^{-N\omega^{(b)}_i \mathcal{D}^{i,j}_{HMM}}$. Similarly, the M-step is $\hat{\mathcal{M}}^{(r)}_j = \arg\min_{\mathcal{M}^{(r)}_j} \sum_{i=1}^{K^{(b)}} \omega^{(b)}_i \hat{z}_{ij} \mathcal{D}^{i,j}_{HMM}$.

in Section 3.1). To build a multi-level hierarchy of HMMs with h levels, we start from the first level of K_1 input HMMs, and recursively use the VHEM-H3M algorithm $h-1$ times. Each new level ℓ is formed by clustering the $K_{\ell-1}$ HMMs at the previous level into $K_\ell < K_{\ell-1}$ groups with the VHEM-H3M algorithm, and using the learned HMMs as cluster centers at the new level. In our experiments, we set the number of virtual samples to $N = 10^4 K^{(\ell-1)}$, a large value that favors "hard" clustering (where each HMM is univocally assigned to a single cluster), and the length of the virtual sequences to $\tau = 10$.

- **PPK-SC**: Jebara et al. (2007) cluster HMMs by calculating a PPK similarity matrix between all HMMs, and then applying spectral clustering. The work in Jebara et al. (2007) only considered HMMs with single Gaussian emissions, which did not always give satisfactory results in our experiments. Hence, we extended the method of Jebara et al. (2007) by allowing GMM emissions, and derived the PPK similarity for this more general case (Jebara et al., 2004). From preliminary experiments, we found the best performance for PPK with $\lambda = \frac{1}{2}$ (i.e., Bhattacharyya affinity), and when integrating over sequences of length $\tau = 10$. Finally, we also extend Jebara et al. (2007) to construct multi-level hierarchies, by using hierarchical k-means in the spectral clustering embedding.

- **SHEM-H3M**: This is a version of HEM-H3M that maximizes the likelihood of *actual* samples generated from the input HMMs, as in (5), rather than the expectation of virtual samples, as in (6). In particular, from each input HMM $\mathcal{M}_i^{(b)}$ we sample a set Y_i of $N_i = \pi_i^{(b)} N$ observation sequences (for a large value of N). We then estimate the reduced H3M from the N samples $Y = \{Y_i\}_{i=1}^{K^{(b)}}$, with the EM-H3M algorithm of Smyth (1997), which was modified to use a single assignment variable for each sample set Y_i, to obtain a consistent clustering.

In many real-life applications, the goal is to cluster a collection of time series, that is, observed sequences. Although the input data is not a collection of HMMs in that case, it can still be clustered with the VHEM-H3M algorithm by first modeling each sequence as an HMM, and then using the HMMs as input for the VHEM-H3M algorithm. With time-series data as input, it is also possible to use clustering approaches that do not model each sequence as a HMM. Hence, in one of the hierarchical motion clustering experiments, we also compare to the following two algorithms, one that clusters time-series data directly (Smyth, 1997), and a second one that clusters the time series after modeling each sequence with a dynamic texture (DT) model (Chan et al., 2010a).

- **EM-H3M**: The EM algorithm for H3Ms (Smyth, 1997) is applied directly on a collection of time series to learn the clustering and HMM cluster centers, thus bypassing the intermediate HMM modeling stage. To obtain a hierarchical clustering (with $h \geq 3$ levels), we proceed in a bottom up fashion and build each new level by simply re-clustering the given time series in a smaller number of clusters using the EM algorithm by Smyth (1997). We extend the algorithm to use a single assignment variable for each set of sequences Y_i that are within the same cluster in the immediately lower

level of the hierarchy. This modification preserves the hierarchical clustering property that sequences in a cluster will remain together at the higher levels.

- **HEM-DTM**: Rather than use HMMs, we consider a clustering model based on linear dynamical systems, that is, dynamic textures (DTs) (Doretto et al., 2003). Hierarchical clustering is performed using the hierarchical EM algorithm for DT mixtures (HEM-DTM) (Chan et al., 2010a), in an analogous way to VHEM-H3M. The main difference is that, with HEM-DTM, time-series are modeled as DTs, which have a *continuous* state space (a Gauss-Markov model) and *unimodal* observation model, whereas VHEM-H3M uses a *discrete* state space and *multimodal* observations (GMMs).

We will use several metrics to quantitatively compare the results of different clustering algorithms. First, we will calculate the *Rand-index* (Hubert and Arabie, 1985), which measures the correctness of a proposed clustering against a given ground truth clustering. Intuitively, this index measures how consistent cluster assignments are with the ground truth (i.e., whether pairs of items are correctly or incorrectly assigned to the same cluster, or different clusters). Second, we will consider the *log-likelihood*, as used by Smyth (1997) to evaluate a clustering. This measures how well the clustering fits the input data. When time series are given as input data, we compute the log-likelihood of a clustering as the sum of the log-likelihoods of each input sequence under the HMM cluster center to which it has been assigned. When the input data consists of HMMs, we will evaluate the log-likelihood of a clustering by using the expected log-likelihood of observations generated from an input HMM under the HMM cluster center to which it is assigned. For PPK-SC, the cluster center is estimated by running the VHEM-H3M algorithm (with $K^{(r)} = 1$) on the HMMs assigned to the cluster.[8] Note that the log-likelihood will be particularly appropriate to compare VHEM-H3M, SHEM-H3M, EM-H3M and HEM-DTM, since they explicitly optimize for it. However, it may be unfair for PPK-SC, since this method optimizes the PPK similarity and not the log-likelihood. As a consequence, we also measure the *PPK cluster-compactness*, which is more directly related to what PPK-SC optimizes for. The PPK cluster-compactness is the sum (over all clusters) of the average intra-cluster PPK pair-wise similarity. This performance metric favors methods that produce clusters with high intra-cluster similarity.

Note that, *time series* can also be clustered with recourse to similarity measures based on dynamic time warping (Oates et al., 1999; Keogh and Pazzani, 2000; Keogh and Ratanamahatana, 2005) or methods that rely on non-parametric sequence kernels (Leslie et al., 2002; Campbell, 2003; Kuksa et al., 2008; Cortes et al., 2008), which have shown good performance in practice. In this work we focus on the problem of clustering *hidden Markov models*, so we do not pursue an empirical evaluation of these methods.

5.2 Hierarchical Motion Clustering

In this experiment we test the VHEM algorithm on hierarchical motion clustering from motion capture data, that is, time series representing human locomotions and actions. To hierarchically cluster a collection of time series, we first model each time series with an HMM and then cluster the HMMs hierarchically. Since each HMM summarizes the appearance

8. Alternatively, we could use as cluster center the HMM mapped the closest to the spectral embedding cluster center, but this always resulted in lower log-likelihood.

(a) "Walk" sequence.

(b) "Run" sequence.

Figure 1: Examples of motion capture sequences from the MoCap data set, shown with stick figures.

and dynamics of the particular motion sequence it represents, the structure encoded in the hierarchy of HMMs directly applies to the original motion sequences. Jebara et al. (2007) uses a similar approach to cluster motion sequences, applying PPK-SC to cluster HMMs. However, they did not extend their study to hierarchies with multiple levels.

5.2.1 Data Sets and Setup

We experiment on two motion capture data sets, the MoCap data set (`http://mocap.cs.cmu.edu/`) and the Vicon Physical Action data set (Theodoridis and Hu, 2007; Asuncion and Newman, 2010). For the MoCap data set, we use 56 motion examples spanning 8 different classes ("jump", "run", "jog", "walk 1", "walk 2", "basket", "soccer", and "sit"). Each example is a sequence of 123-dimensional vectors representing the (x, y, z)-coordinates of 41 body markers tracked spatially through time. Figure 1 illustrates some typical examples. We built a hierarchy of $h = 4$ levels. The first level (Level 1) was formed by the $K_1 = 56$ HMMs learned from each individual motion example (with $S = 4$ hidden states, and $M = 2$ components for each GMM emission). The next three levels contain $K_2 = 8$, $K_3 = 4$ and $K_4 = 2$ HMMs. We perform the hierarchical clustering with VHEM-H3M, PPK-SC, EM-H3M, SHEM-H3M ($N \in \{560, 2800\}$ and $\tau = 10$), and HEM-DTM (state dimension of 7). The experiments were repeated 10 times for each clustering method, using different random initializations of the algorithms.

The Vicon Physical Action data set is a collection of 200 motion sequences. Each sequence consists of a time series of 27-dimensional vectors representing the (x, y, z)-coordinates of 9 body markers captured using the Vicon 3D tracker. The data set includes 10 normal and 10 aggressive activities, performed by each of 10 human subjects a single time. We build a hierarchy of $h = 5$ levels, starting with $K_1 = 200$ HMMs (with $S = 4$ hidden states and $M = 2$ components for each GMM emission) at the first level (i.e., one for each motion sequence), and using $K_2 = 20$, $K_3 = 8$, $K_4 = 4$, and $K_5 = 2$ for the next four levels. The experiment was repeated 5 times with VHEM-H3M and PPK-SC, using different random initializations of the algorithms.

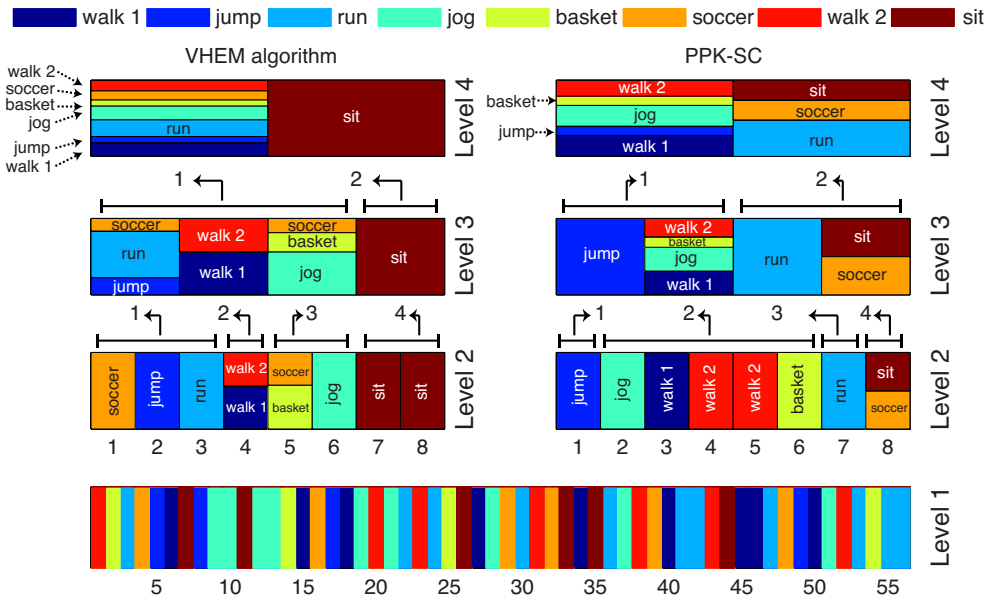


Figure 2: An example of hierarchical clustering of the MoCap data set, with VHEM-H3M and PPK-SC. Different colors represent different motion classes. Vertical bars represent clusters, with the colors indicating the proportions of the motion classes in a cluster, and the numbers on the x-axes representing the clusters' indexes. At Level 1 there are 56 clusters, one for each motion sequence. At Levels 2, 3 and 4 there are 8, 4 and 2 HMM clusters, respectively. For VHEM almost all clusters at Level 2 are populated by examples from a single motion class. The error of VHEM in clustering a portion of "soccer" with "basket" is probably because both actions involve a sequence of movement, shot, and pause. Moving up the hierarchy, the VHEM algorithm clusters similar motions classes together, and at Level 4 creates a dichotomy between "sit" and the other (more dynamic) motion classes. PPK-SC also clusters motion sequences well at Level 2, but incorrectly aggregates "sit" and "soccer", which have quite different dynamics. At Level 4, the clustering obtained by PPK-SC is harder to interpret than that by VHEM.

In similar experiments where we varied the number of levels h of the hierarchy and the number of clusters at each level, we noted similar relative performances of the various clustering algorithms, on both data sets.

5.2.2 Results on the MoCap Data Set

An example of hierarchical clustering of the MoCap data set with VHEM-H3M is illustrated in Figure 2 (left). In the first level, each vertical bar represents a motion sequence, with different colors indicating different ground-truth classes. In the second level, the $K_2 = 8$ HMM clusters are shown with vertical bars, with the colors indicating the proportions of the motion classes in the cluster. Almost all clusters are populated by examples from a single motion class (e.g., "run", "jog", "jump"), which demonstrates that VHEM can

	Rand-index			log-likelihood ($\times 10^6$)			PPK cluster-compactness			time (s)
Level	2	3	4	2	3	4	2	3	4	
VHEM-H3M	0.937	**0.811**	**0.518**	**-5.361**	**-5.682**	**-5.866**	0.0075	**0.0068**	**0.0061**	**30.97**
PPK-SC	**0.956**	0.740	0.393	-5.399	-5.845	-6.068	0.0082	0.0021	0.0008	37.69
SHEM-H3M (560)	0.714	0.359	0.234	-13.632	-69.746	-275.650	0.0062	0.0034	0.0031	843.89
SHEM-H3M (2800)	0.782	0.685	0.480	-14.645	-30.086	-52.227	0.0050	0.0036	0.0030	3849.72
EM-H3M	0.831	0.430	0.340	-5.713	-202.55	-168.90	**0.0099**	0.0060	0.0056	667.97
HEM-DTM	0.897	0.661	0.412	-7.125	-8.163	-8.532	-	-	-	121.32

Table 2: Hierarchical clustering of the MoCap data set using VHEM-H3M, PPK-SC, SHEM-H3M, EM-H3M and HEM-DTM. The number in brackets after SHEM-H3M represents the number of real samples used. We computed Rand-index, data log-likelihood and cluster compactness at each level of the hierarchy, and registered the time (in seconds) to learn the hierarchical structure. Differences in Rand-index at Levels 2, 3, and 4 are statistically significant based on a paired t-test with confidence 95%.

group similar motions together. We note an error of VHEM in clustering a portion of the "soccer" examples with "basket". This is probably caused by the fact that both types of actions begin with a stationary phase (e.g., subject focusing on the execution) followed with a forward movement (note that our "basket" examples correspond to forward dribbles). Moving up the hierarchy, the VHEM algorithm clusters similar motion classes together (as indicated by the arrows), for example "walk 1" and "walk 2" are clustered together at Level 2, and at the highest level (Level 4) it creates a dichotomy between "sit" and the rest of the motion classes. This is a desirable behavior as the kinetics of the "sit" sequences (which in the MoCap data set correspond to starting in a standing position, sitting on a stool, and returning to a standing position) are considerably different from the rest. On the right of Figure 2, the same experiment is repeated with PPK-SC. PPK-SC clusters motion sequences properly, but incorrectly aggregates "sit" and "soccer" at Level 2, even though they have quite different dynamics. Furthermore, the highest level (Level 4) of the hierarchical clustering produced by PPK-SC is harder to interpret than that of VHEM.

Table 2 presents a quantitative comparison between PPK-SC and VHEM-H3M at each level of the hierarchy. While VHEM-H3M has lower Rand-index than PPK-SC at Level 2 (0.937 vs. 0.956), VHEM-H3M has higher Rand-index at Level 3 (0.811 vs. 0.740) and Level 4 (0.518 vs. 0.393). In terms of PPK cluster-compactness, we observe similar results. In particular, VHEM-H3M has higher PPK cluster-compactness than PPK-SC at Level 3 and 4. Overall, keeping in mind that PPK-SC is explicitly driven by PPK-similarity, while the VHEM-H3M algorithm is not, these results can be considered as strongly in favor of VHEM-H3M (over PPK-SC). In addition, the data log-likelihood for VHEM-H3M is higher than that for PPK-SC at each level of the hierarchy. This suggests that the novel HMM cluster centers learned by VHEM-H3M fit the motion capture data better than the spectral cluster centers, since they condense information of the entire underlying clusters. This conclusion is further supported by the results of the density estimation experiments in Sections 6.1 and 6.2. Note that the higher up in the hierarchy, the more clearly this effect is manifested.

Comparing to other methods (also in Table 2), EM-H3M generally has lower Rand-index than VHEM-H3M and PPK-SC (consistent with the results from Jebara et al. (2007)). While EM-H3M directly clusters the original motion sequences, both VHEM-H3M and PPK-SC implicitly integrate over all possible virtual variations of the original motion sequences (according to the intermediate HMM models), which results in more robust clustering procedures. In addition, EM-H3M has considerably longer running times than VHEM-H3M and PPK-SC (i.e., roughly 20 times longer) since it needs to evaluate the likelihood of all training sequences at each iteration, at all levels.

The results in Table 2 favor VHEM-H3M over SHEM-H3M, and empirically validate the variational approximation that VHEM uses for learning. For example, when using $N = 2800$ samples, running SHEM-H3M takes over two orders of magnitude more time than VHEM-H3M, but still does not achieve performance competitive with VHEM-H3M. With an efficient closed-form expression for averaging over all possible virtual samples, VHEM approximates the sufficient statistics of a virtually unlimited number of observation sequences, without the need of using real samples. This has an additional regularization effect that improves the robustness of the learned HMM cluster centers. In contrast, SHEM-H3M uses real samples, and requires a large number of them to learn accurate models, which results in significantly longer running times.

In Table 2, we also report hierarchical clustering performance for HEM-DTM. VHEM-H3M consistently outperforms HEM-DTM, both in terms of Rand-index and data log-likelihood.[9] Since both VHEM-H3M and HEM-DTM are based on the hierarchical EM algorithm for learning the clustering, this indicates that HMM-based clustering models are more appropriate than DT-based models for the human MoCap data. Note that, while PPK-SC is also HMM-based, it has a lower Rand-index than HEM-DTM at Level 4. This further suggests that PPK-SC does not optimally cluster the HMMs.

Finally, to assess stability of the clustering results, starting from the 56 HMMs learned on the motion examples, in turn we exclude one of them and build a hierarchical clustering of the remaining ones ($h = 4$ levels, $K_1 = 55, K_2 = 8, K_3 = 4, K_4 = 2$), using in turn VHEM-H3M and PPK-SC. We then compute cluster stability as the mean Rand-index between all possible pairs of clusterings. The experiment is repeated 10 times for both VHEM-H3M and PPK-SC, using a different random initialization for each trial. The stability of VHEM-H3M is 0.817, 0.808 and 0.950, at Levels 2, 3 and 4. The stability of PPK-SC is 0.786, 0.837 and 0.924, at Levels 2, 3 and 4. Both methods are fairly stable in terms of Rand-index, with a slight advantage for VHEM-H3M over PPK-SC (with average across levels of 0.858 versus 0.849).

The experiments were repeated 10 times for each clustering method, using different random initializations of the algorithms.

Alternatively, we evaluate the out-of-sample generalization of the clusterings discovered by VHEM-H3M and PPK-SC, by computing the fraction of times the held out HMMs is assigned[10] to the cluster that contains the majority of HMMs from its same ground truth

9. We did not report PPK cluster-compactness for HEM-DTM, since it would not be directly comparable with the same metric based on HMMs.

10. For VHEM-H3M we use the expected log-likelihood, for PPK-SC we use the out-of-sample extension for spectral clustering of Bengio et al. (2004).

	Rand-index				log-likelihood ($\times 10^6$)				PPK cluster-compactness			
Level	2	3	4	5	2	3	4	5	2	3	4	5
VHEM-H3M	0.909	0.805	0.610	0.244	-1.494	-3.820	-5.087	-6.172	0.224	0.059	0.020	0.005
PPK-SC	0.900	0.807	0.452	0.092	-3.857	-5.594	-6.163	-6.643	0.324	0.081	0.026	0.008

Table 3: Hierarchical clustering of the Vicon Physical Action data set using VHEM-H3M and PPK-SC. Performance is measured in terms of Rand-index, data log-likelihood and PPK cluster-compactness at each level. Differences in Rand-index at Levels 2, 4 and 5 are statistically significant based on a paired t-test with confidence 95%. The test failed at Level 3.

class. Out of sample generalization of VHEM-H3M and PPK-SC are comparable, registering an averages of 0.875, respectively, 0.851 across the different levels.

5.2.3 Results on the Vicon Physical Action Data Set

Table 3 presents results using VHEM-H3M and PPK-SC to cluster the Vicon Physical Action data set. While the two algorithms performs similarly in terms of Rand-index at lower levels of the hierarchy (i.e., Level 2 and Level 3), at higher levels (i.e., Level 4 and Level 5) VHEM-H3M outperforms PPK-SC. In addition, VHEM-H3M registers higher data log-likelihood than PPK-SC at each level of the hierarchy. This, again, suggests that by learning new cluster centers, the VHEM-H3M algorithm retains more information on the clusters' structure than PPK-SC. Finally, compared to VHEM-H3M, PPK-SC produces clusters that are more compact in terms of PPK similarity. However, this does not necessarily imply a better agreement with the ground truth clustering, as evinced by the Rand-index metrics.

5.3 Clustering Synthetic Data

In this experiment, we compare VHEM-H3M and PPK-SC on clustering a synthetic data set of HMMs.

5.3.1 Data Set and Setup

The synthetic data set of HMMs is generated as follows. Given a set of C HMMs $\{\mathcal{M}^{(c)}\}_{c=1}^{C}$, for each HMM we synthesize a set of K "noisy" versions of the original HMM. Each "noisy" HMM $\tilde{\mathcal{M}}_k^{(c)}$ $(k = 1, \dots K)$ is synthesized by generating a random sequence $y_{1:T}$ of length T from $\mathcal{M}^{(c)}$, corrupting it with Gaussian noise $\sim \mathcal{N}(0, \sigma_n^2 \mathrm{I}_d)$, and estimating the parameters of $\tilde{\mathcal{M}}_k^{(c)}$ on the corrupted version of $y_{1:T}$. Note that this procedure adds noise in the *observation* space. The number of noisy versions (of each given HMM), K, and the noise variance, σ_n^2, will be varied during the experiments.

The collection of original HMMs was created as follows. Their number was always set to $C = 4$, the number of hidden states of the HMMs to $S = 3$, the emission distributions to be single, one-dimensional Gaussians (i.e., GMMs with $M = 1$ component), and the length of the sequences to $T = 100$. For all original HMMs $\mathcal{M}^{(c)}$, *if not otherwise specified*, the initial state probability, the state transition matrix, and the means and variances of the

emission distributions were fixed as

$$\pi^{(c)} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}, \quad A^{(c)} = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.2 & 0.8 & 0 \\ 0 & 0.2 & 0.8 \end{bmatrix}, \quad \begin{cases} \mu_1^{(c)} = 1 \\ \mu_2^{(c)} = 2 \\ \mu_3^{(c)} = 3 \end{cases} \quad {\sigma_\rho^{(c)}}^2 = 0.5 \ \forall \rho, \quad \forall c.$$

We consider three different experimental settings. In the first setting, experiment (a), the HMMs $\mathcal{M}^{(c)}$ only differ in the means of the emission distributions,

$$\begin{cases} \mu_1^{(1)} = 1 \\ \mu_2^{(1)} = 2 \\ \mu_3^{(1)} = 3 \end{cases}, \begin{cases} \mu_1^{(2)} = 3 \\ \mu_2^{(2)} = 2 \\ \mu_3^{(2)} = 1 \end{cases}, \begin{cases} \mu_1^{(3)} = 1 \\ \mu_2^{(3)} = 2 \\ \mu_3^{(3)} = 2 \end{cases}, \begin{cases} \mu_1^{(4)} = 1 \\ \mu_2^{(4)} = 3 \\ \mu_3^{(4)} = 3 \end{cases}$$

In the second setting, experiment (b), the HMMs differ in the variances of the emission distributions,

$${\sigma_\rho^{(1)}}^2 = 0.5, \quad {\sigma_\rho^{(2)}}^2 = 0.1, \quad {\sigma_\rho^{(3)}}^2 = 1, \quad {\sigma_\rho^{(4)}}^2 = 0.05, \quad \forall \rho.$$

In the last setting, experiment (c), the HMMs differ in the transition matrices,

$$A^{(1)} = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.2 & 0.8 & 0 \\ 0 & 0.2 & 0.8 \end{bmatrix} \ A^{(2)} = \begin{bmatrix} 0.2 & 0.2 & 0.2 \\ 0.4 & 0.6 & 0 \\ 0 & 0.4 & 0.6 \end{bmatrix} \ A^{(3)} = \begin{bmatrix} 0.9 & 0.05 & 0.05 \\ 0.1 & 0.9 & 0 \\ 0 & 0.1 & 0.9 \end{bmatrix} \ A^{(4)} = \begin{bmatrix} 0.4 & 0.3 & 0.4 \\ 0.6 & 0.4 & 0 \\ 0 & 0.6 & 0.4 \end{bmatrix}.$$

The VHEM-H3M and PPK-SC algorithms are used to cluster the synthesized HMMs, $\{\{\tilde{\mathcal{M}}_k^{(c)}\}_{k=1}^K\}_{c=1}^C$, into C groups, and the quality of the resulting clusterings is measured with the Rand-index, PPK cluster-compactness, and the expected log-likelihood of the discovered cluster centers with respect to the original HMMs. The expected log-likelihood was computed using the lower bound, as in (13), with each of the original HMMs assigned to the most likely HMM cluster center. The results are averages over 10 trials.

The reader is referred to Appendix E for experiments where the order of the model used for clustering does not match the order of the true model used for generating the data.

5.3.2 Results

Figure 3 reports the performance metrics when varying the number $K \in \{2, 4, 8, 16, 32\}$ of noisy versions of each of the original HMMs, and the noise variance $\sigma_n^2 \in \{0.1, 0.5, 1\}$, for the three experimental settings. (Note that, in each trial, for each class, we first generated 32 noisy HMMs per class, and then, varying $K \in \{4, 8, 16, 32\}$, we subsampled only K of them.) For the majority of settings of K and σ_n^2, the clustering produced by VHEM-H3M is superior to the one produced by PPK-SC, for each of the considered metrics (i.e., in the plots, solid lines are usually above dashed lines of the same color). The only exception is in experiment (b) where, for low noise variance (i.e., $\sigma_n^2 = 0.1$) PPK-SC is the best in terms of Rand-index and cluster compactness. It is interesting to note that the gap in performance between VHEM-H3M and PPK-SC is generally larger at low values of K. We believe this is because, when only a limited number of input HMMs is available, PPK-SC produces an

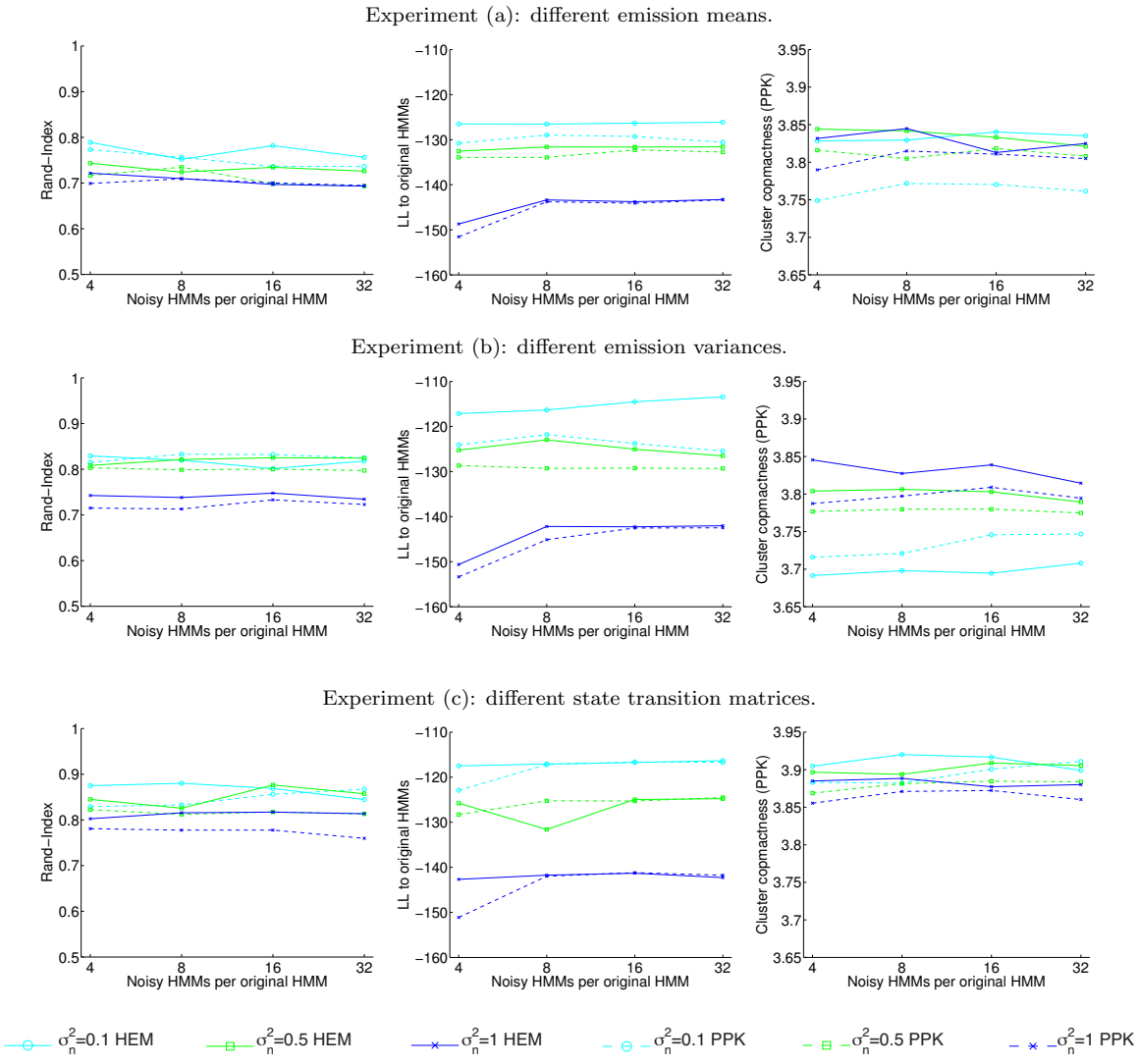


Figure 3: Results on clustering synthetic data with VHEM-H3M and PPK-SC. Performance is measured in terms of Rand-index, expected log-likelihood and PPK cluster-compactness.

embedding of lower quality. This does not affect VHEM-H3M, since it clusters in HMM distribution space and does not use an embedding.

Note that the Rand-Index values for experiment (c) (i.e., different state transition matrices) are superior to the corresponding ones in experiments (a) and (b) (i.e., different emission distributions) for both VHEM-H3M and PPK-SC. This shows that VHEM-H3M (and slightly less robustly PPK-SC as well) can cluster dynamics characterized by different hidden states processes more easily than dynamics that differ only in the emission distri-

butions. In addition, VHEM-H3M is more robust to noise than PPK-SC, as demonstrated by the experiments with $\sigma_n^2 = 1$ (blue lines).

These results suggest that, by clustering HMMs *directly* in distribution space, VHEM-H3M is generally more robust than PPK-SC, the performance of which instead depends on the quality of the underlying embedding.

Finally, the results in Figure 3 show only small fluctuations across different values of K at each noise value, suggesting stability of the clustering results (*relative to the ground truth clustering*) for both VHEM-H3M and PPK-SC, to both subsampling and jittering (i.e., addition of noise) (Hennig, 2007). In particular, from the expected log-likelihood values (central column in Figure 3), we evince that the similarity of the discovered clustering to the true distribution (i.e., the original HMMs) is not largely affected by the amount of subsampling, except when only as little as the 12.5% of the data is used (when $K = 4$).

6. Density Estimation Experiments

In this section, we present an empirical study of VHEM-H3M for density estimation, in automatic annotation and retrieval of music (Section 6.1) and hand-written digits (Section 6.2).

6.1 Music Annotation and Retrieval

In this experiment, we evaluate VHEM-H3M for estimating annotation models in content-based music auto-tagging. As a generative time-series model, H3Ms allow to account for timbre (i.e., through the GMM emission process) as well as longer term temporal dynamics (i.e., through the HMM hidden state process), when modeling musical signals. Therefore, in music annotation and retrieval applications, H3Ms are expected to prove more effective than existing models that do not explicitly account for temporal information (Turnbull et al., 2008; Mandel and Ellis, 2008; Eck et al., 2008; Hoffman et al., 2009).

6.1.1 Music Data Set

We consider the CAL500 collection from Turnbull et al. (2008), which consists of 502 songs and provides binary annotations with respect to a vocabulary $\mathcal{V}$ of 149 tags, ranging from genre and instrumentation, to mood and usage. To represent the acoustic content of a song we extract a time series of audio features $\mathcal{Y} = \{y_1, \ldots, y_{|\mathcal{Y}|}\}$, by computing the first 13 Mel frequency cepstral coefficients (MFCCs) (Rabiner and Juang, 1993) over half-overlapping windows of 46ms of audio signal, augmented with first and second instantaneous derivatives. The song is then represented as a collection of *audio fragments*, which are sequences of $T = 125$ audio features (approximately 6 seconds of audio), using a dense sampling with 80% overlap.

6.1.2 Music Annotation Models

Automatic music tagging is formulated as a supervised multi-label problem (Carneiro et al., 2007), where each class is a tag from $\mathcal{V}$. We approach this problem by modeling the audio content for each tag with a H3M probability distribution. I.e., for each tag, we estimate an H3M over the audio fragments of the songs in the database that have been associated

with that tag, using the hierarchical estimation procedure based on VHEM-H3M. More specifically, the database is first processed at the song level, using the EM algorithm to learn a H3M with $K^{(s)} = 6$ components for each song[11] from its audio fragments. Then, for each tag, the song-level H3Ms labeled with that tag are pooled together to form a large H3M, and the VHEM-H3M algorithm is used to reduce this to a final H3M tag-model with $K = 3$ components ($\tau = 10$ and $N = N_v N_t K^{(s)}$, where $N_v = 1000$ and N_t is the number of training songs for the particular tag).

Given the tag-level models, a song can be represented as a vector of posterior probabilities of each tag (a semantic multinomial, SMN), by extracting features from the song, computing the likelihood of the features under each tag-level model, and applying Bayes' rule.[12] A test song is annotated with the top-ranking tags in its SMN. To retrieve songs given a tag query, a collection of songs is ranked by the tag's probability in their SMNs.

We compare VHEM-H3M with three alternative algorithms for estimating the H3M tag models: PPK-SC, PPK-SC-hybrid, and EM-H3M.[13] For all three alternatives, we use the same number of mixture components in the tag models ($K = 3$). For the two PPK-SC methods, we leverage the work of Jebara et al. (2007) to learn H3M tag models, and use it in place of the VHEM-H3M algorithm in the second stage of the hierarchical estimation procedure. We found that it was necessary to implement the PPK-SC approaches with song-level H3Ms with only $K^{(s)} = 1$ component (i.e., a single HMM), since the computational cost for constructing the initial embedding scales poorly with the number of input HMMs.[14] PPK-SC first applies spectral clustering to the song-level HMMs and then selects as the cluster centers the HMMs that map closest to the spectral cluster centers in the spectral embedding. PPK-SC-hybrid is a hybrid method combining PPK-SC for clustering, and VHEM-H3M for estimating the cluster centers. Specifically, after spectral clustering, HMM cluster centers are estimated by applying VHEM-H3M (with $K^{(r)} = 1$) to the HMMs assigned to each of the resulting clusters. In other words, PPK-SC and PPK-SC-hybrid use spectral clustering to summarize a collection of song-level HMMs with a few HMM centers, forming a H3M tag model. The mixture weight of each HMM component (in the H3M tag model) is set proportional to the number of HMMs assigned to that cluster.

For EM-H3M, the H3M tag models were estimated directly from the audio fragments from the relevant songs using the EM-H3M algorithm.[15] Empirically, we found that, due

11. Most pop songs have 6 structural parts: intro, verse, chorus, solo, bridge and outro.
12. We compute the likelihood of a song under a tag model as the geometric average of the likelihood of the individual segments, and further normalized it by the length of the segments to prevent the posteriors from being too "peaked" (Coviello et al., 2011).
13. For this experiment, we were not able to successfully estimate accurate H3M tag models with SHEM-H3M. In particular, SHEM-H3M requires generating an appropriately large number of real samples to produce accurate estimates. However, due to the computational limits of our cluster, we were able to test SHEM-H3M only using a small number of samples. In preliminary experiments we registered performance only slightly above chance level and training times still twice longer than for VHEM-H3M. For a comparison between VHEM-H3M and SHEM-H3M on density estimation, the reader can refer to the experiment in Section 6.2 on online hand-writing classification and retrieval.
14. Running PPK-SC with $K^{(s)} = 2$ took 3958 hours in total (about 4 times more than when setting $K^{(s)} = 1$), with no improvement in annotation and retrieval performance. A larger $K^{(s)}$ would yield impractically long learning times.
15. The EM algorithm has been used to estimate HMMs from music data in previous work (Scaringella and Zoia, 2005; Reed and Lee, 2006).

to its runtime and RAM requirements, for EM-H3M we must use non-overlapping audio-fragments and evenly subsample by 73% on average, resulting in 14.5% of the sequences used by VHEM-H3M. Note that, however, EM-H3M is still using 73% of the actual song data (just with non-overlapping sequences). We believe this to be a reasonable comparison between EM and VHEM, as both methods use roughly similar resources (the sub-sampled EM is still 3 times slower, as reported in Table 4). Based on our projections, running EM over densely sampled song data would require roughly 9000 hours of CPU time (e.g., more than 5 weeks when parallelizing the algorithm over 10 processors), as opposed to 630 hours for VHEM-H3M. This would be extremely cpu-intensive given the computational limits of our cluster. The VHEM algorithm, on the other hand, can learn from considerable amounts of data while still maintaining low runtime and memory requirements.[16]

Finally, we also compare against two state-of-the-art models for music tagging, HEM-DTM (Coviello et al., 2011), which is based on a different time-series model (mixture of dynamic textures), and HEM-GMM (Turnbull et al., 2008), which is a bag-of-features model using GMMs. Both methods use efficient hierarchical estimation based on a HEM algorithm (Chan et al., 2010a; Vasconcelos and Lippman, 1998) to obtain the tag-level models.[17]

6.1.3 Performance Metrics

A test song is annotated with the 10 most likely tags, and annotation performance is measured with the per-tag precision (P), recall (R), and F-score (F), averaged over all tags. If $|w_H|$ is the number of test songs that have the tag w in their ground truth annotations, $|w_A|$ is the number of times an annotation system uses w when automatically tagging a song, and $|w_C|$ is the number of times w is correctly used, then precision, recall and F-score for the tag w are defined as:

$$\mathrm{P} = \frac{|w_C|}{|w_A|}, \;\; \mathrm{R} = \frac{|w_C|}{|w_H|}, \;\; \mathrm{F} = 2\left((\mathrm{P})^{-1} + (\mathrm{R})^{-1}\right)^{-1}.$$

Retrieval is measured by computing per-tag mean average precision (MAP) and precision at the first k retrieved songs (P@k), for $k \in \{5, 10, 15\}$. The P@k is the fraction of true positives in the top-k of the ranking. MAP averages the precision at each point in the ranking where a song is correctly retrieved. All reported metrics are averages over the 97 tags that have at least 30 examples in CAL500 (11 genre, 14 instrument, 25 acoustic quality, 6 vocal characteristics, 35 emotion and 6 usage tags), and are the result of 5-fold cross-validation.

Finally, we also record the total time (in hours) to learn the 97 tag-level H3Ms on the 5 splits of the data. For hierarchical estimation methods (VHEM-H3M and the PPK-SC approaches), this also includes the time to learn the song-level H3Ms.

16. For example, consider learning a tag-level H3M from 200 songs, which corresponds to over 3GB of audio fragments. Using the hierarchical estimation procedure, we first model each song (in average, 15MB of audio fragments) individually as a song-level H3M, and we save the song models (150 KB of memory each). Then, we pool the 200 song models into a large H3M (in total 30MB of memory), and reduce it to a smaller tag-level H3M using the VHEM-H3M algorithm.

17. Both auto-taggers operate on audio features extracted over half-overlapping windows of 46ms. HEM-GMM uses MFCCs with first and second instantaneous derivatives (Turnbull et al., 2008). HEM-DTM uses 34-bins of Mel-spectral features (Coviello et al., 2011), which are further grouped in audio fragments of 125 consecutive features.

	annotation			retrieval				
	P	R	F	MAP	P@5	P@10	P@15	time (h)
VHEM-H3M	0.446	0.211	0.260	0.440	0.474	0.451	0.435	629.5
PPK-SC	0.299	0.159	0.151	0.347	0.358	0.340	0.329	974.0
PPK-SC-*hybrid*	0.407	0.200	0.221	0.415	0.439	0.421	0.407	991.7
EM-H3M	0.415	0.214	0.248	0.423	0.440	0.422	0.407	1860.4
HEM-DTM	0.431	0.202	0.252	0.439	0.479	0.454	0.428	-
HEM-GMM	0.374	0.205	0.213	0.417	0.441	0.425	0.416	-

Table 4: Annotation and retrieval performance on CAL500, for VHEM-H3M, PPK-SC, PPK-SC-*hybrid*, EM-H3M, HEM-DTM (Coviello et al., 2011) and HEM-GMM (Turnbull et al., 2008).

6.1.4 Results

In Table 4 we report the performance of the various algorithms for both annotation and retrieval on the CAL500 data set. Looking at the overall runtime, VHEM-H3M is the most efficient algorithm for estimating H3M distributions from music data, as it requires only 34% of the runtime of EM-H3M, and 65% of the runtime of PPK-SC. The VHEM-H3M algorithm capitalizes on the first stage of song-level H3M estimation (about one third of the total time) by efficiently and effectively using the song-level H3Ms to learn the final tag models. Note that the runtime of PPK-SC corresponds to setting $K^{(s)} = 1$. When we set $K^{(s)} = 2$, we registered a running time four times longer, with no significant improvement in performance.

The gain in computational efficiency does not negatively affect the quality of the corresponding models. On the contrary, VHEM-H3M achieves better performance than EM-H3M,[18] strongly improving the top of the ranked lists, as evinced by the higher P@k scores. Relative to EM-H3M, VHEM-H3M has the benefit of regularization, and during learning can efficiently leverage all the music data condensed in the song H3Ms. VHEM-H3M also outperforms both PPK-SC approaches on all metrics. PPK-SC discards considerable information on the clusters' structure by selecting one of the original HMMs to approximate each cluster. This significantly affects the accuracy of the resulting annotation models. VHEM-H3M, on the other hand, generates novel HMM cluster centers to summarize the clusters. This allows to retain more accurate information in the final annotation models.

PPK-SC-hybrid achieves considerable improvements relative to standard PPK-SC, at relatively low additional computational costs.[19] This further demonstrates that the VHEM-H3M algorithm can effectively summarize in a smaller model the information contained in *several* HMMs. In addition, we observe that VHEM-H3M still outperforms PPK-SC-hybrid, suggesting that the former produces more accurate cluster centers and density estimates.

18. The differences in performance are statistically significant based on a paired t-test with 95% confidence.
19. In PPK-SC-hybrid, each run of the VHEM-H3M algorithm converges quickly since there is only one HMM component to be learned, and can benefit from clever initialization (i.e., to the HMM mapped the closest to the spectral clustering center).

In fact, VHEM-H3M *couples* clustering and learning HMM cluster centers, and is entirely based on maximum likelihood for estimating the H3M annotation models. PPK-SC-hybrid, on the contrary, separates clustering and parameter estimation, and optimizes them against two different metrics (i.e., PPK similarity and expected log-likelihood, respectively). As a consequence, the two phases may be mismatched, and the centers learned with VHEM may not be the best representatives of the clusters according to PPK affinity.

Finally, VHEM-H3M compares favorably to the auto-taggers based on other generative models. First, VHEM-H3M outperforms HEM-GMM, which does not model temporal information in the audio signal, on all metrics. Second, the performances of VHEM-H3M and HEM-DTM (a continuous-state temporal model) are not statistically different based on a paired t-test with 95% confidence, except for annotation precision where VHEM-H3M scores significantly higher. Since HEM-DTM is based on linear dynamic systems (a continuous-state model), it can model stationary time-series in a linear subspace. In contrast, VHEM-H3M uses HMMs with discrete states and GMM emissions, and can hence better adapt to non-stationary time-series on a non-linear manifold. This difference is illustrated in the experiments: VHEM-H3M outperforms HEM-DTM on the human MoCap data (see Table 2), which has non-linear dynamics, while the two perform similarly on the music data (see Table 4), where audio features are often stationary over short time frames.

6.2 On-Line Hand-Writing Data Classification and Retrieval

In this experiment, we investigate the performance of the VHEM-H3M algorithm in estimating class-conditional H3M distributions for automatic classification and retrieval of on-line hand-writing data.

6.2.1 Data Set

We consider the Character Trajectories Data Set (Asuncion and Newman, 2010), which is a collection of 2858 examples of characters from the same writer, originally compiled to study handwriting motion primitives (Williams et al., 2006).[20] Each example is the trajectory of one of the 20 different characters that correspond to a single pen-down segment. The data was captured from a WACOM tablet at 200 Hz, and consists of (x, y)-coordinates and pen tip force. The data has been numerically differentiated and Gaussian smoothed (Williams et al., 2006). Half of the data is used for training, with the other half held out for testing.

6.2.2 Classification Models and Setup

From the hand-writing examples in the training set, we estimate a series of class-conditional H3M distributions, one for each character, using hierarchical estimation with the VHEM-H3M algorithm. First, for each character, we partition all the relevant training data into groups of 3 sequences, and learn a HMM (with $S = 4$ states and $M = 1$ component for the GMM emissions) from each group using the Baum-Welch algorithm. Next, we estimate the class-conditional distribution (classification model) for each character by aggregating all the relevant HMMs and summarizing them into a H3M with $K = 4$ components using the

20. Even if we limit this experiment to the data set of Williams et al. (2006), we would like to refer the interest readers to the data set of Liwicki and Bunke (2005), which is a collection of *word* trajectories (as opposed to *character* trajectories).

VHEM-H3M algorithm ($\tau = 10$ and $N = N_v N_c$, where $N_v = 10$,[21] and N_c is the number of intermediate models for that character). Using the character-level H3Ms and Bayes' rule, for each hand-writing example in the test set we compute the posterior probabilities of all of the 20 characters. Each example is classified as the character with largest posterior probability. For retrieval given a query character, examples in the test set are ranked by the character's posterior probability.

We repeated the same experiment using PPK-SC or SHEM-H3M ($\tau = 10$, $N = 1000$) to estimate the classification models from the intermediate HMMs. Finally, we considered the EM-H3M algorithm, which directly uses the training sequences to learn the class-conditional H3M ($K = 4$).

Since VHEM-H3M, SHEM-H3M and EM-H3M are *iterative* algorithms, we studied them when varying the stopping criterion. In particular, the algorithms were terminated when the relative variation in the value of the objective function between two consecutive iterations was lower than a threshold ΔLL, which we varied in $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$.[22]

Finally, we measure classification and retrieval performance on the test set using the classification accuracy, and the average per-tag P@3, P@5 and MAP. We also report the total training time, which includes the time used to learn the intermediate HMMs. The experiments consisted of 5 trials with different random initializations of the algorithms.

6.2.3 Results

Table 5 lists the classification and retrieval performance on the test set for the various methods. Consistent with the experiments on music annotation and retrieval (Section 6.1), VHEM-H3M performs better than PPK-SC on all metrics. By learning novel HMM cluster centers, VHEM-H3M estimates H3M distributions that are representative of all the relevant intermediate HMMs, and hence of all the relevant training sequences. While EM-H3M is the best in classification (at the price of longer training times), VHEM-H3M performs better in retrieval, as evinced by the P@3 and P@5 scores. In terms of training time, VHEM-H3M and PPK-SC are about 5 times faster than EM-H3M. In particular, PPK-SC is the fastest algorithm, since the small number of input HMMs (i.e., on average 23 per character) allows to build the spectral clustering embedding efficiently.

The version of HEM based on actual sampling (SHEM-H3M) performs better than VHEM-H3M in classification, but VHEM-H3M has higher retrieval scores. However, the training time for SHEM-H3M is approximately 15 times longer than for VHEM-H3M. These differences in performance can be understood in terms of the different types of *approximation* used by SHEM-H3M and VHEM-H3M. The sampling operation of SHEM-H3M provides an *unbiased* estimator, whose *variance* depends on the number of samples used. Hence, in order to reliably estimate the reduced model (i.e., making the variance small), the SHEM-

21. Note that choosing a lower value of N_v (compared to the music experiments) plays a role in making the clustering algorithm more *reliable*. Using fewer virtual samples equates to attaching smaller "virtual probability masses" to the input HMMs, and leads to *less certain* assignments of the input HMMs to the clusters (cf. Equation 19). This determines more mixing in the initial iterations of the algorithm (e.g., similar to higher annealing temperature), and reduces the risk of prematurely specializing any cluster to one of the original HMMs. This effect is desirable, since the input HMMs are estimated over a smaller number of sequences (compared to the music experiments) and can therefore be noisier and less reliable.
22. In a similar experiment where we used the number of iterations as the stopping criterion, we registered similar results.

	stop	**retrieval**			**classification**	
	ΔLL	P@3	P@5	MAP	Accuracy	total time (s)
VHEM-H3M	10^{-2}	0.750	0.750	0.738	0.618	1838.34
	10^{-3}	0.717	0.750	0.741	0.619	1967.55
	10^{-4}	0.733	0.790	0.773	0.648	2210.77
	10^{-5}	0.750	0.820	0.775	0.651	2310.93
SHEM-H3M	10^{-2}	0.417	0.440	0.517	0.530	13089.32
	10^{-3}	0.683	0.680	0.728	0.664	23203.44
	10^{-4}	0.700	0.750	0.753	0.689	35752.20
	10^{-5}	0.700	0.750	0.754	0.690	50094.36
EM-H3M	10^{-2}	0.583	0.610	0.667	0.646	6118.53
	10^{-3}	0.617	0.650	0.731	0.674	7318.56
	10^{-4}	0.650	0.710	0.756	0.707	9655.08
	10^{-5}	0.517	0.560	0.665	0.635	10957.38
PPK-SC	-	0.600	0.700	0.698	0.646	1463.54

Table 5: Classification and retrieval performance, and training time on the Character Trajectories Data Set, for VHEM-H3M, PPK-SC, SHEM-H3M, and EM-H3M.

H3M algorithm requires generating and handling a sufficiently large number of samples (relative to the model order/complexity, Hastie et al. 2005). In contrast, the variational approximation of VHEM-H3M does not require generating samples (and hence avoids the problem of large variances associate to relatively small samples), but introduces a bias in the estimator (Gunawardana and Byrne, 2005). Hence, it is not surprising that, on a relatively less complex problem where a relatively smaller sample size suffices, SHEM-H3M can perform more robustly than VHEM-H3M (even if still at a larger computational cost).

It is also interesting to note that EM-H3M appears to suffer from overfitting of the training set, as suggested by the *overall drop* in performance when the stopping criterion changes from $\Delta LL = 10^{-4}$ to $\Delta LL = 10^{-5}$. In contrast, both VHEM-H3M and SHEM-H3M consistently improve on all metrics as the algorithm converges (again looking at $\Delta LL \in \{10^{-4}, 10^{-5}\}$). These results suggest that the regularization effect of hierarchical estimation, which is based on averaging over *more* samples (either virtual or actual), can positively impact the generalization of the learned models.[23]

Finally, we elaborate on how these results compare to the experiments on music annotation and retrieval (in Section 6.1). First, in the Character Trajectory Data Set the number of training sequences associated with each class (i.e, each character) is small compared to the CAL500 data set.[24] As a result, the EM-H3M algorithm is able to process all the data, and achieve good classification performance. However, EM-H3M still needs to evaluate the

23. For smaller values of ΔLL (e.g., $\Delta LL < 10^{-5}$), the performance of EM-H3M did not improve.

24. In the Character Trajectory data set there are on average 71 training sequences per character. In CAL500, each tag is associated with *thousands* of training sequences at the song level (e.g., an average of about 8000 audio fragments per tag).

	$N_v = 1000$				$\tau = 10$			
	$\tau = 2$	$\tau = 5$	$\tau = 10$	$\tau = 20$	$N_v = 1$	$N_v = 10$	$N_v = 100$	$N_v = 1000$
P@5	0.4656	0.4718	0.4738	0.4734	0.4775	0.4689	0.4689	0.4738
P@10	0.4437	0.4487	0.4507	0.4478	0.4534	0.4491	0.4466	0.4507
P@15	0.4236	0.4309	0.4346	0.4327	0.4313	0.4307	0.4242	0.4346

Table 6: Annotation and retrieval performance on CAL500 for VHEM-H3M when varying the virtual sample parameters N_v and τ.

likelihood of all the original sequences at each iteration. This leads to slower iterations, and results in a total training time about 5 times longer than that of VHEM-H3M (see Table 5). Second, the Character Trajectory data is more "controlled" than the CAL500 data, since each class corresponds to a single character, and all the examples are from the same writer. As a consequence, there is less variation in the intermediate HMMs (i.e., they are clustered more closely), and several of them may summarize the cluster well, providing good candidate cluster centers for PPK-SC. In conclusion, PPK-SC faces only a limited loss of information when selecting one of the initial HMMs to represent each cluster, and achieves reasonable performances.

6.3 Robustness of VHEM-H3M to Number and Length of Virtual Samples

The generation of virtual samples in VHEM-H3M is controlled by two parameters: the number of virtual sequences (N), and their length (τ). In this section, we investigate the impact of these parameters on annotation and retrieval performance on CAL500. For a given tag t, we set $N = N_v N_t K^{(s)}$, where N_v is a constant, N_t the number of training songs for the tag, and $K^{(s)}$ the number of mixture components for each song-level H3M. Starting with $(N_v, \tau) = (1000, 10)$, each parameter is varied while keeping the other one fixed, and annotation and retrieval performance on the CAL500 data set are calculated, as described in Section 6.1.

Table 6 presents the results, for $\tau \in \{2, 5, 10, 20\}$ and $N_v \in \{1, 10, 100, 1000\}$. The performances when varying τ are close on all metrics. For example, average P@5, P@10 and P@15 vary in small ranges (0.0082, 0.0070 and 0.0110, respectively). Similarly, varying the number of virtual sequences N_v has a limited impact on performance as well.[25] This demonstrates that VHEM-H3M is fairly robust to the choice of these parameters.

Finally, we tested VHEM-H3M for music annotation and retrieval on CAL500, using virtual sequences of the same length as the audio fragments used at the song level, that is, $\tau = T = 125$. Compared to $\tau = 10$ (the setting used in earlier experiments), we registered an 84% increase in total running time, with no corresponding improvement in performance. Thus, in our experimental setting, making the virtual sequences relatively short positively impacts the running time, without reducing the quality of the learned models.

25. Note that the E-step of the VHEM-H3M algorithm averages over all possible observations compatible with the input models, also when we choose a low value of N_v (e.g., $N_v = 1$). The number of virtual samples controls the "virtual mass" of each input HMMs and thus the certainty of cluster assignments.

7. Conclusion

In this paper, we presented a variational HEM (VHEM) algorithm for clustering HMMs with respect to their probability distributions, while generating a novel HMM center to represent each cluster. Experimental results demonstrate the efficacy of the algorithm on various clustering, annotation, and retrieval problems involving time-series data, showing improvement over current methods. In particular, using a two-stage, *hierarchical estimation* procedure—learn H3Ms on many smaller subsets of the data, and summarize them in a more compact H3M model of the data—the VHEM-H3M algorithm estimates annotation models from data more efficiently than standard EM and also improves model robustness through better regularization. Specifically, averaging over all possible virtual samples prevents over-fitting, which can improve the generalization of the learned models. Moreover, using relatively short virtual sequences positively impacts the running time of the algorithm, without negatively affecting its performance on practical applications. In addition, we have noted that the VHEM-H3M algorithm is robust to the choice of the number and length of virtual samples.

In our experiments, we have implemented the first stage of the hierarchical estimation procedure by partitioning data in *non-overlapping* subsets (and learning an intermediate H3M on each subset). In particular, partitioning the CAL500 data at the song level had a practical advantage. Since individual songs in CAL500 are relevant to several tags, the estimation of the song H3Ms can be executed one single time for each song in the database, and *re-used* in the VHEM estimation of all the associated tag models. This has a positive impact on computational efficiency. Depending on the particular application, however, a slightly different implementation of this first stage (of the hierarchical estimation procedure) may be better suited. For example, when estimating a H3M from a very large amount of training data, one could use a procedure that does not necessarily cover all data, inspired by Kleiner et al. (2011). If n is the size of the training data, first estimate $B > 1$ intermediate H3Ms on as many (possibly overlapping) "little" bootstrap subsamples of the data,[26] each of size $b < n$. Then summarize all the intermediate H3Ms into a final H3M using the VHEM-H3M algorithm.

In future work we plan to extend VHEM-H3M to the case where all HMMs share a large GMM universal background model for the emission distributions (with each HMM state having a different set of weights for the Gaussian components), which is commonly used in speech (Huang and Jack, 1989; Bellegarda and Nahamoo, 1990; Rabiner and Juang, 1993) or in hand-writing recognition (Rodriguez-Serrano and Perronnin, 2012). This would allow for faster training (moving the complexity to estimating the noise background model) and would require a faster implementation of the inference (e.g., using a strategy inspired by Coviello et al. (2012b)). In addition, we plan to derive a HEM algorithm for HMMs with discrete emission distributions, and compare its performance to the work presented here and to the extension with the large GMM background model.

Finally, in this work we have not addressed the model selection problem, that is, selecting the number of reduced mixture components. Since VHEM is based on maximum likelihood principles, it is possible to apply standard statistical model selection techniques, such as

26. Several techniques have been proposed to bootstrap from sequences of samples, for example refer to Hall et al. (1995).

Bayesian information criterion (BIC) and Akaike information criterion (AIC) (MacKay, 2003). Alternatively, inspired by Bayesian non-parametric statistics, the VHEM formulation could be extended to include a Dirichlet process prior (Blei and Jordan, 2006), with the number of components adapting to the data.

Acknowledgments

E.C. thanks Yingbo Song for providing the code for the PPK similarity between HMMs (Jebara et al., 2007) and assistance with the experiments on motion clustering. The authors acknowledge support from Google, Inc. E.C. and G.R.G.L. also wish to acknowledge support from Qualcomm, Inc., Yahoo!, Inc., KETI under the PHTM program, and NSF Grants CCF-0830535 and IIS-1054960. A.B.C. was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (CityU 110513). G.R.G.L. acknowledges support from the Alfred P. Sloan Foundation. This research was supported in part by the UCSD FWGrid Project, NSF Research Infrastructure Grant Number EIA-0303622.

Appendix A. Derivation of the Lower Bounds

The lower bounds are derived as follow.

A.1 Lower Bound on $\mathrm{E}_{\mathcal{M}_i^{(b)}}\left[\log p(Y_i|\mathcal{M}^{(r)})\right]$

The lower bound (15) on $\mathrm{E}_{\mathcal{M}_i^{(b)}}\left[\log p(Y_i|\mathcal{M}^{(r)})\right]$ is computed by introducing the variational distribution $q_i(z_i)$ and applying (10)

$$
\begin{aligned}
&\mathrm{E}_{\mathcal{M}_i^{(b)}}\left[\log p(Y_i|\mathcal{M}^{(r)})\right] \\
&\geq \max_{q_i} \sum_j q_i(z_i=j)\left\{\log\frac{p(z_i=j|\mathcal{M}^{(r)})}{q_i(z_i=j)} + \mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(Y_i|\mathcal{M}_j^{(r)})]\right\} \\
&= \max_{q_i} \sum_j q_i(z_i=j)\left\{\log\frac{p(z_i=j|\mathcal{M}^{(r)})}{q_i(z_i=j)} + \mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})^{N_i}]\right\} \qquad (30) \\
&= \max_{q_i} \sum_j q_i(z_i=j)\left\{\log\frac{p(z_i=j|\mathcal{M}^{(r)})}{q_i(z_i=j)} + N_i\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})]\right\} \qquad (31) \\
&\geq \max_{q_i} \sum_j q_i(z_i=j)\left\{\log\frac{p(z_i=j|\mathcal{M}^{(r)})}{q_i(z_i=j)} + N_i\mathcal{L}_{HMM}^{i,j}\right\} \triangleq \mathcal{L}_{H3M}^{i}, \qquad (32)
\end{aligned}
$$

where in (30) we use the fact that Y_i is a set of N_i i.i.d. samples. In (31), $\log p(\mathbf{y}|\mathcal{M}_j^{(r)})$ is the observation log-likelihood of an HMM, which is essentially a mixture distribution. Since the latter expectation cannot be calculated directly, in (32) we use instead the lower bound $\mathcal{L}_{HMM}^{i,j}$ defined in (13).

A.2 Lower Bound on $\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})]$

To calculate the lower bound $\mathcal{L}_{HMM}^{i,j}$,starting with (13), we first rewrite the expectation $\mathrm{E}_{\mathcal{M}_i^{(b)}}$ to explicitly marginalize over the HMM state sequence $\boldsymbol{\beta}$ from $\mathcal{M}_i^{(b)}$,

$$\begin{aligned}\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})] &= \mathrm{E}_{\boldsymbol{\beta}|\mathcal{M}_i^{(b)}}\left[\mathrm{E}_{\mathbf{y}|\boldsymbol{\beta},\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})]\right] \\ &= \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \mathrm{E}_{\mathbf{y}|\boldsymbol{\beta},\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})]. \end{aligned} \tag{33}$$

We introduce a variational distribution $q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})$ on the state sequence $\boldsymbol{\rho}$, which depends on a particular sequence $\boldsymbol{\beta}$ from $\mathcal{M}_i^{(b)}$. Applying (10) to (33), we have

$$\begin{aligned}&\mathrm{E}_{\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\mathcal{M}_j^{(r)})] \\ &\geq \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \max_{q^{i,j}} \sum_{\boldsymbol{\rho}} q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left\{ \log \frac{p(\boldsymbol{\rho}|\mathcal{M}_j^{(r)})}{q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \mathrm{E}_{\mathbf{y}|\boldsymbol{\beta},\mathcal{M}_i^{(b)}}[\log p(\mathbf{y}|\boldsymbol{\rho},\mathcal{M}_j^{(r)})] \right\} \\ &= \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \max_{q^{i,j}} \sum_{\boldsymbol{\rho}} q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left\{ \log \frac{p(\boldsymbol{\rho}|\mathcal{M}_j^{(r)})}{q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \sum_t \mathrm{E}_{\mathcal{M}_{i,\beta_t}^{(b)}}[\log p(y_t|\mathcal{M}_{j,\rho_t}^{(r)})] \right\} \qquad (34) \\ &\geq \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \max_{q^{i,j}} \sum_{\boldsymbol{\rho}} q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left\{ \log \frac{p(\boldsymbol{\rho}|\mathcal{M}_j^{(r)})}{q^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \sum_t \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \right\} \triangleq \mathcal{L}_{HMM}^{i,j}, \qquad (35)\end{aligned}$$

where in (34) we use the conditional independence of the observation sequence given the state sequence, and in (35) we use the lower bound, defined in (14), on each expectation.

A.3 Lower Bound on $\mathrm{E}_{\mathcal{M}_{i,\beta}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho}^{(r)})]$

To derive the lower bound $\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}$ for (14), we first rewrite the expectation with respect to $\mathcal{M}_{i,\beta}^{(b)}$ to explicitly marginalize out the GMM hidden assignment variable ζ,

$$\begin{aligned}\mathrm{E}_{\mathcal{M}_{i,\beta}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho}^{(r)})] &= \mathrm{E}_{\zeta|\mathcal{M}_{i,\beta}^{(b)}}\left[\mathrm{E}_{\mathcal{M}_{i,\beta,\zeta}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho}^{(r)})]\right] \\ &= \sum_{m=1}^{M} c_{\beta,m}^{(b),i} \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}\left[\log p(y|\mathcal{M}_{j,\rho}^{(r)})\right].\end{aligned}$$

Note that $p(y|\mathcal{M}_{j,\rho}^{(r)})$ is a GMM emission distribution as in (1). Hence, the observation variable is y, and the hidden variable is ζ. Therefore, we introduce the variational distribution $q_{\beta,\rho}^{i,j}(\zeta|m)$, which is conditioned on the observation y arising from the mth component in $\mathcal{M}_{i,\beta}^{(b)}$, and apply (10),

$$\begin{aligned}&\mathrm{E}_{\mathcal{M}_{i,\beta}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho}^{(r)})] \\ &\geq \sum_{m=1}^{M} c_{\beta,m}^{(b),i} \max_{q_{\beta,\rho}^{i,j}} \sum_{\zeta=1}^{M} q_{\beta,\rho}^{i,j}(\zeta|m) \left\{ \log \frac{p(\zeta|\mathcal{M}_{j,\rho}^{(r)})}{q_{\beta,\rho}^{i,j}(\zeta|m)} + \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}}[\log p(y|\mathcal{M}_{j,\rho,\zeta}^{(r)})] \right\} \triangleq \mathcal{L}_{GMM}^{(i,\beta),(j,\rho)}.\end{aligned}$$

Appendix B. Derivation of the E-Step

GMM: Substituting the variational distribution $\eta_{\ell|m}^{(i,\beta),(j,\rho)}$ into (17), we have

$$\mathcal{L}_{GMM}^{(i,\beta),(j,\rho)} = \sum_{m=1}^{M} c_{\beta,m}^{(b),i} \max_{\eta_{\ell|m}^{(i,\beta),(j,\rho)}} \sum_{\ell=1}^{M} \eta_{\ell|m}^{(i,\beta),(j,\rho)} \left\{ \log \frac{c_{\rho,\ell}^{(r),j}}{\eta_{\ell|m}^{(i,\beta),(j,\rho)}} + \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}} [\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})] \right\}.$$

The maximizing variational parameters are obtained as (see Appendix D.2)

$$\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)} = \frac{c_{\rho,\ell}^{(r),j} \exp\left\{ \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}} [\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})] \right\}}{\sum_{\ell'} c_{\rho,\ell'}^{(r),j} \exp\left\{ \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}} [\log p(y|\mathcal{M}_{j,\rho,\ell'}^{(r)})] \right\}},$$

HMM: Substituting the variational distribution $\phi^{i,j}$ into (16), we have

$$\mathcal{L}_{HMM}^{i,j} = \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \max_{\phi^{i,j}} \sum_{\boldsymbol{\rho}} \phi^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left\{ \log \frac{\pi_{\boldsymbol{\rho}}^{(r),j}}{\phi^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \sum_{t} \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \right\}. \tag{36}$$

The maximization of (36) with respect to $\phi_t^{i,j}(\rho_t|\rho_{t-1},\beta_t)$ and $\phi_1^{i,j}(\rho_1|\beta_1)$ is carried out independently for each pair (i,j), and follow Hershey et al. (2007). In particular it uses a backward recursion, starting with $\mathcal{L}_{\tau+1}^{i,j}(\beta_t,\rho_t) = 0$, for $t = \tau,\ldots,2$,

$$\hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) = \frac{a_{\rho_{t-1},\rho_t}^{(r),j} \exp\left\{ \mathcal{L}_{\mathrm{GMM}}^{(i,\beta_t),(j,\rho_t)} + \mathcal{L}_{t+1}^{i,j}(\beta_t,\rho_t) \right\}}{\sum_{\rho}^{S} a_{\rho_{t-1},\rho}^{(r),j} \exp\left\{ \mathcal{L}_{\mathrm{GMM}}^{(i,\beta_t),(j,\rho)} + \mathcal{L}_{t+1}^{i,j}(\beta_t,\rho) \right\}}$$

$$\mathcal{L}_t^{i,j}(\beta_{t-1},\rho_{t-1}) = \sum_{\beta=1}^{S} a_{\beta_{t-1},\beta}^{(b),i} \log \sum_{\rho=1}^{S} a_{\rho_{t-1},\rho}^{(r),j} \exp\left\{ \mathcal{L}_{\mathrm{GMM}}^{(i,\beta),(j,\rho)} + \mathcal{L}_{t+1}^{i,j}(\beta,\rho) \right\},$$

and terminates with

$$\hat{\phi}_1^{i,j}(\rho_1|\beta_1) = \frac{\pi_{\rho_1}^{(r),j} \exp\left\{ \mathcal{L}_{\mathrm{GMM}}^{(i,\beta_1),(j,\rho_1)} + \mathcal{L}_2^{i,j}(\beta_1,\rho_1) \right\}}{\sum_{\rho}^{S} \pi_{\rho}^{(r),j} \exp\left\{ \mathcal{L}_{\mathrm{GMM}}^{(i,\beta_1),(j,\rho)} + \mathcal{L}_2^{i,j}(\beta_1,\rho) \right\}}$$

$$\mathcal{L}_{\mathrm{HMM}}^{i,j} = \sum_{\beta=1}^{S} \pi_{\beta}^{(b),i} \log \sum_{\rho=1}^{S} \pi_{\rho}^{(r),j} \exp\left\{ \mathcal{L}_{\mathrm{GMM}}^{(i,\beta),(j,\rho)} + \mathcal{L}_2^{i,j}(\beta,\rho) \right\} \tag{37}$$

where (37) is the maxima of the terms in (36) in Section 3.3.1.
H3M: Substituting the variational distribution z_{ij} into (15), we have

$$\mathcal{L}_{H3M}^{i} = \max_{z_{ij}} \sum_{j} z_{ij} \left\{ \log \frac{\omega_j^{(r)}}{z_{ij}} + N_i \mathcal{L}_{HMM}^{i,j} \right\}. \tag{38}$$

The maximizing variational parameters of (38) are obtained using Appendix D.2,

$$\hat{z}_{ij} = \frac{\omega_j^{(r)} \exp(N_i \mathcal{L}_{HMM}^{i,j})}{\sum_{j'} \omega_{j'}^{(r)} \exp(N_i \mathcal{L}_{HMM}^{i,j'})}.$$

Appendix C. Derivation of the M-Step

The M-steps involves maximizing the lower bound in (11) with respect to $\mathcal{M}^{(r)}$, while holding the variational distributions fixed,

$$\mathcal{M}^{(r)^*} = \operatorname*{argmax}_{\mathcal{M}^{(r)}} \sum_{i=1}^{K^{(b)}} \mathcal{L}^i_{H3M}. \tag{39}$$

Substituting (20) and (21) into the objective function of (39),

$$\begin{aligned}\mathcal{L}(\mathcal{M}^{(r)}) &= \sum_{i=1}^{K^{(b)}} \mathcal{L}^i_{H3M} \\ &= \sum_{i,j} \hat{z}_{ij} \left\{ \log \frac{\omega_j^{(r)}}{\hat{z}_{ij}} + N_i \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \sum_{\boldsymbol{\rho}} \hat{\phi}^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \left[\log \frac{\pi_{\boldsymbol{\rho}}^{(r),j}}{\hat{\phi}^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta})} + \sum_t \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \right] \right\}\end{aligned} \tag{40}$$

In the following, we detail the update rules for the parameters of the reduced model $\mathcal{M}^{(r)}$.

C.1 HMMs Mixture Weights

Collecting terms in (40) that only depend on the mixture weights $\{\omega_j^{(r)}\}_{j=1}^{K^{(r)}}$, we have

$$\widetilde{\mathcal{L}}(\{\omega_j^{(r)}\}) = \sum_i \sum_j \hat{z}_{ij} \log \omega_j^{(r)} = \sum_j \left[\sum_i \hat{z}_{ij} \right] \log \omega_j^{(r)} \tag{41}$$

Given the constraints $\sum_{j=1}^{K^{(r)}} \omega_j^{(r)} = 1$ and $\omega_j^{(r)} \geq 0$, (41) is maximized using the result in Appendix D.1, which yields the update in (24).

C.2 Initial State Probabilities

The objective function in (40) factorizes for each HMM $\mathcal{M}_j^{(r)}$, and hence the parameters of each HMM are updated independently. For the j-th HMM, we collect terms in (40) that depend on the initial state probabilities $\{\pi_\rho^{(r),j}\}_{\rho=1}^{S}$,

$$\begin{aligned}\widetilde{\mathcal{L}}_j(\{\pi_\rho^{(r),j}\}) &= \sum_i \hat{z}_{ij} N_i \sum_{\beta_1} \pi_{\beta_1}^{(b),i} \sum_{\rho_1} \hat{\phi}_1^{i,j}(\rho_1|\beta_1) \log \pi_{\rho_1}^{(r),j} \\ &= \sum_{\rho_1} \sum_i \hat{z}_{ij} N_i \underbrace{\sum_{\beta_1} \pi_{\beta_1}^{(b),i} \hat{\phi}_1^{i,j}(\rho_1|\beta_1)}_{\hat{\nu}_1^{i,j}(\rho_1)} \log \pi_{\rho_1}^{(r),j} \\ &= \sum_\rho \sum_i \hat{z}_{ij} N_i \hat{\nu}_1^{i,j}(\rho) \log \pi_\rho^{(r),j} \qquad (42) \\ &\propto \sum_\rho \left[\sum_i \hat{z}_{ij} \omega_i^{(b)} \hat{\nu}_1^{i,j}(\rho) \right] \log \pi_\rho^{(r),j}, \qquad (43)\end{aligned}$$

where in the (42) we have used the summary statistic defined in (23). Considering the constraints $\sum_{\rho=1}^{S} \pi_{\rho}^{(r),j} = 1$ and $\pi_{\rho}^{(r),j} \geq 0$, (43) is maximized using the result in Appendix D.1, giving the update formula in (25).

C.3 State Transition Probabilities

Similarly, for each HMM $\mathcal{M}_j^{(r)}$ and previous state ρ, we collect terms in (40) that depend on the transition probabilities $\{a_{\rho,\rho'}^{(r),j}\}_{\rho'=1}^{S}$,

$$
\begin{aligned}
\widetilde{\mathcal{L}}_{j,\rho}(\{a_{\rho,\rho'}^{(r),j}\}_{\rho'=1}^{S}) &= \sum_i \hat{z}_{ij} N_i \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \sum_{\boldsymbol{\rho}} \hat{\phi}^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \log \pi_{\boldsymbol{\rho}}^{(r),j} \\
&\propto \sum_i \hat{z}_{ij} N_i \sum_{\boldsymbol{\beta}} \left[\pi_{\beta_1}^{(b),i} \prod_{t=2}^{\tau} a_{\beta_{t-1},\beta_t}^{(b),i} \right] \sum_{\boldsymbol{\rho}} \left[\hat{\phi}_1^{i,j}(\rho_1|\beta_1) \prod_{t=2}^{\tau} \hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) \right] \left[\sum_{t=2}^{\tau} \log a_{\rho_{t-1},\rho_t}^{(r),j} \right] \\
&= \sum_i \hat{z}_{ij} N_i \sum_{\rho_1} \sum_{\rho_2} \sum_{\beta_2} \underbrace{\sum_{\beta_1} \underbrace{\pi_{\beta_1}^{(b),i} \hat{\phi}_1^{i,j}(\rho_1|\beta_1)}_{\nu_1^{i,j}(\rho_1,\beta_1)} a_{\beta_1,\beta_2}^{(b),i} \hat{\phi}_2^{i,j}(\rho_2|\rho_1,\beta_2)}_{\xi_2^{i,j}(\rho_1,\rho_2,\beta_2)} \Big[\log a_{\rho_1,\rho_2}^{(r),j} \\
&\quad + \sum_{\beta_3 \cdots \beta_\tau} \sum_{\rho_3 \cdots \rho_\tau} \prod_{t=3}^{\tau} a_{\beta_{t-1},\beta_t}^{(b),i} \prod_{t=3}^{\tau} \hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) \sum_{t=3}^{\tau} \log a_{\rho_{t-1},\rho_t}^{(r),j} \Big] \\
&= \sum_i \hat{z}_{ij} N_i \sum_{\rho_1} \sum_{\rho_2} \sum_{\beta_2} \xi_2^{i,j}(\rho_1,\rho_2,\beta_2) \log a_{\rho_1,\rho_2}^{(r),j} \\
&\quad + \sum_i \hat{z}_{ij} N_i \sum_{\rho_2} \sum_{\rho_3} \sum_{\beta_3} \underbrace{\sum_{\beta_2} \underbrace{\sum_{\rho_1} \xi_2^{i,j}(\rho_1,\rho_2,\beta_2)}_{\nu_2^{i,j}(\rho_2,\beta_2)} a_{\beta_2,\beta_3}^{(b),i} \hat{\phi}_3^{i,j}(\rho_3|\rho_2,\beta_3)}_{\xi_3^{i,j}(\rho_2,\rho_3,\beta_3)} \Big[\log a_{\rho_2,\rho_3}^{(r),j} \\
&\quad + \sum_{\beta_4 \cdots \beta_\tau} \sum_{\rho_4 \cdots \rho_\tau} \prod_{t=4}^{\tau} a_{\beta_{t-1},\beta_t}^{(b),i} \prod_{t=4}^{\tau} \hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) \sum_{t=4}^{\tau} \log a_{\rho_{t-1},\rho_t}^{(r),j} \Big] \\
&= \dots \\
&= \sum_i \hat{z}_{ij} N_i \sum_{t=2}^{\tau} \sum_{\rho_{t-1}} \sum_{\rho_t} \sum_{\beta_t} \xi_t^{i,j}(\rho_{t-1},\rho_t,\beta_t) \log a_{\rho_{t-1},\rho_t}^{(r),j} \\
&\propto \sum_i \hat{z}_{ij} N_i \sum_{\rho'} \underbrace{\sum_{t=2}^{\tau} \sum_{\beta} \xi_t^{i,j}(\rho,\rho',\beta)}_{\hat{\xi}^{i,j}(\rho,\rho')} \log a_{\rho,\rho'}^{(r),j} \\
&\propto \sum_{\rho'=1}^{S} \left[\sum_i \hat{z}_{ij} \omega_i^{(b)} \hat{\xi}^{i,j}(\rho,\rho') \right] \log a_{\rho,\rho'}^{(r),j}. \qquad (44)
\end{aligned}
$$

Considering the constraints $\sum_{\rho'=1}^{S} a_{\rho,\rho'}^{(r),j} = 1$ and $a_{\rho,\rho'}^{(r),j} \geq 0$, (44) is maximized using the result in Appendix D.1, giving the update in (25).

C.4 Emission Probability Density Functions

The cost function (40) factors also for each GMM indexed by (j, ρ, ℓ). Factoring (40),

$$
\begin{aligned}
\widetilde{\mathcal{L}}(\mathcal{M}_{j,\rho,\ell}^{(r)}) &= \sum_i \hat{z}_{ij} N_i \sum_{\boldsymbol{\beta}} \pi_{\boldsymbol{\beta}}^{(b),i} \sum_{\boldsymbol{\rho}} \hat{\phi}^{i,j}(\boldsymbol{\rho}|\boldsymbol{\beta}) \sum_t \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \\
&= \sum_i \hat{z}_{ij} N_i \sum_{\boldsymbol{\beta}} \left[\pi_{\beta_1}^{(b),i} \prod_{t=2}^{\tau} a_{\beta_{t-1},\beta_t}^{(b),i} \right] \sum_{\boldsymbol{\rho}} \left[\hat{\phi}_1^{i,j}(\rho_1|\beta_1) \prod_{t=2}^{\tau} \hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) \right] \sum_t \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \\
&= \sum_i \hat{z}_{ij} N_i \sum_{\rho_1} \sum_{\beta_1} \underbrace{\pi_{\beta_1}^{(b),i} \hat{\phi}_1^{i,j}(\rho_1|\beta_1)}_{\nu_1^{i,j}(\rho_1,\beta_1)} \Big[\mathcal{L}_{GMM}^{(i,\beta_1),(j,\rho_1)} \dots \\
&\qquad + \sum_{\beta_2 \cdots \beta_\tau} \prod_{t=2}^{\tau} a_{\beta_{t-1},\beta_t}^{(b),i} \sum_{\rho_2 \cdots \rho_\tau} \prod_{t=2}^{\tau} \hat{\phi}^{i,j}(\rho_t|\rho_{t-1},\beta_t) \sum_{t=2}^{\tau} \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \Big] \\
&= \sum_i \hat{z}_{ij} N_i \sum_{\rho_1} \sum_{\beta_1} \nu_1^{i,j}(\rho_1,\beta_1) \mathcal{L}_{GMM}^{(i,\beta_1),(j,\rho_1)} \\
&\quad + \sum_i \hat{z}_{ij} N_i \sum_{\rho_2} \sum_{\beta_2} \underbrace{\sum_{\rho_1} \underbrace{\sum_{\beta_1} \left(\nu_1^{i,j}(\rho_1,\beta_1) a_{\beta_1,\beta_2}^{(b),i} \right) \hat{\phi}_2^{i,j}(\rho_2|\rho_1,\beta_2)}_{\xi_2^{i,j}(\rho_1,\rho_2,\beta_2)}}_{\nu_2^{i,j}(\rho_2,\beta_2)} \Big[\mathcal{L}_{GMM}^{(i,\beta_2),(j,\rho_2)} \dots \\
&\qquad + \sum_{\beta_3 \cdots \beta_\tau} \prod_{t=3}^{\tau} a_{\beta_{t-1},\beta_t}^{(b),i} \sum_{\rho_3 \cdots \rho_\tau} \prod_{t=3}^{\tau} \hat{\phi}_t^{i,j}(\rho_t|\rho_{t-1},\beta_t) \sum_{t=3}^{\tau} \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \Big] \\
&= \dots \\
&= \sum_i \hat{z}_{ij} N_i \sum_{t=1}^{\tau} \sum_{\rho_t} \sum_{\beta_t} \nu_t^{i,j}(\rho_t,\beta_t) \mathcal{L}_{GMM}^{(i,\beta_t),(j,\rho_t)} \\
&\propto \sum_i \hat{z}_{ij} N_i \sum_{\beta} \underbrace{\sum_{t=1}^{\tau} \nu_t^{i,j}(\rho,\beta)}_{\hat{\nu}_t^{i,j}(\rho,\beta)} \mathcal{L}_{GMM}^{(i,\beta),(j,\rho)} \\
&\propto \sum_i \hat{z}_{ij} N_i \sum_{\beta=1}^{S} \hat{\nu}^{i,j}(\rho,\beta) \sum_{m=1}^{M} c_{\beta,m}^{(b),i} \, \hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)} \left[\log c_{\rho,\ell}^{(r),j} + \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}} [\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})] \right] \\
&= \Omega_{j,\rho} \left(\hat{\eta}_{\ell|m}^{(i,\beta),(j,\rho)} \left[\log c_{\rho,\ell}^{(r),j} + \mathrm{E}_{\mathcal{M}_{i,\beta,m}^{(b)}} [\log p(y|\mathcal{M}_{j,\rho,\ell}^{(r)})] \right] \right), \qquad (45)
\end{aligned}
$$

where in (45) we use the weighted-sum operator defined in (28), which is over all base model GMMs $\{\mathcal{M}_{i,\beta,m}^{(b)}\}$. The GMM mixture weights are subject to the constraints $\sum_{\ell=1}^{M} c_{\rho,\ell}^{(r),j} = 1$,

$\forall j, \rho$. Taking the derivative with respect to each parameter and setting it to zero,[27] gives the GMM update Equations (26) and (27).

Appendix D. Useful Optimization Problems

The following two optimization problems are used in the derivation of the E-step and M-step.

D.1

The optimization problem

$$\max_{\alpha_\ell} \quad \sum_{\ell=1}^{L} \beta_\ell \log \alpha_\ell \qquad \text{s.t.} \qquad \sum_{\ell=1}^{L} \alpha_\ell = 1, \quad \alpha_\ell \geq 0, \forall \ell$$

is optimized by $\alpha_\ell^* = \frac{\beta_\ell}{\sum_{\ell'=1}^{L} \beta'_\ell}$.

D.2

The optimization problem

$$\max_{\alpha_\ell} \quad \sum_{\ell=1}^{L} \alpha_\ell \left(\beta_\ell - \log \alpha_\ell\right) \qquad \text{s.t.} \qquad \sum_{\ell=1}^{L} \alpha_\ell = 1 \quad \alpha_\ell \geq 0, \forall \ell$$

is optimized by $\alpha_\ell^* = \frac{\exp \beta_\ell}{\sum_{\ell'=1}^{L} \exp \beta'_\ell}$.

Appendix E. Clustering Synthetic Data where the Clustering Model Does Not Match the "True" Model

In this appendix we present two experiments on clustering synthetic data, where the order of the model used for clustering does not necessarily match the order of the "true" model used to generate the data. In both experiments, the true model consists of $C = 4$ HMMs each with $S = 3$ hidden states—we used the HMMs of experiment (c) in Section 5.3. Data sequences are generated from each of the C HMMs, and then corrupted with Gaussian noise. An HMM with S' states is estimated over each sequence. These HMMs are denoted as *noisy* HMMs, to differentiate them from the original HMMs representing the C classes. The entirety of noisy HMMs are then clustered into C' groups using in turn our VHEM-H3M algorithm and PPK-SC.

In the first experiments we set $S' = S(= 3)$, and vary $C' \in \{3, 4, 5, 6\}$. Hence, the number of clusters does not necessarily match the true number of groups. In the second experiments, we fix $C' = C(= 4)$, and vary $S' \in \{2, 3, 4, 5\}$, that is, the model order of the noisy HMMs does not match the order of the original HMMs. Results are reported in Table 7 below.

27. We also considered the constraints on the covariance matrices $\Sigma_{\rho,\ell}^{(r),j} \succ \mathbf{0}$.

	experiment (d) number of clusters C' varies, $S' = 3$				experiment (e) number of HMM states S' varies, $C' = 4$ fixed			
	2	3	4 (true)	5	2	3 (true)	4	5
VHEM-H3M	0.665	0.782	0.811	0.816	0.801	0.811	0.828	0.832
PPK-SC	0.673	0.729	0.768	0.781	0.766	0.768	0.781	0.798

Table 7: Results on clustering synthetic data with VHEM-H3M and PPK-SC, when the model order does not match the order of the true model used for generating the data. We fist vary the number of clusters C', keeping $S' = 3$ fixed to the true value. Then, we vary the number of HMM states S', keeping $C' = 4$ fixed to the true value. Performance is measured in terms of Rand-index, and is averaged over $K \in \{2, 4, 8, 16, 32\}$.

Performance are more sensitive to selecting a sufficient number of clusters than using the right number of HMM states. In particular, when using fewer clusters than the true number of classes (e.g., $C' < C$), the Rand-index degrades for both VHEM-H3M and PPK-SC, see experiment (d) in Table 7. On the opposite, performance are relatively stable when the number of HMM states does not match the true one, for example, $S' \neq S$, see experiment (e) in Table 7. In particular, when using fewer HMM states (e.g., $S' < S$) the model can still capture some of the dynamics, and the drop in performance is not significant. It is also interesting to note that using a larger number of HMM states (e.g., $S' > S$) leads to slightly better results. The reason is that, when estimating the HMMs on the corrupted data sequences, there are additional states to better account for the effect of the noise.

References

Dimitris Achlioptas, Frank McSherry, and Bernhard Schölkopf. Sampling techniques for kernel methods. In *Advances in Neural Information Processing Systems 14*, volume 1, pages 335–342. MIT Press, 2002.

Jonathan Alon, Stan Sclaroff, George Kollios, and Vladimir Pavlovic. Discovering clusters in motion time-series data. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 368–375. IEEE, 2003.

Arthur Asuncion and David J Newman. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

Claus Bahlmann and Hans Burkhardt. Measuring HMM similarity with the Bayes probability of error and its application to online handwriting recognition. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 406–411. IEEE, 2001.

Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005.

Eloi Batlle, Jaume Masip, and Enric Guaus. Automatic song identification in noisy broadcast audio. In *Proceeding of the International Conference on Signal and Image Processing*. IASTED, 2002.

Jerome R Bellegarda and David Nahamoo. Tied mixture continuous parameter modeling for speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(12):2033–2045, 1990.

Yoshua Bengio, Jean-François Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Le Roux, and Marie Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems*, volume 16, pages 177–184. MIT Press, 2004.

David M Blei and Michael I Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 2006.

William M Campbell. A SVM/HMM system for speaker recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II–209. IEEE, 2003.

Gustavo Carneiro, Antoni B Chan, Pedro J Moreno, and Nuno Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):394–410, 2007.

Antoni B Chan, Emanuele Coviello, and Gert RG Lanckriet. Clustering Dynamic Textures with the Hierarchical EM Algorithm. In *Proceeding of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010a.

Antoni B Chan, Emanuele Coviello, and Gert RG Lanckriet. Derivation of the hierarchical EM Algorithm for Dynamic Textures. Technical report, City University of Hong Kong, 2010b.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Learning sequence kernels. In *IEEE Workshop on Machine Learning for Signal Processing*, pages 2–8. IEEE, 2008.

Emanuele Coviello, Antoni B Chan, and Gert RG Lanckriet. Time series models for semantic music annotation. *IEEE Transactions on Audio, Speech, and Language Processing*, 5(19): 1343–1359, 2011.

Emanuele Coviello, Antoni Chan, and Gert Lanckriet. The variational hierarchical EM algorithm for clustering hidden Markov models. In *Advances in Neural Information Processing Systems 25*, pages 413–421, 2012a.

Emanuele Coviello, Adeel Mumtaz, Antoni B Chan, and Gert RG Lanckriet. Growing a Bag of Systems Tree for fast and accurate classification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1979–1986. IEEE, 2012b.

Imre Csiszár and Gábor Tusnády. Information geometry and alternating minimization procedures. *Statistics and decisions*, 1984.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

Inderjit S. Dhillon. Differential entropic clustering of multivariate Gaussians. In *Advances in Neural Information Processing Systems*, volume 19, page 337. MIT Press, 2007.

Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal on Computer Vision*, 51(2):91–109, 2003.

Douglas Eck, Paul Lamere, Thierry Bertin-Mahieux, and Stephen Green. Automatic generation of social tags for music recommendation. In *Advances in Neural Information Processing Systems*, pages 385–392. MIT Press, 2008.

Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

Jacob Goldberger and Sam Roweis. Hierarchical clustering of a mixture model. In *Advances in Neural Information Processing Systems*, pages 505–512. MIT Press, 2004.

Asela Gunawardana and William Byrne. Convergence theorems for generalized alternating minimization procedures. *Journal of Machine Learning Research*, 6:2049–73, 2005.

Peter Hall, Joel L Horowitz, and Bing-Yi Jing. On blocking rules for the bootstrap with dependent data. *Biometrika*, 82(3):561–574, 1995.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

Christian Hennig. Cluster-wise assessment of cluster stability. *Computational Statistics & Data Analysis*, 52(1):258–271, 2007.

John R Hershey and Peder A Olsen. Variational Bhattacharyya divergence for hidden Markov models. In *IEEE Computer Society International Conference on Acoustics, Speech and Signal Processing*, pages 4557–4560. IEEE, 2008.

John R Hershey, Peder A Olsen, and Steven J Rennie. Variational Kullback-Leibler divergence for hidden Markov models. In *IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 323–328. IEEE, 2007.

Matthew D Hoffman, David M Blei, and Perry R Cook. Easy as CBA: A simple probabilistic model for tagging music. In *Proceedings of the 10th International Symposium on Music Information Retrieval*, pages 369–374, 2009.

Xuedong D Huang and Mervyn A Jack. Semi-continuous hidden Markov models for speech signals. *Computer Speech & Language*, 3(3):239–251, 1989.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2 (1):193–218, 1985.

Tommi S. Jaakkola. Tutorial on Variational Approximation Methods. In *Advanced Mean Field Methods: Theory and Practice*, pages 129–159. MIT Press, 2000.

Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *The Journal of Machine Learning Research*, 5:819–844, 2004.

Tony Jebara, Yingbo Song, and Kapil Thadani. Spectral clustering and embedding with hidden Markov models. In *Machine Learning: ECML 2007*, pages 164–175. 2007.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Biing-Hwang Juang and Lawrence R Rabiner. A probabilistic distance measure for hidden Markov models. *AT&T Technical Journal*, 64(2):391–408, February 1985.

Leonard Kaufman and Peter Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis Based on the L1-Norm and Related Methods*, pages 405–416, 1987.

Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

Eamonn J Keogh and Michael J Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 285–289. ACM, 2000.

Ariel Kleiner, Ameet Talwalkar, Purnamrita Sarkar, and Michael I Jordan. Bootstrapping big data. *Advances in Neural Information Processing Systems, Workshop: Big Learning: Algorithms, Systems, and Tools for Learning at Scale*, 2011.

Anders Krogh, Michael Brown, I Saira Mian, Kimmen Sjölander, and David Haussler. Hidden Markov models in computational biology. Applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1531, 1994.

Pavel Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic. Scalable algorithms for string kernels with inexact matching. In *Advances in Neural Information Processing Systems*, pages 881–888, 2008.

Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 7, pages 566–575, 2002.

Marcus Liwicki and Horst Bunke. Iam-ondb-an on-line english sentence database acquired from handwritten text on a whiteboard. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 956–961. IEEE, 2005.

Rune B Lyngso, Christian NS Pedersen, and Henrik Nielsen. Metrics and similarity measures for hidden Markov models. In *Proceedings International Conference on Intelligent Systems for Molecular Biology*, pages 178–186, 1999.

David JC MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge university press, 2003.

Michael I Mandel and Daniel PW Ellis. Multiple-instance learning for music information retrieval. In *Proceedings of the 9th International Symposium on Music Information Retrieval*, pages 577–582, 2008.

Nag, Kin-Hong Wong, and Frank Fallside. Script recognition using hidden Markov models. In *IEEE Computer Society International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 2071–2074. IEEE, 1986.

Radford M Neal and Geoffrey E Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in Graphical Models*, 89:355–370, 1998.

Evert J Nyström. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54(1):185–204, 1930.

Tim Oates, Laura Firoiu, and Paul R Cohen. Clustering time series with hidden markov models and dynamic time warping. In *Joint Conferences on Artificial Intelligence, Workshop on Neural, Symbolic and Reinforcement Learning Methods for Sequence Learning*, pages 17–21, 1999.

Antonello Panuccio, Manuele Bicego, and Vittorio Murino. A hidden Markov model-based approach to sequential data clustering. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 734–743, 2002.

William D Penny and Stephen J Roberts. Notes on variational learning. Technical report, Oxford University, 2000.

Yuting Qi, John William Paisley, and Lawrence Carin. Music analysis using hidden Markov mixture models. *IEEE Transactions on Signal Processing*, 55(11):5209–5224, 2007.

Lawrence R Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Upper Saddle River (NJ, USA), 1993.

Jeremy Reed and Chin-Hui Lee. A study on music genre classification based on universal acoustic models. In *Proceedings of the 7th International Symposium on Music Information Retrieval*, pages 89–94, 2006.

Jose Rodriguez-Serrano and Florent Perronnin. A model-based sequence similarity with application to handwritten word-spotting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34:2108 – 2120, 2012.

Nicolas Scaringella and Giorgio Zoia. On the modeling of time information for automatic genre recognition systems in audio signals. In *Proc. 6th International Symposium Music Information Retrieval*, pages 666–671, 2005.

Padhraic Smyth. Clustering sequences with hidden Markov models. In *Advances in Neural Information Processing Systems*, pages 648–654. MIT Press, 1997.

Theodoros Theodoridis and Huosheng Hu. Action classification of 3d human models using dynamic ANNs for mobile robot surveillance. In *IEEE Computer Society International Conference on Robotics and Biomimetics*, pages 371–376. IEEE, 2007.

Douglas Turnbull, Luke Barrington, David Torres, and Gert RG Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, February 2008.

Nuno Vasconcelos. Image indexing with mixture hierarchies. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

Nuno Vasconcelos and Andrew Lippman. Learning mixture hierarchies. In *Advances in Neural Information Processing Systems*, 1998.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Ben H Williams, Marc Toussaint, and Amos J Storkey. Extracting motion primitives from natural handwriting data. In *Proceeding of the 16th International Conference on Artificial Neural Networks*, pages 634–643. Springer, 2006.

Jie Yin and Qiang Yang. Integrating hidden Markov models and spectral analysis for sensory time series clustering. In *IEEE Computer Society International Conference on Data Mining*. IEEE, 2005.

Shi Zhong and Joydeep Ghosh. A unified framework for model-based clustering. *The Journal of Machine Learning Research*, 4:1001–1037, 2003.

Journal of Machine Learning Research 15 (2014) 749-808 Submitted 12/11; Revised 6/13; Published 2/14

A Novel M-Estimator for Robust PCA

Teng Zhang ZHANG620@UMN.EDU
Institute for Mathematics and its Applications
University of Minnesota
Minneapolis, MN 55455, USA

Gilad Lerman* LERMAN@UMN.EDU
School of Mathematics
University of Minnesota
Minneapolis, MN 55455, USA

Editor: Martin Wainwright

Abstract

We study the basic problem of robust subspace recovery. That is, we assume a data set that some of its points are sampled around a fixed subspace and the rest of them are spread in the whole ambient space, and we aim to recover the fixed underlying subspace. We first estimate "robust inverse sample covariance" by solving a convex minimization procedure; we then recover the subspace by the bottom eigenvectors of this matrix (their number correspond to the number of eigenvalues close to 0). We guarantee exact subspace recovery under some conditions on the underlying data. Furthermore, we propose a fast iterative algorithm, which linearly converges to the matrix minimizing the convex problem. We also quantify the effect of noise and regularization and discuss many other practical and theoretical issues for improving the subspace recovery in various settings. When replacing the sum of terms in the convex energy function (that we minimize) with the sum of squares of terms, we obtain that the new minimizer is a scaled version of the inverse sample covariance (when exists). We thus interpret our minimizer and its subspace (spanned by its bottom eigenvectors) as robust versions of the empirical inverse covariance and the PCA subspace respectively. We compare our method with many other algorithms for robust PCA on synthetic and real data sets and demonstrate state-of-the-art speed and accuracy.

Keywords: principal components analysis, robust statistics, M-estimator, iteratively re-weighted least squares, convex relaxation

1. Introduction

The most useful paradigm in data analysis and machine learning is arguably the modeling of data by a low-dimensional subspace. The well-known total least squares solves this modeling problem by finding the subspace minimizing the sum of squared errors of data points. This is practically done via principal components analysis (PCA) of the data matrix. Nevertheless, this procedure is highly sensitive to outliers. Many heuristics have been proposed for robust recovery of the underlying subspace. Recent progress in the rigorous study of sparsity and low-rank of data has resulted in provable convex algorithms for this purpose. Here, we propose a different rigorous and convex approach, which is a special M-estimator.

*. Gilad Lerman is the corresponding author.

Robustness of statistical estimators has been carefully studied for several decades (Huber and Ronchetti, 2009; Maronna et al., 2006). A classical example is the robustness of the geometric median (Lopuhaä and Rousseeuw, 1991). For a data set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^D$, the geometric median is the minimizer of the following function of $\mathbf{y} \in \mathbb{R}^D$:

$$\sum_{i=1}^{N} \|\mathbf{y} - \mathbf{x}_i\|, \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm. This is a typical example of an M-estimator, that is, a minimizer of a function of the form $\sum_{i=1}^N \rho(r_i)$, where r_i is a residual of the ith data point, $\mathbf{x}_i$, from the parametrized object we want to estimate. Here, $r_i = \|\mathbf{y}-\mathbf{x}_i\|$, $\rho(x) = |x|$ and we estimate $\mathbf{y} \in \mathbb{R}^D$, which is parametrized by its D coordinates.

There are several obstacles in developing robust and effective estimators for subspaces. For simplicity, we discuss here estimators of linear subspaces and thus assume that the data is centered at the origin.[1] A main obstacle is due to the fact that the set of d-dimensional linear subspaces in $\mathbb{R}^D$, that is, the Grassmannian $\mathrm{G}(D,d)$, is not convex. Therefore, a direct optimization on $\mathrm{G}(D,d)$ (or a union of $\mathrm{G}(D,d)$ over different d's) will not be convex (even not geodesically convex) and may result in several (or many) local minima. Another problem is that extensions of simple robust estimators of vectors to subspaces (e.g., using l_1-type averages) can fail by a single far away outlier. For example, one may extend the d-dimensional geometric median minimizing (1) to the minimizer over $\mathrm{L} \in \mathrm{G}(D,d)$ of the function

$$\sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{P}_{\mathrm{L}}\mathbf{x}_i\| \equiv \sum_{i=1}^{N} \|\mathbf{P}_{\mathrm{L}^\perp}\mathbf{x}_i\|, \tag{2}$$

where $\mathrm{L}^\perp$ is the orthogonal complement of L and $\mathbf{P}_{\mathrm{L}}$ and $\mathbf{P}_{\mathrm{L}^\perp}$ are the orthogonal projections on L and $\mathrm{L}^\perp$ respectively (see, e.g., Ding et al., 2006; Lerman and Zhang, 2010). However, a single outlier with arbitrarily large magnitude will enforce the minimizer of (2) to contain it.

The first obstacle can be resolved by applying a convex relaxation of the minimization of (2) so that subspaces are mapped into a convex set of matrices (the objective function may be adapted respectively). Indeed, the subspace recovery proposed by Xu et al. (2010b) can be interpreted in this way. Their objective function has one component which is similar to (2), though translated to matrices. They avoid the second obstacle by introducing a second component, which penalizes inliers of large magnitude (so that outliers of large magnitude may not be easily identified as inliers). However, the combination of the two components involves a parameter that needs to be carefully estimated.

Here, we suggest a different convex relaxation that does not introduce arbitrary parameters and its implementation is significantly faster. However, it introduces some restrictions on the distributions of inliers and outliers. Some of these restrictions have analogs in other

1. This is a common assumption to reduce the complexity of the subspace recovery problem (Candès et al., 2011; Xu et al., 2010b, 2012; McCoy and Tropp, 2011), where McCoy and Tropp (2011) suggest centering by the geometric median. Nevertheless, our methods easily adapt to affine subspace fitting by simultaneously estimating both the offset and the shifted linear component, but the justification is a bit more complicated then.

works (see, e.g., §2.2), while others are unique to this framework (see §2.3 and the non-technical description of all of our restrictions in §1.2).

1.1 Previous Work

Many algorithms (or pure estimators) have been proposed for robust subspace estimation or equivalently robust low rank approximation of matrices. Maronna (1976), Huber and Ronchetti (2009, §8), Devlin et al. (1981), Davies (1987), Xu and Yuille (1995), Croux and Haesbroeck (2000) and Maronna et al. (2006, §6) estimate a robust covariance matrix. Some of these methods use M-estimators (Maronna et al., 2006, §6) and compute them via iteratively re-weighted least squares (IRLS) algorithms, which linearly converge (Arslan, 2004). The convergence of algorithms based on other estimators or strategies is not as satisfying. The objective functions of the MCD (Minimum Covariance Determinant) and S-estimators converge (Maronna et al., 2006, §6), but no convergence rates are specified. Moreover, there are no guarantees for the actual convergence to the global optimum of these objective functions. There is no good algorithm for the MVE (Minimum Volume Ellipsoid) or Stahel-Donoho estimators (Maronna et al., 2006, §6). Furthermore, convergence analysis is problematic for the online algorithm of Xu and Yuille (1995).

Li and Chen (1985), Ammann (1993), Croux et al. (2007), Kwak (2008) and McCoy and Tropp (2011, §2) find low-dimensional projections by "Projection Pursuit" (PP), now commonly referred to as PP-PCA (the initial proposal is due to Huber, see, e.g., Huber and Ronchetti, 2009, p. 204 of first edition). The PP-PCA procedure is based on the observation that PCA maximizes the projective variance and can be implemented incrementally by computing the residual principal component or vector each time. Consequently, PP-PCA replaces this variance by a more robust function in this incremental implementation. Most PP-based methods are based on non-convex optimization and consequently lack satisfying guarantees. In particular, Croux et al. (2007) do not analyze convergence of their non-convex PP-PCA and Kwak (2008) only establishes convergence to a local maximum. McCoy and Tropp (2011, §2) suggest a convex relaxation for PP-PCA. However, they do not guarantee that the output of their algorithm coincides with the exact maximizer of their energy (though they show that the energies of the two are sufficiently close). Ammann (1993) applies a minimization on the sphere, which is clearly not convex. It iteratively tries to locate vectors spanning the orthogonal complement of the underlying subspace, that is, $D-d$ vectors for a subspace in $\mathrm{G}(D,d)$. We remark that our method also suggests an optimization revealing the orthogonal complement, but it requires a single convex optimization, which is completely different from the method of Ammann (1993).

Torre and Black (2001, 2003), Brubaker (2009) and Xu et al. (2010a) remove possible outliers, followed by estimation of the underlying subspace by PCA. These methods are highly non-convex. Nevertheless, Xu et al. (2010a) provide a probabilistic analysis for their near recovery of the underlying subspace.

The non-convex minimization of (2) as a robust alternative for principal component analysis was suggested earlier by various authors for hyperplane modeling (Osborne and Watson, 1985; Späth and Watson, 1987; Nyquist, 1988; Bargiela and Hartley, 1993), surface modeling (Watson, 2001, 2002), subspace modeling (Ding et al., 2006) and multiple subspaces modeling (Zhang et al., 2009). This minimization also appeared in a pure geometric-analytic

context of general surface modeling without outliers (David and Semmes, 1991). Lerman and Zhang (2010, 2011) have shown that this minimization can be robust to outliers under some conditions on the sampling of the data.

Ke and Kanade (2003) tried to minimize (over all low-rank approximations) the element-wise l_1 norm of the difference of a given matrix and its low-rank approximation. Chandrasekaran et al. (2011) and Candès et al. (2011) have proposed to minimize a linear combination of such an l_1 norm and the nuclear norm of the low-rank approximation in order to find the optimal low-rank estimator. Candès et al. (2011) considered the setting where uniformly sampled elements of the low-rank matrix are corrupted, which does not apply to our outlier model (where only some of the rows are totally corrupted). Chandrasekaran et al. (2011) consider a general setting, though their underlying condition is too restrictive; weaker condition was suggested by Hsu et al. (2011), though it is still not sufficiently general. Nevertheless, Chandrasekaran et al. (2011) and Candès et al. (2011) are groundbreaking to the whole area, since they provide rigorous analysis of exact low-rank recovery with unspecified rank.

Xu et al. (2010b) and McCoy and Tropp (2011) have suggested a strategy analogous to Chandrasekaran et al. (2011) and Candès et al. (2011) to solve the outlier problem. They divide the matrix $\mathbf{X}$ whose rows are the data points as follows: $\mathbf{X} = \mathbf{L} + \mathbf{O}$, where $\mathbf{L}$ is low-rank and $\mathbf{O}$ represents outliers (so that only some of its rows are non-zero). They minimize $\|\mathbf{L}\|_* + \lambda\|\mathbf{O}\|_{(2,1)}$, where $\|\cdot\|_*$ and $\|\cdot\|_{(2,1)}$ denote the nuclear norm and sum of l_2 norms of rows respectively and λ is a parameter that needs to be carefully chosen. We note that the term $\|\mathbf{O}\|_{(2,1)}$ is analogous to (2). Xu et al. (2012) have established an impressive theory showing that under some incoherency conditions, a bound on the fraction of outliers and correct choice of the parameter λ, they can exactly recover the low-rank approximation. Hsu et al. (2011) and Agarwal et al. (2012a) improved error bounds for this estimator as well as for the ones of Chandrasekaran et al. (2011) and Candès et al. (2011).

In practice, the implementations by Chandrasekaran et al. (2011), Candès et al. (2011), Xu et al. (2010b) and McCoy and Tropp (2011) use the iterative procedure described by Lin et al. (2009). The difference between the objective functions of the minimizer and its estimator obtained at the kth iteration is of order $O(k^{-2})$ (Lin et al., 2009, Theorem 2.1). On the other hand, for our algorithm the convergence rate is of order $O(\exp(-ck))$ for some constant c (i.e., it r-linearly converges). This rate is the order of the Frobenius norm of the difference between the minimizer sought by our algorithm (formulated in (4) below) and its estimator obtained at the kth iteration (it is also the order of the difference of the regularized objective functions of these two matrices). Recently, Agarwal et al. (2012b) showed that projected gradient descent algorithms for these estimators obtain linear convergence rates, though with an additional statistical error.

Our numerical algorithm can be categorized as IRLS. Weiszfeld (1937) used a procedure similar to ours to find the geometric median. Lawson (1961) later used it to solve uniform approximation problems by the limits of weighted l_p-norm solutions. This procedure was generalized to various minimization problems, in particular, it is native to M-estimators (Huber and Ronchetti, 2009; Maronna et al., 2006), and its linear convergence was proved for special instances (see, e.g., Cline, 1972; Voss and Eckhardt, 1980; Chan and Mulet, 1999). Recently, IRLS algorithms were also applied to sparse recovery and matrix completion (Daubechies et al., 2010; Fornasier et al., 2011).

1.2 This Work

We suggest another convex relaxation of the minimization of (2). We note that the original minimization is over all subspaces L or equivalently all orthogonal projectors $\mathbf{P} \equiv \mathbf{P}_{\mathrm{L}^\perp}$. We can identify $\mathbf{P}$ with a $D \times D$ matrix satisfying $\mathbf{P}^2 = \mathbf{P}$ and $\mathbf{P}^T = \mathbf{P}$ (where $\cdot^T$ denotes the transpose). Since the latter set is not convex, we relax it to include all symmetric matrices, but avoid singularities by enforcing unit trace. That is, we minimize over the set:

$$\mathbb{H} := \{\mathbf{Q} \in \mathbb{R}^{D\times D} : \mathbf{Q} = \mathbf{Q}^T, \mathrm{tr}(\mathbf{Q}) = 1\} \tag{3}$$

as follows

$$\hat{\mathbf{Q}} = \operatorname*{arg\,min}_{\mathbf{Q}\in\mathbb{H}} F(\mathbf{Q}), \text{ where } F(\mathbf{Q}) := \sum_{i=1}^{N} \|\mathbf{Q}\mathbf{x}_i\|. \tag{4}$$

For the noiseless case (i.e., inliers lie exactly on L^*), we estimate the subspace L^* by

$$\hat{\mathrm{L}} := \ker(\hat{\mathbf{Q}}). \tag{5}$$

If the intrinsic dimension d is known (or can be estimate from the data), we estimate the subspace by the span of the bottom d eigenvectors of $\hat{\mathbf{Q}}$ (or equivalently, the top d eigenvectors of $-\hat{\mathbf{Q}}$). This procedure is robust to sufficiently small levels of noise. We refer to it as the Geometric Median Subspace (GMS) algorithm and summarize it in Algorithm 1. We elaborate on this scheme throughout the paper,

Algorithm 1 The Geometric Median Subspace Algorithm

Input: $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N \subseteq \mathbb{R}^D$: data, d: dimension of L^*, an algorithm for minimizing (4)
Output: $\hat{\mathrm{L}}$: a d-dimensional linear subspace in $\mathbb{R}^D$.
Steps:
- $\{\mathbf{v}_i\}_{i=1}^d =$ the bottom d eigenvectors of $\hat{\mathbf{Q}}$ (see (4))
- $\hat{\mathrm{L}} = \mathrm{Sp}(\{\mathbf{v}_i\}_{i=1}^d)$

We remark that $\hat{\mathbf{Q}}$ is semi-definite positive (we verify this later in Lemma 14). We can thus restrict $\mathbb{H}$ to contain only semi-definite positive matrices and thus make it even closer to a set of orthogonal projectors. Theoretically, it makes sense to require that the trace of the matrices in $\mathbb{H}$ is $D-d$ (since they are relaxed versions of projectors onto the orthogonal complement of a d-dimensional subspace). However, scaling of the trace in (3) results in scaling the minimizer of (4) by a constant, which does not effect the subspace recovery procedure.

We note that (4) is an M-estimator with residuals $r_i = \|\mathbf{Q}\mathbf{x}_i\|$, $1 \le i \le N$, and $\rho(x) = |x|$. Unlike (2), which can also be seen as a formal M-estimator, the estimator $\hat{\mathbf{Q}}$ is unique under a weak condition that we will state later.

We are unaware of similar formulations for the problem of robust PCA. Nevertheless, the Low-Rank Representation (LRR) framework of Liu et al. (2010, 2013) for modeling data by multiple subspaces (and not a single subspace as in here) is formally similar. LRR tries to assign to a data matrix $\mathbf{X}$, which is viewed as a dictionary of N column vectors in $\mathbb{R}^D$, dictionary coefficients $\mathbf{Z}$ by minimizing $\lambda\|\mathbf{Z}\|_* + \|(\mathbf{X}(\mathbf{I}-\mathbf{Z}))^T\|_{(2,1)}$ over all $\mathbf{Z} \in \mathbb{R}^{N\times N}$, where λ is a free parameter. Our formulation can be obtained by their formulation with

$\lambda = 0$, $\mathbf{Q} = (\mathbf{I} - \mathbf{Z})^T$ and the additional constraint $\mathrm{tr}(\mathbf{Z}) = D - 1$ (which is equivalent with the scaling $\mathrm{tr}(\mathbf{Q}) = 1$), where $\{\mathbf{x}_i\}_{i=1}^N$ are the row vectors of $\mathbf{X}$ (and not the column vectors that represent the original data points). In fact, our work provides some intuition for LRR as robust recovery of the low rank row space of the data matrix and its use (via $\mathbf{Z}$) in partitioning the column space into multiple subspaces. We also remark that a trace 1 constraint is quite natural in convex relaxation problems and was applied, for example, in the convex relaxation of sparse PCA (d'Aspremont et al., 2007), though the optimization problem there is completely different.

Our formulation is rather simple and intuitive, but results in the following fundamental contributions to robust recovery of subspaces:

1. We prove that our proposed minimization can achieve exact recovery under some assumptions on the underlying data (which we clarify below) and without introducing an additional parameter.

2. We propose a fast iterative algorithm for achieving this minimization and prove its linear convergence.

3. We demonstrate the state-of-the-art accuracy and speed of our algorithm when compared with other methods on both synthetic and real data sets.

4. We establish the robustness of our method to noise and to a common regularization of IRLS algorithms.

5. We explain how to incorporate knowledge of the intrinsic dimension and also how to estimate it empirically.

6. We show that when replacing the sum of norms in (4) by the sum of squares of norms, then the modified minimizer $\hat{\mathbf{Q}}$ is a scaled version of the empirical inverse covariance. The subspace spanned by the bottom d eigenvectors is clearly the d-dimensional subspace obtained by PCA. The original minimizer of (4) can thus be interpreted as a robust version of the inverse covariance matrix.

7. We show that previous and well-known M-estimators (Maronna, 1976; Huber and Ronchetti, 2009; Maronna et al., 2006) do not solve the subspace recovery problem under a common assumption.

1.3 Exact Recovery and Conditions for Exact Recovery by GMS

In order to study the robustness to outliers of our estimator for the underlying subspace, we formulate the exact subspace recovery problem (see also Xu et al. 2012). This problem assumes a fixed d-dimensional linear subspace L^*, inliers sampled from L^* and outliers sampled from its complement; it asks to recover L^* as well as identify correctly inliers and outliers.

In the case of point estimators, like the geometric median minimizing (1), robustness is commonly measured by the breakdown point of the estimator (Huber and Ronchetti, 2009; Maronna et al., 2006). Roughly speaking, the breakdown point measures the proportion of

arbitrarily large observations (that is, the proportion of "outliers") an estimator can handle before giving an arbitrarily large result.

In the case of estimating subspaces, we cannot directly extend this definition, since the set of subspaces, that is, the Grassmannian (or unions of it), is compact, so we cannot talk about "an arbitrarily large result", that is, a subspace with arbitrarily large distance from all other subspaces. Furthermore, given an arbitrarily large data point, we can always form a subspace containing it; that is, this point is not arbitrarily large with respect to this subspace. Instead, we identify the outliers as the ones in the compliment of L^* and we are interested in the largest fraction of outliers (or smallest fraction of inliers per outliers) allowing exact recovery of L^*. Whenever an estimator can exactly recover a subspace under a given sampling scenario we view it as robust and measure its effectiveness by the largest fraction of outliers it can tolerate. However, when an estimator cannot exactly recover a subspace, one needs to bound from below the distance between the recovered subspace and the underlying subspace of the model. Alternatively, one would need to point out at interesting scenarios where exact recovery cannot even occur in the limit when the number of points approaches infinity. We are unaware of other notions of robustness of subspace estimation (but of robustness of covariance estimation, which does not apply here; see, for example, §6.2.1 of Maronna et al. 2006).

In order to guarantee exact recovery of our estimator we basically require three kinds of restrictions on the underlying data, which we explain here on a non-technical level (technical discussion appears in §2). First of all, the inliers need to permeate through the whole underlying subspace L^*, in particular, they cannot concentrate on a lower dimensional subspace of L^*. Second of all, outliers need to permeate throughout the whole complement of L^*. This assumption is rather restrictive and its violation is a failure mode of the algorithm. We thus show that this failure mode does not occur when the knowledge of d is used appropriately. We also suggest some practical methods to avoid this failure mode when d is unknown (see §5.1). Third of all, the "magnitude' of outliers needs to be restricted. We may initially scale all points to the unit sphere in order to avoid extremely large outliers. However, we still need to avoid outliers concentrating along lines, which may have an equivalent effect of a single arbitrarily large outlier. Figure 1 (which appears later in §2) demonstrates cases where these assumptions are not satisfied.

The failure mode discussed above occurs in particular when the number of outliers is rather small and the dimension d is unknown. While we suggest some practical methods to avoid it (see §5.1), we also note that there are many modern applications with high percentages of outliers, where this failure mode may not occur. In particular, computer vision data often contain high percentages of outliers (Stewart, 1999; Chin et al., 2012). However, such data usually involve multiple geometric models, in particular, multiple underlying linear subspaces. We believe that the robust subspace modeling is still relevant to these kinds of data. First of all, robust single subspace strategies can be well-integrated into common schemes of modeling data by multiple subspaces. For example, the K-flats algorithm is based on repetitive clustering and single subspace modeling per cluster (Tipping and Bishop, 1999; Bradley and Mangasarian, 2000; Tseng, 2000; Ho et al., 2003; Zhang et al., 2009, 2012) and the LBF and SLBF algorithms use local subspace modeling (Zhang et al., 2010, 2012). Second of all, some of the important preprocessing tasks in computer vision require single subspace modeling. For example, in face recognition, a preprocessing step

requires efficient subspace modeling of images of the same face under different illuminating conditions (Basri and Jacobs, 2003; Basri et al., 2011). There are also problems in computer vision with more complicated geometric models and large percentage of corruption, where our strategies can be carefully adapted. One important example is the synchronization problem, which finds an important application in Cryo-EM. The goal of this problem is to recover rotation matrices $R_1, \ldots, R_N \in SO(3)$ from noisy and mostly corrupted measurements of $R_i^{-1}R_j$ for some values of $1 \leq i, j \leq N$. Wang and Singer (2013) adapted ideas of both this work and Lerman et al. (2012) to justify and implement a robust solution for the synchronization problem.

1.4 Recent Subsequent Work

In the case where d is known, Lerman et al. (2012) followed this work and suggested a tight convex relaxation of the minimization of (31) over all projectors $\mathbf{P}_{\mathrm{L}^\perp}$ of rank d. Their optimizer, which they refer to as the REAPER (of the needle-in-haystack problem) minimize the same function $F(\mathbf{Q})$ (see (4)) over the set

$$\mathbb{H}' = \{\mathbf{Q} \in \mathbb{R}^{D\times D} : \mathbf{Q} = \mathbf{Q}^T, \mathrm{tr}(\mathbf{Q}) = 1, \|\mathbf{Q}\| \leq \frac{1}{D-d}\}.$$

They estimate the underlying subspace by the bottom d eigenvectors of the REAPER. The new constraints in $\mathbb{H}'$ result in more elegant conditions for exact recovery and tighter probabilistic theory (due to the tighter relaxation). Since d is known the failure mode of GMS mentioned above is avoided. Their REAPER algorithm for computing the REAPER is based on the IRLS procedure of this paper with additional constraints, which complicate its analysis. The algorithmic and theoretical developments of Lerman et al. (2012) are based on the ones here.

While the REAPER framework applies a tighter relaxation, the GMS framework still has several advantages over the REAPER framework. First of all, in various practical situations the dimension of the data is unknown and thus REAPER is inapplicable. On the other hand, GMS can be used for dimension estimation, as we demonstrate in §6.3. Second of all, the GMS algorithm is faster than REAPER (the REAPER requires additional eigenvalue decomposition of a $D \times D$ matrix at each iteration of the IRLS algorithm). Furthermore, we present here a complete theory for the linear convergence of the GMS algorithm, where the convergence theory for the REAPER algorithm is currently incomplete. Third of all, when the failure mode mentioned above is avoided, the empirical performances of REAPER and GMS are usually comparable (while GMS is faster). At last, GMS and REAPER have different objectives with different consequences. REAPER aims to find a projector onto the underlying subspace. On the other hand, GMS aims to find a "generalized inverse covariance" (see §3.3) and is formally similar to other M-estimators (see §3.1 and §3.2). Therefore, the eigenvalues and eigenvectors of the GMS estimator (i.e., the "generalized inverse covariance") can be interpreted as robust eigenvalues and eigenvectors of the empirical covariance (see §6.3 and §6.5).

1.5 Structure of This Paper

In §2 we establish exact and near subspace recovery via the GMS algorithm. We also carefully explain the common obstacles for robust subspace recovery and the way they are

handled by previous rigorous solutions (Candès et al., 2011; Chandrasekaran et al., 2011; Xu et al., 2012) as well as our solution. Section 3 aims to interpret our M-estimator in two different ways. First of all, it shows a formal similarity to a well-known class of M-estimators (Maronna, 1976; Huber and Ronchetti, 2009; Maronna et al., 2006), though clarifies the difference. Those estimators aims to robustly estimate the sample covariance. However, we show there that unlike our M-estimator, they cannot solve the subspace recovery problem (under a common assumption). Second of all, it shows that non-robust adaptation of our M-estimator provides both direct estimation of the inverse covariance matrix as well as convex minimization equivalent to the non-convex total least squares (this part requires full rank data and thus a possible initial dimensionality reduction but without any loss of information). We thus interpret (4) as a robust estimation of the inverse covariance. In §4 we propose an IRLS algorithm for minimizing (4) and establish its linear convergence. Section 5 discusses practical versions of the GMS procedure that allow more general distributions than the ones guaranteed by the theory. One of these versions, the Extended GMS (EGMS) even provides robust alternative to principal components. In §6 we demonstrate the state-of-the-art accuracy and speed of our algorithm when compared with other methods on both synthetic and real data sets and also numerically clarify some earlier claims. Section 7 provides all details of the proofs and §8 concludes with brief discussion.

2. Exact and Near Subspace Recovery by GMS

We establish exact and near subspace recovery by the GMS algorithm. In §2.1 we formulate the problems of exact and near subspace recovery. In §2.2 we describe common obstacles for solving these problems and how they were handled in previous works; in §2.3 we formulate some conditions that the data may satisfy; whereas in §2.4 we claim that these conditions are sufficient to avoid the former obstacles, that is, they guarantee exact recovery (see Theorem 1); We also propose weaker conditions for exact recovery and demonstrate their near-tightness in §2.4.1. Section 2.5 describes a simple general condition for uniqueness of GMS (beyond the setting of exact recovery). Section 2.6 establishes (with some specified limitations) unique exact recovery with high probability under basic probabilistic models (see Theorems 4 and 5); it also covers cases with asymmetric outliers. At last, §2.7 and §2.8 establish results for near recovery under noise and under regularization respectively.

2.1 Problem Formulation

Let us repeat the formulation of the exact subspace recovery problem, which we motivated in §1.2 as a robust measure for the performance of our estimator. We assume a linear subspace $\mathrm{L}^* \in \mathrm{G}(D,d)$ and a data set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, which contains inliers sampled from L^* and outliers sampled from $\mathbb{R}^D \setminus \mathrm{L}^*$. Given the data set $\mathcal{X}$ and no other information, the objective of the exact subspace recovery problem is to exactly recover the underlying subspace L^*.

In order to make the problem well-defined, one needs to assume some conditions on the sampled data set, which may vary with the proposed solution. We emphasize that this is a formal mathematical problem, which excludes some ambiguous scenarios and allows us to determine admissible distributions of inliers and outliers.

In the noisy case (where inliers do not lie on L^*, but perturbed by noise), we ask about near subspace recovery, that is, recovery up to an error depending on the underlying noise level. We argue below that in this case additional information on the model is needed. Here we assume the knowledge of d, though under some assumptions we can estimate d from the data (as we demonstrate later). We remark that exact asymptotic recovery under some conditions on the noise distribution is way more complicated and is discussed in another work (Coudron and Lerman, 2012).

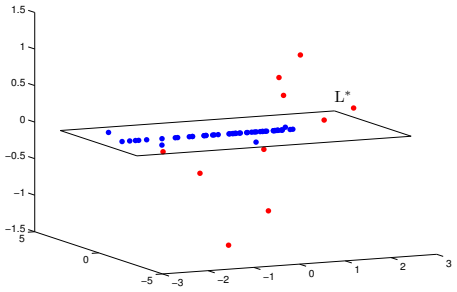
2.2 Common Difficulties with Subspace Recovery

We introduce here three typical enemies of subspace recovery and exemplify them in Figure 1. We also explain how they are handled by the previous convex solutions for exact recovery of subspaces as well as low-rank matrices (Chandrasekaran et al., 2011; Candès et al., 2011; Xu et al., 2012).

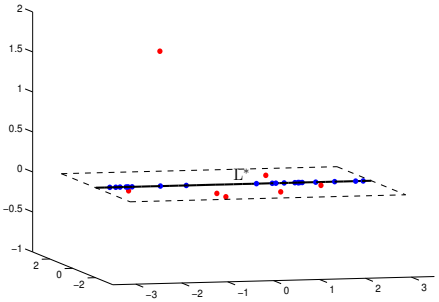
A type 1 enemy occurs when the inliers are mainly sampled from a subspace $\mathrm{L}' \subset \mathrm{L}^*$. In this case, it seems impossible to recover L^*. We would expect a good algorithm to recover L' (instead of L^*) or a subspace containing it with slightly higher dimension (see for example Figure 1(a)). Chandrasekaran et al. (2011), Candès et al. (2011) and Xu et al. (2012) have addressed this issue by requiring incoherence conditions for the inliers. For example, if m and $N-m$ points are sampled from L' and $L^* \setminus L'$ respectively, then the incoherency condition of Xu et al. (2012) requires that $\mu \geq N/(\dim(\mathrm{L}^*) \cdot (N-m))$, where μ is their incoherency parameter. That is, their theory holds only when the fraction of points sampled from $L^* \setminus L'$ is sufficiently large.

A type 2 enemy occurs when the outliers are mainly sampled from a subspace $\tilde{\mathrm{L}}$ such that $\dim(\tilde{\mathrm{L}} \oplus \mathrm{L}^*) < D$. In this case $\mathrm{L}^* \oplus \tilde{\mathrm{L}}$ can be mistakenly identified as the low-rank subspace (see for example Figure 1(b)). This is a main issue when the intrinsic dimension is unknown; if on the other hand the intrinsic dimension is known, then one can often overcome this enemy. Candès et al. (2011) handle it by assuming that the distribution of corrupted elements is uniform. Chandrasekaran et al. (2011) address it by restricting their parameter μ (see their main condition, which is used in Theorem 2 of their work, and their definition of μ in (1.2) of their work) and consequently limit the values of the mixture parameter (denoted here by λ). On the other hand, Xu et al. (2012) use the true percentage of outliers to infer the right choice of the mixture parameter λ. That is, they practically invoke model selection (for estimating this percentage) in order to reject $\tilde{\mathrm{L}} \oplus \mathrm{L}^*$ and choose the true model, which is L^*.

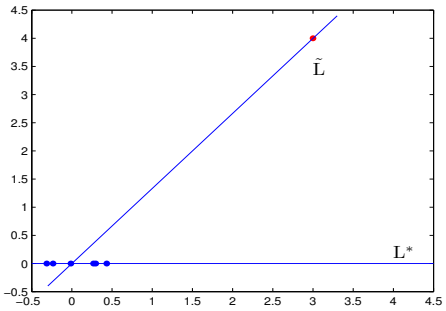
A type 3 enemy occurs due to large magnitudes of outliers. For example, a single outlier with arbitrarily large magnitude will be contained in the minimizer of (2), which will thus be different than the underlying subspace (see for example Figure 1(c)). Also, many outliers with not-so-small magnitudes that lie around a fixed line may have the effect of a single large outlier (see for example Figure 1(d)). This enemy is avoided by Chandrasekaran et al. (2011), Candès et al. (2011) and Xu et al. (2012) by the additional mixture component of nuclear norm, which penalizes the magnitude (or combined magnitude) of the supposed inliers (so that outliers of large magnitude may not be easily identified as inliers). It is interesting to note that if the rank is used instead of the nuclear norm (as sometimes advocated), then it will not resolve this issue.



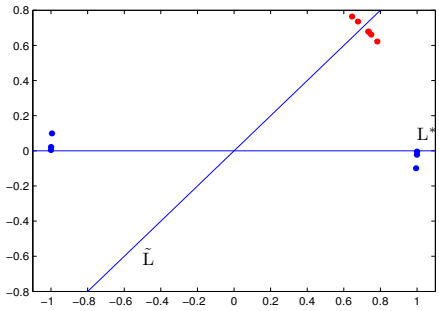
(a) Example of a type 1 enemy: L^* is a plane represented by a rectangle, "inliers" (in L^*) are colored blue and "outliers" (in $\mathbb{R}^3 \setminus L^*$) red. Most inliers lie on a line inside L^*. It seems unlikely to distinguish between inliers, which are not on "the main line", and the outliers. It is thus likely to recover the main line instead of L^*.



(b) Example of a type 2 enemy: L^* is a line represented by a black line segment, "inliers" (in L^*) are colored blue and "outliers" (in $\mathbb{R}^3 \setminus L^*$) red. All outliers but one lie within a plane containing L^*, which is represented by a dashed rectangle. There seems to be stronger distinction between the points on this plane and the isolated outlier than the original inliers and outliers. Therefore, an exact recovery algorithm may output this plane instead of L^*.



(c) Example 1 of a type 3 enemy: The inliers (in blue) lie on the line L^* and there is a single outlier (in red) with relatively large magnitude. An exact recovery algorithm can output the line $\tilde{L}$ (determined by the outlier) instead of L^*. If the data is normalized to the unit circle, then any reasonable robust subspace recovery algorithm can still recover L^*.



(d) Example 2 of a type 3 enemy: Points are normalized to lie on the unit circle, inliers (in blue) lie around the line L^* and outliers (in red) concentrate around another line, $\tilde{L}$. A subspace recovery algorithm can output $\tilde{L}$ instead of L^*.

Figure 1: Enemies of the mathematical formulation of exact subspace recovery.

Another issue for our mathematical problem of exact subspace recovery is whether the subspace obtained by a proposed algorithm is unique. Many of the convex algorithms depend on convex l_1-type methods that may not be strictly convex. But it may still happen that in the setting of pure inliers and outliers and under some conditions avoiding the three types of enemies, the recovered subspace is unique (even though it may be obtained by several non-unique minimizers). This is indeed the case in Chandrasekaran et al. (2011), Candès et al. (2011), Xu et al. (2012) and our own work. Nevertheless, uniqueness of our minimizer (and not the recovered subspace) is important for analyzing the numerical algorithm approximating it and for perturbation analysis (e.g., when considering near recovery with noisy data). It is also helpful for practically verifying the conditions we will propose for exact recovery. Uniqueness of the minimizer (and not just the subspace) is also important in Chandrasekaran et al. (2011) and Candès et al. (2011) and they thus established conditions for it.

At last, we comment that subspace recovery with unknown intrinsic dimension may require a model selection procedure (possibly implicitly). That is, even though one can provide a theory for exact subspace recovery (under some conditions), which might be stable to perturbations, in practice, some form of model selection will be necessary in noisy cases. For example, the impressive theories by Chandrasekaran et al. (2011) and Xu et al. (2012) require the estimation of the mixture parameter λ. Xu et al. (2012) propose such an estimate for λ, which is based on knowledge of the data set (e.g., the distribution of corruptions and the fraction of outliers). However, we noticed that in practice this proposal did not work well (even for simple synthetic examples), partly due to the fact that the deduced conditions are only sufficient, not necessary and there is much room left for improvement. The theory by Candès et al. (2011) specified a choice for λ that is independent of the model parameters, but it applies only for the special case of uniform corruption without noise; moreover, they noticed that other values of λ could achieve better results.

2.3 Conditions for Handling the Three Enemies

We introduce additional assumptions on the data to address the three types of enemies. We denote the sets of exact inliers and outliers by $\mathcal{X}_1$ and $\mathcal{X}_0$ respectively, that is, $\mathcal{X}_1 = \mathcal{X} \cap \mathrm{L}^*$ and $\mathcal{X}_0 = \mathcal{X} \setminus \mathrm{L}^*$. The following two conditions simultaneously address both type 1 and type 3 enemies:

$$\min_{\mathbf{Q}\in\mathbb{H},\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}=\mathbf{0}} \sum_{\mathbf{x}\in\mathcal{X}_1} \|\mathbf{Q}\mathbf{x}\| > \sqrt{2} \min_{\mathbf{v}\in\mathrm{L}^{*\perp},\|\mathbf{v}\|=1} \sum_{\mathbf{x}\in\mathcal{X}_0} |\mathbf{v}^T\mathbf{x}|, \tag{6}$$

$$\min_{\mathbf{Q}\in\mathbb{H},\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}=\mathbf{0}} \sum_{\mathbf{x}\in\mathcal{X}_1} \|\mathbf{Q}\mathbf{x}\| > \sqrt{2} \max_{\mathbf{v}\in\mathrm{L}^{*},\|\mathbf{v}\|=1} \sum_{\mathbf{x}\in\mathcal{X}_0} |\mathbf{v}^T\mathbf{x}|. \tag{7}$$

A lower bound on the common LHS of both (6) and (7) is designed to avoid type 1 enemies. This common LHS is a weak version of the permeance statistics, which was defined in (3.1) of Lerman et al. (2012) as follows:

$$\mathcal{P}(\mathrm{L}^*) := \min_{\substack{\mathbf{u}\in\mathrm{L}^* \\ \|\mathbf{u}\|=1}} \sum_{\mathbf{x}\in\mathcal{X}_1} |\mathbf{u}^T\mathbf{x}|.$$

Similarly to the permeance statistics, it is zero if and only if all inliers are contained in a proper subspace of L^*. Indeed, if all inliers lie in a subspace $\mathrm{L}' \subset \mathrm{L}^*$, then this common LHS is zero with the minimizer $\mathbf{Q} = \mathbf{P}_{\mathrm{L}'^{\perp} \cap \mathrm{L}^*} / \operatorname{tr}(\mathbf{P}_{\mathrm{L}'^{\perp} \cap \mathrm{L}^*})$. Similarly, if it is zero, then $\mathbf{Q}\mathbf{x} = \mathbf{0}$ for any $\mathbf{x} \in \mathcal{X}_1$ and for some $\mathbf{Q}$ with kernel containing $\mathrm{L}^{*\perp}$. This is only possible when $\mathcal{X}_1$ is contained in a proper subspace of L^*. Similarly to the permeance statistics, if the inliers nicely permeate through L^*, then this common LHS clearly obtain large values.

The upper bounds on the RHS's of (6) and (7) address two complementing type 3 enemies. If $\mathcal{X}_0$ contains few data points of large magnitude, which are orthogonal to L^*, then the RHS of (6) may be too large and (6) may not hold. If on the other hand $\mathcal{X}_0$ contains few data points with large magnitude and a small angle with L^*, then the RHS of (7) will be large so that (7) may not hold. Conditions (6) and (7) thus complete each other.

The RHS of condition (7) is similar to the linear structure statistics (for L^*), which was defined in (3.3) of Lerman et al. (2012). The linear structure statistics uses an l_2 average of dot products instead of the l_1 average used here and was applied in this context to $\mathbb{R}^D$ (instead of L^*) in Lerman et al. (2012). Similarly to the linear structure statistics, the RHS of (7) is large when outliers either have large magnitude or they lie close to a line (so that their combined contribution is similar to an outlier with a very large magnitude as exemplified in Figure 1(d)). The RHS of condition (7) is a very weak analog of the linear structure statics of $\mathrm{L}^{*\perp}$ since it uses a minimum instead of a maximum. There are some significant outliers within $\mathrm{L}^{*\perp}$ that will not be avoided by requiring (7). For example, if the codimension of L^* is larger than 1 and there is a single outlier with an arbitrary large magnitude orthogonal to L^*, then the RHS of (7) is zero.

The next condition avoids type 2 enemies and also significant outliers within $\mathrm{L}^{*\perp}$ (i.e., type 3 enemies) that were not avoided by condition (7). This condition requires that any minimizer of the following oracle problem

$$\hat{\mathbf{Q}}_0 := \underset{\mathbf{Q} \in \mathbb{H}, \mathbf{Q}\mathbf{P}_{\mathrm{L}^*} = \mathbf{0}}{\arg\min} F(\mathbf{Q}) \tag{8}$$

satisfies

$$\operatorname{rank}(\hat{\mathbf{Q}}_0) = D - d. \tag{9}$$

We note that the requirement $\mathbf{Q}\mathbf{P}_{\mathrm{L}^*} = \mathbf{0}$ is equivalent to the condition $\ker(\mathbf{Q}) \supseteq \mathrm{L}^*$ and therefore the rank of the minimizer is at most $D - d$. Enforcing the rank of the minimizer to be exactly $D - d$ restricts the distribution of the projection of $\mathcal{X}$ onto $\mathrm{L}^{*\perp}$. In particular, it avoids its concentration on lower dimensional subspaces and is thus suitable to avoid type 2 enemies. Indeed, if all outliers are sampled from $\tilde{\mathrm{L}} \subset \mathrm{L}^{*\perp}$, then any $\mathbf{Q} \in \mathbb{H}$ with $\ker(\mathbf{Q}) \supset \tilde{\mathrm{L}} + \mathrm{L}^*$ satisfies $F(\mathbf{Q}) = 0$ and therefore it is a minimizer of the oracle problem (4), but it contradicts (9).

We note that this condition also avoids some type 3 enemies, which were not handled by conditions (6) and (7). For example, any $D - d - 1$ outliers with large magnitude orthogonal to L^* will not be excluded by requiring (6) or (7), but will be avoided by (9).

This condition is restrictive though, especially in very high ambient dimensions. Indeed, it does not hold when the number of outliers is smaller than $D - d$ (since then the outliers are sampled from some $\tilde{\mathrm{L}}$ with $\dim(\tilde{\mathrm{L}} \oplus \mathrm{L}^*) < D$). We thus explain in §5.2 and §5.2.1 how to avoid this condition when knowing the dimension. We also suggest in §5.1 some practical

solutions to overcome the corresponding restrictive lower bound on the number of outliers when the dimension is unknown.

Example 1 *We demonstrate the violation of the conditions above for the examples depicted in Figure 1. The actual calculations rely on ideas explained in §2.4.1.*

For the example in Figure 1(a), which represents a type 1 enemy, both conditions (6) *and* (7) *are violated. Indeed, the common LHS of* (6) *and* (7) *is* 5.69*, whereas the RHS of* (6) *is* 8.57 *and the RHS of* (7) *is larger than* 10.02 *(this lower bound is obtained by substituting* $\mathbf{v} = [0, 1, 0]$ *in the RHS of* (7)*; note that* $\mathbf{v}$ *is a unit vector in* L^**).*

For the example in Figure 1(b), which represents a type 2 enemy, condition (9) *is violated. Indeed, we obtained numerically a solution* $\hat{\mathbf{Q}}_0$ *with* $\mathrm{rank}(\hat{\mathbf{Q}}_0) = 1 \neq D - d = 2$ *(one can actually prove in this case that* $\hat{\mathbf{Q}}_0$ *is the projector onto the orthogonal complement of the plane represented by the dashed rectangle).*

For the example in Figure 1(c), which represents a type 3 enemy, both conditions (6) *and* (7) *are violated. Indeed, the common LHS of* (6) *and* (7) *is* 1.56 *and the RHS's of* (6) *and* (7) *are* 5.66 *and* 4.24 *respectively. However, if we normalize all points to lie on the unit circle, then this enemy can be overcome. Indeed, for the normalized data, the common LHS of* (6) *and* (7) *is* 6 *and the RHS's of* (6) *and* (7) *are* 1.13 *and* 0.85 *respectively.*

For the example in Figure 1(d), which also represents a type 3 enemy, both conditions (6) *and* (7) *are violated. Indeed, the LHS of* (6) *and* (7) *are* 5.99 *and the RHS's of* (6) *and* (7) *are* 6.91 *and* 7.02 *respectively.*

2.4 Exact Recovery Under Combinatorial Conditions

We show that the minimizer of (4) solves the exact recovery problem under the above combinatorial conditions.

Theorem 1 *Assume that* $d, D \in \mathbb{N}$*,* $d < D$*,* $\mathcal{X}$ *is a data set in* $\mathbb{R}^D$ *and* $\mathrm{L}^* \in \mathrm{G}(D, d)$*. If conditions* (6)*,* (7) *and* (9) *hold (w.r.t.* $\mathcal{X}$ *and* L^**), then any minimizer of* (4)*,* $\hat{\mathbf{Q}}$*, recovers the subspace* L^* *in the following way:* $\ker(\hat{\mathbf{Q}}) = \mathrm{L}^*$*. If only* (6) *and* (7) *hold, then* $\ker(\hat{\mathbf{Q}}) \supseteq \mathrm{L}^*$*.*

2.4.1 Weaker Alternatives of Conditions (6) and (7)

It is sufficient to guarantee exact recovery by requiring (9) and that for an arbitrarily chosen solution of (8), $\hat{\mathbf{Q}}_0$, the following two conditions are satisfied:

$$\min_{\mathbf{Q} \in \mathbb{H}, \mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}} = \mathbf{0}} \sum_{\mathbf{x} \in \mathcal{X}_1} \|\mathbf{Q}\mathbf{x}\| > \sqrt{2} \left\| \sum_{\mathbf{x} \in \mathcal{X}_0} \hat{\mathbf{Q}}_0 \mathbf{x}\mathbf{x}^T \mathbf{P}_{\mathrm{L}^{*\perp}} / \|\hat{\mathbf{Q}}_0 \mathbf{x}\| \right\| \tag{10}$$

and

$$\min_{\mathbf{Q} \in \mathbb{H}, \mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}} = \mathbf{0}} \sum_{\mathbf{x} \in \mathcal{X}_1} \|\mathbf{Q}\mathbf{x}\| > \sqrt{2} \left\| \sum_{\mathbf{x} \in \mathcal{X}_0} \hat{\mathbf{Q}}_0 \mathbf{x}\mathbf{x}^T \mathbf{P}_{\mathrm{L}^{*}} / \|\hat{\mathbf{Q}}_0 \mathbf{x}\| \right\|. \tag{11}$$

We note that condition (9) guarantees that $\hat{\mathbf{Q}}_0 \mathbf{x} \neq \mathbf{0}$ for all $\mathbf{x} \in \mathcal{X}_0$ and thus the RHS's of (10) and (11) are well-defined. We prove this statement in (7.3).

We note that conditions (10) and (11) can be verified when $\mathcal{X}_0$, $\mathcal{X}_1$ and L^* are known (unlike (6) and (7)), where $\hat{\mathbf{Q}}_0$ can be found by Algorithm 2. Furthermore, (10) and (11) are weaker than (6) and (7), though they are more technically involved and harder to motivate.

In order to demonstrate the near-tightness of (10) and (11), we formulate the following necessary conditions for the recovery of L^* as $\ker(\hat{\mathbf{Q}})$ (see the idea of their justification at the end of §7.3): For an arbitrarily chosen solution of (8), $\hat{\mathbf{Q}}_0$:

$$\min_{\mathbf{Q}\in\mathbb{H},\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}=\mathbf{0}} \sum_{\mathbf{x}\in\mathcal{X}_1} \|\mathbf{Q}\mathbf{x}\| \geq \| \sum_{\mathbf{x}\in\mathcal{X}_1} \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\| \tag{12}$$

and

$$\sum_{\mathbf{x}\in\mathcal{X}_1} \|\mathbf{Q}(\tilde{\mathbf{P}}_{\mathrm{L}^*}\mathbf{x})\| \geq \sum_{\mathbf{x}\in\mathcal{X}_0} \left\langle \mathbf{Q}, \tilde{\mathbf{P}}^T_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T\tilde{\mathbf{P}}_{\mathrm{L}^*}/\|\hat{\mathbf{Q}}_0\mathbf{x}\| \right\rangle_F \text{ for any } \mathbf{Q}\in\mathbb{R}^{(D-d)\times d}, \tag{13}$$

where for matrices $\mathbf{A}, \mathbf{B}\in\mathbb{R}^{k\times l}$: $\langle\mathbf{A},\mathbf{B}\rangle_F = \mathrm{tr}(A\,B^T)$ is the Frobenius dot product. Indeed, conditions (12) and (13) are close to conditions (10) and (11). In particular, (12) and (10) are only different by the constant factor $\sqrt{2}$, that is, (10) is practically tight.

2.5 Uniqueness of the Minimizer

We recall that Theorem 1 implies that if (6), (7) and (9) hold, then $\ker(\hat{\mathbf{Q}})$ is unique. Here we guarantee the uniqueness of $\hat{\mathbf{Q}}$ (which is required in §2.4.1, §2.7, §2.8 and §4.2) independently of the exact subspace recovery problem.

Theorem 2 *If the following condition holds:*

$$\{\mathcal{X}\cap \mathrm{L}_1\}\cup\{\mathcal{X}\cap \mathrm{L}_2\} \neq \mathcal{X} \text{ for all } (D-1)\text{-dimensional subspaces } \mathrm{L}_1, \mathrm{L}_2 \subset \mathbb{R}^D, \tag{14}$$

then $F(\mathbf{Q})$ is a strictly convex function on $\mathbb{H}$.

2.6 Exact Recovery under Probabilistic Models

We show that our conditions for exact recovery (or the main two of them) and our condition for uniqueness of the minimizer $\hat{\mathbf{Q}}$ hold with high probability under basic probabilistic models. Such a probabilistic theory is cleaner when the outliers are sampled from a spherically symmetric distribution as we carefully demonstrate in §2.6.1 (with two different models). The problem is that when the outliers are spherically symmetric then various non-robust algorithms (such as PCA) can asymptotically approach exact recovery and nearly recover the underlying subspace with sufficiently large sample. We thus also show in §2.6.2 how the theory in §2.6.1 can be slightly modified to establish exact recovery of the GMS algorithm in an asymmetric case, where PCA cannot even nearly recover the underlying subspace.

2.6.1 Cases with Spherically Symmetric Distributions of Outliers

First we assume a more general probabilistic model. We say that μ on $\mathbb{R}^D$ is an Outliers-Inliers Mixture (OIM) measure (w.r.t. the fixed subspace $\mathrm{L}^*\in\mathrm{G}(D,d)$) if $\mu = \alpha_0\mu_0 + \alpha_1\mu_1$, where $\alpha_0,\ \alpha_1 > 0,\ \alpha_0+\alpha_1 = 1$, μ_1 is a sub-Gaussian probability measure and μ_0 is a sub-Gaussian probability measure on $\mathbb{R}^D$ (representing outliers) that can be decomposed to

a product of two independent measures $\mu_0 = \mu_{0,\mathrm{L}^*} \times \mu_{0,\mathrm{L}^{*\perp}}$ such that the supports of μ_{0,L^*} and $\mu_{0,\mathrm{L}^{*\perp}}$ are L^* and $\mathrm{L}^{*\perp}$ respectively, and $\mu_{0,\mathrm{L}^{*\perp}}$ is spherically symmetric with respect to rotations within $\mathrm{L}^{*\perp}$.

To provide cleaner probabilistic estimates, we also invoke the needle-haystack model of Lerman et al. (2012). It assumes that both μ_0 and μ_1 are the Gaussian distributions: $\mu_0 = N(\mathbf{0}, \sigma_0^2 \mathbf{I}/D)$ and $\mu_1 = N(\mathbf{0}, \sigma_1^2 \mathbf{P}_{\mathrm{L}^*} \mathbf{P}_{\mathrm{L}^*}^T/d)$ (the factors $1/D$ and $1/d$ normalize the magnitude of outliers and inliers respectively so that their norms are comparable). While Lerman et al. (2012) assume a fixed number of outliers and inliers independently sampled from μ_0 and μ_1 respectively, here we independently sample from the mixture measure $\mu = \alpha_0\mu_0 + \alpha_1\mu_1$; we refer to μ as a needle-haystack mixture measure.

In order to prove exact recovery under any of these models, one needs to restrict the fraction of inliers per outliers (or equivalently, the ratio α_1/α_0). We refer to this ratio as SNR (signal to noise ratio) since we may view the inliers as the pure signal and the outliers as some sort of "noise". For the needle-haystack model we require the following SNR, which is similar to the one of Lerman et al. (2012):

$$\frac{\alpha_1}{\alpha_0} > 4\,\frac{\sigma_0}{\sigma_1}\,\frac{d}{\sqrt{(D-d)D}}. \tag{15}$$

We later explain how to get rid of the term σ_1/σ_0. For the OIM model we assume the following more general condition:

$$\alpha_1 \min_{\mathbf{Q}\in\mathbb{H},\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}=\mathbf{0}} \int \|\mathbf{Q}\mathbf{x}\|\,\mathrm{d}\mu_1(\mathbf{x}) > 2\sqrt{2}\frac{\alpha_0}{D-d}\int \|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|\,\mathrm{d}\mu_0(\mathbf{x}). \tag{16}$$

Under the needle-haystack model, this condition is a weaker version of (15). That is,

Lemma 3 *If μ is a needle-haystack mixture measure, then* (15) *implies* (16).

For i.i.d. samples from an OIM measure satisfying (16), we can establish our modified conditions of unique exact recovery (i.e., (10), (11) and (9)) with overwhelming probability in the following way (we also guarantee the uniqueness of the minimizer $\hat{\mathbf{Q}}$).

Theorem 4 *If $\mathcal{X}$ is an i.i.d. sample from an OIM measure μ satisfying* (16)*, then conditions* (10)*,* (11)*, and* (9) *hold with probability $1 - C\exp(-N/C)$, where C is a constant depending on μ and its parameters. Moreover,* (14) *holds with probability* 1 *if there are at least $2D-1$ outliers (i.e., the number of points in $\mathcal{X}\setminus \mathrm{L}^*$ is at least $2D-1$).*

Under the needle-haystack model, the SNR established by Theorem 4 is comparable to the best SNR among other convex exact recovery algorithms (this is later clarified in Table 1). However, the probabilistic estimate under which this SNR holds is rather loose and thus its underlying constant C is not specified. Indeed, the proof of Theorem 4 uses ϵ-nets and union-bounds arguments, which are often not useful for deriving tight probabilistic estimates (see, e.g., Mendelson 2003, page 18). One may thus view Theorem 4 as a near-asymptotic statement.

The statement of Theorem 4 does not contradict our previous observation that the number of outliers should be larger than at least $D-d$. Indeed, the constant C is sufficiently

large so that the corresponding probability is negative when the number of outliers is smaller than $D - d$.

In the next theorem we assume only a needle-haystack model and thus we can provide a stronger probabilistic estimate based on the concentration of measure phenomenon (our proof follows directly Lerman et al., 2012). However, the SNR is worse than the one in Theorem 4 by a factor of order $\sqrt{D-d}$. This is because we are unable to estimate $\hat{\mathbf{Q}}_0$ of (8) by concentration of measure. Similarly, in this theorem we do not estimate the probability of (9) (which also involves $\hat{\mathbf{Q}}_0$). Nevertheless, we observed in experiments that (9) holds with high probability for $N_0 = 2(D-d)$ and the probability seems to go to 1 as $N_0 = 2(D-d)$ and $D - d \to \infty$. Moreover, one of the algorithms proposed below (EGMS) does not require condition (9).

Theorem 5 *If $\mathcal{X}$ is an i.i.d. sample of size N from a needle-haystack mixture measure μ and if*

$$\frac{\alpha_1}{\alpha_0} > \frac{\sigma_0}{\sigma_1} \frac{\sqrt{2/\pi} - 1/4 - 1/10}{\sqrt{2/\pi} + 1/4 + 1/10} \sqrt{\frac{d^2}{D}} \tag{17}$$

and

$$N > 64 \max(2d/\alpha_1, 2d/\alpha_0, 2(D-d)/\alpha_0), \tag{18}$$

then (6) *and* (7) *hold with probability* $1 - e^{-\alpha_1^2 N/2} - 2e^{-\alpha_0^2 N/2} - e^{-\alpha_1 N/800} - e^{-\alpha_0 N/800}$.

In Table 1 we present the theoretical asymptotic SNRs for exact recovery of some recent algorithms. We assume the needle-haystack model with fixed d, D, α_0, α_1, σ_0 and σ_1 and $N \to \infty$. Let us clarify these results. We first remark that the pure SNR of the High-dimensional Robust PCA (HR-PCA) algorithm of Xu et al. (2010a) approaches infinity (see Remark 3 of Xu et al. 2010a). However, as we explained earlier the violation of exact recovery does not necessarily imply non-robustness of the estimator as it may nearly recover the subspace. Indeed, Xu et al. (2010a) show that if (for simplicity) $\sigma_0 = \sigma_1$ and the SNR is greater than 1, then the subspace estimated by HR-PCA is a good approximation in the following sense: there exists a constant $c > 0$ such that for the inliers set $\mathcal{X}_0$ and the estimated subspace L: $\sum_{\mathbf{x} \in \mathcal{X}_0} \|\mathbf{P}_{\mathrm{L}}\mathbf{x}\|_2^2 > c \sum_{\mathbf{x} \in \mathcal{X}_0} \|\mathbf{x}\|_2^2$ (see Remark 4 of Xu et al. (2010a)). We thus use the notation: SNR(HR-PCA) "$\gtrapprox$" 1 (see Table 1 with appropriate scales of σ_0 and σ_1). Xu et al. (2012) established the SNR for their Outlier Pursuit (OP) algorithm (equivalently the Low-Leverage Decomposition (LLD) of McCoy and Tropp 2011) in Theorem 1 of their work. Their analysis assumes a deterministic condition, but it is possible to show that this condition is asymptotically valid under the needle-haystack model. Lerman et al. (2012) established w.h.p. the SNR of the REAPER algorithm in Theorem 1 of their work (for simplicity of their expressions they assumed that $d \leq (D-1)/2$). Zhang (2012) established the SNR for Tyler's M-Estimator (TME) in Theorem 1 of his work. His result is deterministic, but it is easy to show that the deterministic condition holds with probability 1 under the needle-haystack model. Hardt and Moitra (2013) proposed randomized and deterministic robust recovery algorithms, RF (or RandomizedFind) and DRF (or DERandomizedFind) respectively, and proved that they obtained the same SNR as in Zhang (2012) under a similar (slightly weaker) combinatorial condition (they only guarantee polynomial time, where Zhang, 2012 specifies a complexity similar to that of

HR-PCA	LLD (OP)	$\hat{\mathrm{L}} := \ker(\hat{\mathbf{Q}})$	REAPER ($d \leq (D-1)/2$)	TME & D/RF
$\frac{\sigma_1\alpha_1}{\sigma_0\alpha_0}$ "$\gtrapprox$" 1	$\frac{\alpha_1}{\alpha_0} \geq \frac{121d}{9}$	$\frac{\alpha_1}{\alpha_0} > 4\frac{\sigma_0}{\sigma_1}\frac{d}{\sqrt{(D-d)D}}$	$\frac{\alpha_1}{\alpha_0} > \frac{\sigma_0}{\sigma_1}\left(C_1\frac{d}{D} - \frac{d}{C_2\alpha_1}\right)$	$\frac{\alpha_1}{\alpha_0} > \frac{d}{D-d}$

Table 1: Theoretical SNR (lowest bound on α_1/α_0) for exact recovery when $N \to \infty$

GMS). We remark that both Zhang (2012) and Hardt and Moitra (2013) appeared after the submission of this manuscript.

The asymptotic SNR of the minimization proposed in this paper is of the same order as that of the REAPER algorithm (which was established for $d \leq (D-1)/2$) and both of them are better than that of the HR-PCA algorithm. The asymptotic SNRs of OP, TME, RF and DRF are independent of σ_1 and σ_0. However, by normalizing all data points to the unit sphere, we may assume that $\sigma_1 = \sigma_0$ in all other algorithms and treat them equally (see Lerman et al., 2012). In this case, the SNR of OP is significantly worse than that of the minimization proposed in here, especially when $d \ll D$ (it is also worse than the weaker SNR specified in (17)). When $d \ll D$, the SNR of TME, RF and DRF is of the same order as the asymptotic SNR of our formulation. However, when d is very close to D, the SNR of our formulation is better than the SNR of TME by a factor of $\sqrt{D}$. We question whether a better asymptotic rate than the one of GMS and REAPER can be obtained by a convex algorithm for robust subspace recovery for the needle-haystack model. Hardt and Moitra (2013) showed that it is small set expansion hard for any algorithm to obtain better SNR than theirs for all scenarios satisfying their combinatorial condition.

We note though that there are non-convex methods for removing outliers with asymptotically zero SNRs. Such SNRs are valid only for the noiseless case and may be differently formulated for detecting the hidden low-dimensional structure among uniform outliers. For example, Arias-Castro et al. (2005) proved that the scan statistics may detect points sampled uniformly from a d-dimensional graph in $\mathbb{R}^D$ of an m-differentiable function among uniform outliers in a cube in $\mathbb{R}^D$ with SNR of order $O(N^{-m(D-d)/(d+m(D-d))})$. Arias-Castro et al. (2011) used higher order spectral clustering affinities to remove outliers and thus detect differentiable surfaces (or certain unions of such surfaces) among uniform outliers with similar SNR to that of the scan statistics. Soltanolkotabi and Candès (2012) removed outliers with "large dictionary coefficients" and showed that this detection works well for outliers uniform in S^{D-1}, inliers uniform in $S^{D-1} \cap \mathrm{L}^*$ and SNR at least $\frac{d}{D} \cdot ((\frac{\alpha_1 N-1}{d})^{\frac{cD}{d}-1} - 1)^{-1}$ (where α_1 is the fraction of inliers) as long as $N < e^{c\sqrt{D}}/D$. For fixed D and d and sufficiently large N, this SNR, which depends on N, can be arbitrarily small. Furthermore, Lerman and Zhang (2010) showed that the global minimizer of (2) (that we relax in this paper so that the minimization is convex) can in theory recover the subspace with asymptotically zero SNR. They also showed that the underlying subspace is a local minimum of (2) with SNR of order $\omega(1/\sqrt{N})$. However, these non-convex procedures do not have efficient or sufficiently fast implementations for subspace recovery. Furthermore, their impressive theoretical estimates often break down in the presence of noise. Indeed, in the noisy case their near-recovery is not better than the one stated for GMS in Theorem 6 (see, e.g., (16) and (17) of Arias-Castro et al. (2011) or Theorem 1.2 of Lerman and Zhang (2010)). On the other hand, in view of Coudron and Lerman (2012) we may obtain significantly better

asymptotic SNR for GMS when the noise is symmetrically distributed with respect to the underlying subspace.

2.6.2 A Special Case with Asymmetric Outliers

In the case of spherically symmetric outliers, PCA cannot exactly recover the underlying subspace, but it can asymptotically recover it (see, e.g., Lerman and Zhang, 2010). In particular, with sufficiently large sample with spherically symmetric outliers, PCA nearly recovers the underlying subspace. We thus slightly modify the two models of §2.6.1 so that the distribution of outliers is asymmetric and show that our combinatorial conditions for exact recovery still hold (with overwhelming probability). On the other hand, the subspace recovered by PCA, when sampling data from these models, is sufficiently far from the underlying subspace for any given sample size.

We first generalize Theorem 5 under a generalized needle-haystack model: Let $\mu = \alpha_0\mu_0 + \alpha_1\mu_1$, $\mu_0 = N(\mathbf{0}, \mathbf{\Sigma}_0/D)$, where $\mathbf{\Sigma}_0$ is an arbitrary positive definite matrix (not necessarily a scalar matrix as before), and as before $\mu_1 = N(\mathbf{0}, \sigma_1^2\mathbf{P}_{\mathrm{L}^*}\mathbf{P}_{\mathrm{L}^*}^T/d)$. We claim that Theorem 5 still holds in this case if we replace σ_0 in the RHS of (17) with $\sqrt{\lambda_{\max}(\mathbf{\Sigma}_0)}$, where $\lambda_{\max}(\mathbf{\Sigma}_0)$ denotes the largest eigenvalue of $\mathbf{\Sigma}_0$ (see justification in §7.6.1).

In order to generalize Theorem 4 for asymmetric outliers, we assume that the outlier component μ_0 of the OIM measure μ is a sub-Gaussian distribution with an arbitrary positive definite covariance matrix $\mathbf{\Sigma}_0$. Following Coudron and Lerman (2012), we define the expected version of F, F_I, and its oracle minimizer, $\hat{\mathbf{Q}}_I$, which is analogous to (8) (the subscript I indicates integral):

$$F_I(\mathbf{Q}) = \int \|\mathbf{Q}\mathbf{x}\| \, \mathrm{d}\mu(x) \tag{19}$$

and

$$\hat{\mathbf{Q}}_I = \underset{\mathbf{Q}\in\mathbb{H},\mathbf{Q}\mathbf{P}_{\mathrm{L}^*}=\mathbf{0}}{\arg\min} F_I(\mathbf{Q}). \tag{20}$$

We assume that $\hat{\mathbf{Q}}_I$ is the unique minimizer in (20) (we remark that the two-subspaces criterion in (25) for the projection of μ onto $L^{*\perp}$ implies this assumption). Under these assumptions Theorem 4 still holds if we multiply the RHS of (16) by the ratio between the largest eigenvalue of $\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_I\mathbf{P}_{\mathrm{L}^{*\perp}}$ and the $(D-d)$th eigenvalue of $\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_I\mathbf{P}_{\mathrm{L}^{*\perp}}$ (see justification in §7.5.1).

2.7 Near Subspace Recovery for Noisy Samples

We show that in the case of sufficiently small additive noise (i.e., the inliers do not lie exactly on the subspace L^* but close to it), the GMS algorithm nearly recovers the underlying subspace.

We use the following notation: $\|\mathbf{A}\|_F$ and $\|\mathbf{A}\|$ denote the Frobenius and spectral norms of $\mathbf{A} \in \mathbb{R}^{k\times l}$ respectively. Furthermore, $\mathbb{H}_1$ denotes the set of all positive semidefinite matrices in $\mathbb{H}$, that is, $\mathbb{H}_1 = \{\mathbf{Q} \in \mathbb{H} : \mathbf{Q} \succcurlyeq 0\}$. We also define the following two constants

$$\gamma_0 = \frac{1}{N} \min_{\mathbf{Q}\in\mathbb{H}_1,\|\mathbf{\Delta}\|_F=1,\mathrm{tr}(\mathbf{\Delta})=0} \sum_{i=1}^{N} \frac{\|\mathbf{\Delta}\mathbf{x}_i\|^2\|\mathbf{Q}\mathbf{x}_i\|^2 - (\mathbf{x}_i^T\mathbf{\Delta}\mathbf{Q}\mathbf{x}_i)^2}{\|\mathbf{Q}\mathbf{x}_i\|^3}, \tag{21}$$

and

$$\gamma_0' = \frac{1}{N} \min_{\mathbf{Q}\in\mathbb{H}_1, \|\mathbf{\Delta}\|=1, \mathrm{tr}(\mathbf{\Delta})=0} \sum_{i=1}^{N} \frac{\|\mathbf{\Delta}\mathbf{x}_i\|^2\|\mathbf{Q}\mathbf{x}_i\|^2 - (\mathbf{x}_i^T\mathbf{\Delta}\mathbf{Q}\mathbf{x}_i)^2}{\|\mathbf{Q}\mathbf{x}_i\|^3}. \tag{22}$$

The sum in the RHS's of (21) and (22) is the following second directional derivative: $\frac{\mathrm{d}^2}{\mathrm{d}t^2}F(\mathbf{Q}+t\mathbf{\Delta})$; when $\mathbf{Q}\mathbf{x}_i = 0$, its ith term can be set to 0. It is interesting to note that both (21) and (22) express the Restricted Strong Convexity (RSC) parameter γ_l of Agarwal et al. (2012b, Definition 1), where their notation translates into ours as follows: $\mathcal{L}_n(\mathbf{Q}) := F(\mathbf{Q})/N$, $\tau_l := 0$, $\Omega' := \mathbb{H}_1$ and $\theta - \theta' := \Delta$. The difference between γ_0 and γ_0' of (21) and (22) is due to the choice of either the Frobenius or the spectral norms respectively for measuring the size of $\theta - \theta'$.

Using this notation, we formulate our noise perturbation result as follows.

Theorem 6 *Assume that $\{\epsilon_i\}_{i=1}^N$ is a set of positive numbers, $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ and $\tilde{\mathcal{X}} = \{\tilde{\mathbf{x}}_i\}_{i=1}^N$ are two data sets such that $\|\tilde{\mathbf{x}}_i - \mathbf{x}_i\| \le \epsilon_i \ \ \forall 1 \le i \le N$ and $\mathcal{X}$ satisfies (14). Let $F_{\mathcal{X}}(\mathbf{Q})$ and $F_{\tilde{\mathcal{X}}}(\mathbf{Q})$ denote the corresponding versions of $F(\mathbf{Q})$ w.r.t. the sets $\mathcal{X}$ and $\tilde{\mathcal{X}}$ and let $\hat{\mathbf{Q}}$ and $\tilde{\mathbf{Q}}$ denote their respective minimizers. Then we have*

$$\|\tilde{\mathbf{Q}} - \hat{\mathbf{Q}}\|_F < \sqrt{2\sum_{i=1}^{N}\epsilon_i/(N\gamma_0)} \quad \textit{and} \quad \|\tilde{\mathbf{Q}} - \hat{\mathbf{Q}}\| < \sqrt{2\sum_{i=1}^{N}\epsilon_i/(N\gamma_0')}. \tag{23}$$

Moreover, if $\tilde{\mathrm{L}}$ and $\hat{\mathrm{L}}$ are the subspaces spanned by the bottom d eigenvectors of $\tilde{\mathbf{Q}}$ and $\hat{\mathbf{Q}}$ respectively and ν_{D-d} is the $(D-d)$th eigengap of $\hat{\mathbf{Q}}$, then

$$\|\mathbf{P}_{\hat{\mathrm{L}}} - \mathbf{P}_{\tilde{\mathrm{L}}}\|_F \le \frac{2\sqrt{2\sum_{i=1}^{N}\epsilon_i/(N\gamma_0)}}{\nu_{D-d}} \quad \textit{and} \quad \|\mathbf{P}_{\hat{\mathrm{L}}} - \mathbf{P}_{\tilde{\mathrm{L}}}\| \le \frac{2\sqrt{2\sum_{i=1}^{N}\epsilon_i/(N\gamma_0')}}{\nu_{D-d}}. \tag{24}$$

We note that if $\mathcal{X}$ and $\tilde{\mathcal{X}}$ satisfy the conditions of Theorem 6, then given the perturbed data set $\tilde{\mathcal{X}}$ and the dimension d, Theorem 6 guarantees that GMS nearly recovers L^*. More interestingly, the theorem also implies that we may properly estimate the dimension of the underlying subspace in this case (we explain this in details in §7.7.1). Such dimension estimation is demonstrated later in Figure 2.

Theorem 6 is a perturbation result in the spirit of the stability analysis by Candès et al. (2006) and Xu et al. (2012, Theorem 2). In order to observe that the statement of Theorem 6 is comparable to that of Theorem 2 of Xu et al. (2012), we note that asymptotically the bounds on the recovery errors in (23) and (24) depend only on the empirical mean of $\{\epsilon_i\}_{i=1}^N$ and do not grow with N. To clarify this point we formulate the following proposition.

Proposition 7 *If $\mathcal{X}$ is i.i.d. sampled from a bounded distribution μ and*

$$\mu(\mathrm{L}_1) + \mu(\mathrm{L}_2) < 1 \ \ \textit{for any two } D-1\textit{-dimensional subspaces } \mathrm{L}_1 \textit{ and } \mathrm{L}_2, \tag{25}$$

then there exist constants $c_0(\mu) > 0$ and $c_0'(\mu) > 0$ depending on μ such that

$$\liminf_{N\to\infty} \gamma_0(\mathcal{X}) \ge c_0(\mu) \quad \textit{and} \quad \liminf_{N\to\infty} \gamma_0'(\mathcal{X}) \ge c_0'(\mu) \ \textit{almost surely.} \tag{26}$$

If (25) is strengthened so that $\mu(\mathrm{L}_1) + \mu(\mathrm{L}_2)$ is sufficiently smaller than 1, then it can be noticed empirically that $c_0(\mu)$ and $c_0'(\mu)$ are sufficiently larger than zero.

Nevertheless, the stability theory of Candès et al. (2006), Xu et al. (2012) and this section is not optimal. Stronger stability results require nontrivial analysis and we leave it to a possible future work. We comment though on some of the deficiencies of our stability theory and their possible improvements.

We first note that the bounds in Theorem 6 are generally not optimal. Indeed, if $\epsilon_i = O(\epsilon)$ for all $1 \leq i \leq N$, then the error bounds in Theorem 6 are $O(\sqrt{\epsilon})$, whereas we empirically noticed that these error bounds are $O(\epsilon)$. In §7.7.2 we sketch a proof for this empirical observation when ϵ is sufficiently small and $\mathrm{rank}(\hat{\mathbf{Q}}) = D$.

The dependence of the error on D, which follows from the dependence of γ_0 and γ_0' on D, is a difficult problem and strongly depends on the underlying distribution of $\mathcal{X}$ and of the noise. For example, in the very special case where the set $\mathcal{X}$ is sampled from a subspace $L_0 \subset \mathbb{R}^D$ of dimension $D_0 < D$, and the noise distribution is such that $\tilde{\mathcal{X}}$ also lies in L_0, then practically we are performing GMS over $P_{L_0}(\mathcal{X})$ and $P_{L_0}(\tilde{\mathcal{X}})$, and the bound in (23) would depend on D_0 instead of D.

Coudron and Lerman (2012) suggested a stronger perturbation analysis and also remarked on the dependence of the error on D in a very special scenario.

2.8 Near Subspace Recovery for Regularized Minimization

For our practical algorithm it is advantageous to regularize the function F as follows (see Theorems 11 and 12 below):

$$F_\delta(\mathbf{Q}) := \sum_{i=1,\|\mathbf{Q}\mathbf{x}_i\|\geq\delta}^{N} \|\mathbf{Q}\mathbf{x}_i\| + \sum_{i=1,\|\mathbf{Q}\mathbf{x}_i\|<\delta}^{N} \left(\frac{\|\mathbf{Q}\mathbf{x}_i\|^2}{2\delta} + \frac{\delta}{2} \right).$$

We remark that other convex algorithms (Candès et al., 2011; Xu et al., 2012; McCoy and Tropp, 2011) also regularize their objective function by adding the term $\delta\|\mathbf{X} - \mathbf{L} - \mathbf{O}\|_F^2$. However, their proofs are not formulated for this regularization.

In order to address the regularization in our case and conclude that the GMS algorithm nearly recovers L^* for the regularized objective function, we adopt a similar perturbation procedure as in §2.7. We denote by $\hat{\mathbf{Q}}_\delta$ and $\hat{\mathbf{Q}}$ the minimizers of $F_\delta(\mathbf{Q})$ and $F(\mathbf{Q})$ in $\mathbb{H}$ respectively. Furthermore, let $\hat{\mathrm{L}}_\delta$ and $\hat{\mathrm{L}}$ denote the subspaces recovered by the bottom d eigenvectors of $\hat{\mathbf{Q}}_\delta$ and $\hat{\mathbf{Q}}$ respectively. Using the constants ν_{D-d} and γ_0 of Theorem 6, the difference between the two minimizers and subspaces can be controlled as follows.

Theorem 8 *If $\mathcal{X}$ is a data set satisfying* (14)*, then*

$$\|\hat{\mathbf{Q}}_\delta - \hat{\mathbf{Q}}\|_F < \sqrt{\delta/2\gamma_0}$$

and

$$\|\mathbf{P}_{\hat{\mathrm{L}}_\delta} - \mathbf{P}_{\hat{\mathrm{L}}}\|_F \leq \frac{2\sqrt{\delta/2\gamma_0}}{\nu_{D-d}}. \tag{27}$$

3. Understanding Our M Estimator: Interpretation and Formal Similarities with Other M Estimators

We highlight the formal similarity of our M-estimator with a common M-estimator and with Tyler's M-estimator in §3.1 and §3.2 respectively. We also show that in view of the standard assumptions on the algorithm for computing the common M-estimator, it may fail in exactly recovering the underlying subspace (see §3.1.1). At last, in §3.3 we interpret our M-estimator as a robust inverse covariance estimator.

3.1 Formal Similarity with the Common M-estimator for Robust Covariance Estimation

A well-known robust M-estimator for the $\mathbf{0}$-centered covariance matrix (Maronna, 1976; Huber and Ronchetti, 2009; Maronna et al., 2006) minimizes the following function over all $D \times D$ positive definite matrices (for some choices of a function ρ)

$$L(\mathbf{A}) = \sum_{i=1}^{N} \rho(\mathbf{x}_i^T \mathbf{A}^{-1} \mathbf{x}_i) - \frac{N}{2} \log(\det(\mathbf{A}^{-1})). \tag{28}$$

The image of the estimated covariance is clearly an estimator to the underlying subspace L^*.

If we set $\rho(x) = \sqrt{x}$ and $\mathbf{A}^{-1} = \mathbf{Q}^2$ then the objective function $L(\mathbf{A})$ in (28) is $\sum_{i=1}^{N} \|\mathbf{Q}\mathbf{x}_i\| - N \log(\det(\mathbf{Q}))$. This energy function is formally similar to our energy function. Indeed, using Lagrangian formulation, the minimizer $\hat{\mathbf{Q}}$ in (4) is also the minimizer of $\sum_{i=1}^{N} \|\mathbf{Q}\mathbf{x}_i\| - \lambda \operatorname{tr}(\mathbf{Q})$ among all $D \times D$ symmetric matrices (or equivalently nonnegative symmetric matrices) for some $\lambda > 0$ (the parameter λ only scales the minimizer and does not effect the recovered subspace). Therefore, the two objective functions differ by their second terms. In the common M-estimator (with $\rho(x) = \sqrt{x}$ and $\mathbf{A}^{-1} = \mathbf{Q}^2$) it is $\log(\det(\mathbf{Q}))$, or equivalently, $\operatorname{tr}(\log(\mathbf{Q}))$, where in our M-estimator, it is $\operatorname{tr}(\mathbf{Q})$.

3.1.1 Problems with Exact Recovery by the Common M-estimator

The common M-estimator is designed for robust covariance estimation, however, we show here that in general it cannot exactly recover the underlying subspace. To make this statement more precise we recall the following uniqueness and existence conditions for the minimizer of (28), which were established by Kent and Tyler (1991): 1) $u = 2\rho'$ is positive, continuous and non-increasing. 2) Condition M: $u(x)x$ is strictly increasing. 3) Condition D_0: For any linear subspace L: $|\mathcal{X} \cap \mathrm{L}|/N < 1 - (D - \dim(\mathrm{L}))/\lim_{x\to\infty} xu(x)$. The following Theorem 9 shows that the uniqueness and existence conditions of the common M-estimator are incompatible with exact recovery.

Theorem 9 *Assume that $d, D \in \mathbb{N}$, $d < D$, $\mathcal{X}$ is a data set in $\mathbb{R}^D$ and $\mathrm{L}^* \in \mathrm{G}(D, d)$ and let $\hat{\mathbf{A}}$ be the minimizer of (28). If conditions M and D_0 hold, then $\operatorname{Im}(\hat{\mathbf{A}}) \neq \mathrm{L}^*$.*

For symmetric outliers (as the ones of §2.6.1) the common M-estimator can still asymptotically achieve exact recovery (similarly to PCA). However, for many scenarios of asymmetric outliers, in particular, the one of §2.6.2, the subspace recovered by the common M-estimator is sufficiently far from the underlying subspace for any given sample size.

We remark that Tyler's M-estimator (Tyler, 1987) can still recover the subspace exactly. This estimator uses $\rho(x) = D\log(x)/2$ in (28) and adds an additional assumption $\mathrm{tr}(\mathbf{A}) = 1$. Zhang (2012) recently showed that this M-estimator satisfies $\mathrm{Im}(\hat{\mathbf{A}}) = \mathrm{L}^*$. However, it does not belong to the class of estimators of Kent and Tyler (1991) addressed by Theorem 9 (it requires that $\mathrm{tr}(\mathbf{A}) = 1$, otherwise it has multiple minimizers; it also does not satisfy condition M).

3.2 Formal Similarity with Tyler's M-Estimator

We show here that the algorithms for our estimator and Tyler's M-estimator (Tyler, 1987) are formally similar. Following Tyler (1987), we write the iterative algorithm for the Tyler's M-estimator for robust covariance estimation as follows:

$$\mathbf{\Sigma}_{n+1} = \sum_{i=1}^{N} \frac{\mathbf{x}_i \mathbf{x}_i^T}{\mathbf{x}_i^T \mathbf{\Sigma}_n^{-1} \mathbf{x}_i} \Big/ \mathrm{tr}\left(\sum_{i=1}^{N} \frac{\mathbf{x}_i \mathbf{x}_i^T}{\mathbf{x}_i^T \mathbf{\Sigma}_n^{-1} \mathbf{x}_i} \right). \tag{29}$$

The unregularized iterative algorithm for GMS is later described in (38). Let us formally substitute $\mathbf{\Sigma} = \mathbf{Q}^{-1}/\mathrm{tr}(\mathbf{Q}^{-1})$ in (38); in view of the later discussion of 3.3, $\mathbf{\Sigma}$ (if exists) can be interpreted as a robust estimator for the covariance matrix (whose top d eigenvectors span the estimated subspace). Then an unregularized version for GMS can be formally written as

$$\mathbf{\Sigma}_{n+1} = \sum_{i=1}^{N} \frac{\mathbf{x}_i \mathbf{x}_i^T}{\|\mathbf{\Sigma}_n^{-1} \mathbf{x}_i\|} \Big/ \mathrm{tr}\left(\sum_{i=1}^{N} \frac{\mathbf{x}_i \mathbf{x}_i^T}{\|\mathbf{\Sigma}_n^{-1} \mathbf{x}_i\|} \right). \tag{30}$$

Clearly, (30) is obtained from (29) by replacing $\mathbf{x}_i^T \mathbf{\Sigma}_n^{-1} \mathbf{x}_i$ with $\|\mathbf{\Sigma}_n^{-1} \mathbf{x}_i\| \equiv \sqrt{\mathbf{x}_i^T \mathbf{\Sigma}_n^{-2} \mathbf{x}_i}$.

3.3 Interpretation of $\hat{\mathbf{Q}}$ as Robust Inverse Covariance Estimator

The total least squares subspace approximation is practically the minimization over $\mathrm{L} \in \mathrm{G}(D, d)$ of the function

$$\sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{P}_{\mathrm{L}} \mathbf{x}_i\|^2 \equiv \sum_{i=1}^{N} \|\mathbf{P}_{\mathrm{L}^\perp} \mathbf{x}_i\|^2 . \tag{31}$$

Its solution is obtained by the span of the top d right vectors of the data matrix $\mathbf{X}$ (whose rows are the data points in $\mathcal{X}$), or equivalently, the top d eigenvectors of the covariance matrix $\mathbf{X}^T\mathbf{X}$. The convex relaxation used in (31) can be also applied to (31) to obtain the following convex minimization problem:

$$\hat{\mathbf{Q}}_2 := \operatorname*{arg\,min}_{\mathbf{Q} \in \mathbb{H}} \sum_{i=1}^{N} \|\mathbf{Q}\mathbf{x}_i\|^2. \tag{32}$$

The "relaxed" total least squares subspace is then obtained by the span of the bottom d eigenvectors of $\hat{\mathbf{Q}}$.

We show here that $\hat{\mathbf{Q}}_2$ coincides with a scaled version of the empirical inverse covariance matrix. This clearly imply that the "relaxed" total least squared subspace coincides with the original one (as the bottom eigenvectors of the inverse empirical covariance are the

top eigenvectors of the empirical covariance). We require though that the data is of full rank so that the empirical inverse covariance is well-defined. This requirement does not hold if the data points are contained within a lower-dimensional subspace, in particular, if their number is smaller than the dimension. We can easily avoid this restriction by initial projection of the data points onto the span of eigenvectors of the covariance matrix with nonzero eigenvalues. That is, by projecting the data onto the lowest-dimensional subspace containing it without losing any information.

Theorem 10 *If* $\mathbf{X}$ *is the data matrix,* $\hat{\mathbf{Q}}_2$ *is the minimizer of* (32) *and* $\operatorname{rank}(\mathbf{X}) = D$ *(equivalently the data points span* $\mathbb{R}^D$*), then*

$$\hat{\mathbf{Q}}_2 = (\mathbf{X}^T\mathbf{X})^{-1}/\operatorname{tr}((\mathbf{X}^T\mathbf{X})^{-1}). \tag{33}$$

We view (4) as a robust version of (32). Since we verified robustness of the subspace recovered by (4) and also showed that (32) yields the inverse covariance matrix, we sometimes refer to the solution of (4) as a robust inverse covariance matrix (though we have only verified robustness to subspace recovery). This idea helps us interpret our numerical procedure for minimizing (4), which we present in §4.

4. IRLS Algorithms for Minimizing (4)

We propose a fast algorithm for computing our M-estimator by using a straightforward iterative re-weighted least squares (IRLS) strategy. We first motivate this strategy in §4.1 (in particular, see (38) and (40)). We then establish its linear convergence in §4.2. At last, we describe its practical choices in §4.3 and summarize its complexity in §4.4.

4.1 Heuristic Proposal for Two IRLS Algorithms

The procedure for minimizing (4) formally follows from the simple fact that the directional derivative of F at $\hat{\mathbf{Q}}$ in any direction $\tilde{\mathbf{Q}} - \hat{\mathbf{Q}}$, where $\tilde{\mathbf{Q}} \in \mathbb{H}$, is 0, that is,

$$\left\langle F'(\hat{\mathbf{Q}})\Big|_{\mathbf{Q}=\hat{\mathbf{Q}}}, \tilde{\mathbf{Q}} - \hat{\mathbf{Q}} \right\rangle_F = 0 \quad \text{for any} \quad \tilde{\mathbf{Q}} \in \mathbb{H}. \tag{34}$$

We remark that since $\mathbb{H}$ is an affine subspace of matrices, (34) holds globally in $\mathbb{H}$ and not just locally around $\hat{\mathbf{Q}}$.

We formally differentiate (4) at $\hat{\mathbf{Q}}$ as follows (see more details in (44), which appears later):

$$F'(\mathbf{Q})\Big|_{\mathbf{Q}=\hat{\mathbf{Q}}} = \sum_{i=1}^{N} \frac{\hat{\mathbf{Q}}\mathbf{x}_i\mathbf{x}_i^T + \mathbf{x}_i\mathbf{x}_i^T\hat{\mathbf{Q}}}{2\|\hat{\mathbf{Q}}\mathbf{x}_i\|}. \tag{35}$$

Throughout the formal derivation we ignore the possibility of zero denominator in (35), that is, we assume that $\hat{\mathbf{Q}}\mathbf{x}_i \neq \mathbf{0}\ \forall\ 1 \leq i \leq N$; we later address this issue.

Since $F'(\hat{\mathbf{Q}})$ is symmetric and $\tilde{\mathbf{Q}} - \hat{\mathbf{Q}}$ can be any symmetric matrix with trace 0, it is easy to note that (34) implies that $F'(\hat{\mathbf{Q}})$ is a scalar matrix (e.g., multiply it by a basis of symmetric matrices with trace 0 whose members have exactly 2 nonzero matrix elements).

That is,

$$\sum_{i=1}^{N} \frac{\hat{\mathbf{Q}}\mathbf{x}_i\mathbf{x}_i^T + \mathbf{x}_i\mathbf{x}_i^T\hat{\mathbf{Q}}}{2\|\hat{\mathbf{Q}}\mathbf{x}_i\|} = c\mathbf{I} \tag{36}$$

for some $c \in \mathbb{R}$. This implies that

$$\hat{\mathbf{Q}} = c\left(\sum_{i=1}^{N} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\|\hat{\mathbf{Q}}\mathbf{x}_i\|}\right)^{-1}. \tag{37}$$

Indeed, we can easily verify that (37) solves (36), furthermore, (36) is a Lyapunov equation whose solution is unique (see, e.g., page 1 of Bhatia and Drissi (2005)). Since $\mathrm{tr}(\hat{\mathbf{Q}}) = 1$, we obtain that

$$\hat{\mathbf{Q}} = \left(\sum_{i=1}^{N} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\|\hat{\mathbf{Q}}\mathbf{x}_i\|}\right)^{-1} / \mathrm{tr}\left(\left(\sum_{i=1}^{N} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\|\hat{\mathbf{Q}}\mathbf{x}_i\|}\right)^{-1}\right),$$

which suggests the following iterative estimate of $\hat{\mathbf{Q}}$:

$$\mathbf{Q}_{k+1} = \left(\sum_{i=1}^{N} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\|\mathbf{Q}_k\mathbf{x}_i\|}\right)^{-1} / \mathrm{tr}\left(\left(\sum_{i=1}^{N} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\|\mathbf{Q}_k\mathbf{x}_i\|}\right)^{-1}\right). \tag{38}$$

Formula (38) is undefined whenever $\mathbf{Q}_k\mathbf{x}_i = \mathbf{0}$ for some $k \in \mathbb{N}$ and $1 \le i \le N$. In theory, we address it as follows. Let $I(\mathbf{Q}) = \{1 \le i \le N : \mathbf{Q}\mathbf{x}_i = \mathbf{0}\}$, $\mathrm{L}(\mathbf{Q}) = \mathrm{Sp}\{\mathbf{x}_i\}_{i\in I(\mathbf{Q})}$ and

$$T(\mathbf{Q}) = \mathbf{P}_{\mathrm{L}(\mathbf{Q})^\perp}\left(\sum_{i\notin I(\mathbf{Q})} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\|\mathbf{Q}\mathbf{x}_i\|}\right)^{-1}\mathbf{P}_{\mathrm{L}(\mathbf{Q})^\perp} / \mathrm{tr}\left(\mathbf{P}_{\mathrm{L}(\mathbf{Q})^\perp}\left(\sum_{i\notin I(\mathbf{Q})} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\|\mathbf{Q}\mathbf{x}_i\|}\right)^{-1}\mathbf{P}_{\mathrm{L}(\mathbf{Q})^\perp}\right).$$

Using this notation, the iterative formula can be corrected as follows

$$\mathbf{Q}_{k+1} = T(\mathbf{Q}_k). \tag{39}$$

In practice, we can avoid data points satisfying $\|\mathbf{Q}_k\mathbf{x}_i\| \le \delta$ for a sufficiently small parameter δ (instead of $\|\mathbf{Q}_k\mathbf{x}_i\| = 0$). We follow a similar idea by replacing F with the regularized function F_δ for a regularized parameter δ. In this case, (39) obtains the following form:

$$\mathbf{Q}_{k+1} = \left(\sum_{i=1}^{N} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\max(\|\mathbf{Q}_k\mathbf{x}_i\|, \delta)}\right)^{-1} / \mathrm{tr}\left(\left(\sum_{i=1}^{N} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\max(\|\mathbf{Q}_k\mathbf{x}_i\|, \delta)}\right)^{-1}\right). \tag{40}$$

We note that the RHS of (39) is obtained as the limit of the RHS of (40) when δ approaches 0.

The two iterative formulas, that is, (39) and (40), give rise to IRLS algorithms. For simplicity of notation, we exemplify this idea with the formal expression in (38). It iteratively finds the solution to the following weighted (with weight $1/\|\mathbf{Q}_k\mathbf{x}_i\|$) least squares problem:

$$\arg\min_{\mathbf{Q}\in\mathbb{H}} \sum_{i=1}^{N} \frac{1}{\|\mathbf{Q}_k\mathbf{x}_i\|}\|\mathbf{Q}\mathbf{x}_i\|^2. \tag{41}$$

To show this, we note that (41) is a quadratic function and any formal directional derivative at $\mathbf{Q}_{k+1}$ is 0. Indeed,

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{Q}}\sum_{i=1}^{N}\frac{1}{\|\mathbf{Q}_k\mathbf{x}_i\|}\|\mathbf{Q}\mathbf{x}_i\|^2\Big|_{\mathbf{Q}=\mathbf{Q}_{k+1}}=\mathbf{Q}_{k+1}\left(\sum_{i=1}^{N}\frac{\mathbf{x}_i\mathbf{x}_i^T}{\|\mathbf{Q}_k\mathbf{x}_i\|}\right)+\left(\sum_{i=1}^{N}\frac{\mathbf{x}_i\mathbf{x}_i^T}{\|\mathbf{Q}_k\mathbf{x}_i\|}\right)\mathbf{Q}_{k+1}=c\mathbf{I}$$

for some $c\in\mathbb{R}$, and $\left\langle \mathbf{I},\tilde{\mathbf{Q}}-\mathbf{Q}_{k+1}\right\rangle_F=0$ for any $\tilde{\mathbf{Q}}\in\mathbb{H}$. Consequently, $\mathbf{Q}_{k+1}$ of (38) is the minimizer of (41).

Formula (40) (as well as (39)) provides another interpretation for $\hat{\mathbf{Q}}$ as robust inverse covariance (in addition to the one discussed in §3.3). Indeed, we note for example that the RHS of (40) is the scaled inverse of a weighted covariance matrix; the scaling enforces the trace of the inverse to be 1 and the weights of $\mathbf{x}_i\mathbf{x}_i^T$ are significantly larger when $\mathbf{x}_i$ is an inlier. In other words, the weights apply a shrinkage procedure for outliers. Indeed, since $\mathbf{Q}_k\mathbf{x}_i$ approaches $\hat{\mathbf{Q}}\mathbf{x}_i$ and the underlying subspace, which contain the inliers, is recovered by $\ker(\hat{\mathbf{Q}})$, for an inlier $\mathbf{x}_i$ the coefficient of $\mathbf{x}_i\mathbf{x}_i^T$ approaches $1/\delta$, which is a very large number (in practice we use $\delta=10^{-20}$). On the other hand, when $\mathbf{x}_i$ is sufficiently far from the underlying subspace, the coefficient of $\mathbf{x}_i\mathbf{x}_i^T$ is significantly smaller.

4.2 Theory: Convergence Analysis of the IRLS Algorithms

The following theorem analyzes the convergence of the sequence proposed by (39) to the minimizer of (4).

Theorem 11 *Let $\mathcal{X}=\{\mathbf{x}_i\}_{i=1}^N$ be a data set in $\mathbb{R}^D$ satisfying* (14)*, $\hat{\mathbf{Q}}$ the minimizer of* (4)*, $\mathbf{Q}_0$ an arbitrary symmetric matrix with $\mathrm{tr}(\mathbf{Q}_0)=1$ and $\{\mathbf{Q}_i\}_{i\in\mathbb{N}}$ the sequence obtained by iteratively applying* (39) *(while initializing it with $\mathbf{Q}_0$), then $\{\mathbf{Q}_i\}_{i\in\mathbb{N}}$ converges to a matrix $\tilde{\mathbf{Q}}\in\mathbb{H}$. If $\tilde{\mathbf{Q}}\mathbf{x}_i\neq\mathbf{0}$ for all $1\le i\le N$, then $\tilde{\mathbf{Q}}=\hat{\mathbf{Q}}$ and furthermore, $\{F(\mathbf{Q}_i)\}_{i\in\mathbb{N}}$ converges linearly to $F(\tilde{\mathbf{Q}})$ and $\{\mathbf{Q}_i\}_{i\in\mathbb{N}}$ converges r-linearly to $\tilde{\mathbf{Q}}$.*

The condition for the linear convergence to $\hat{\mathbf{Q}}$ in Theorem 11 (i.e., $\hat{\mathbf{Q}}\mathbf{x}_i\neq\mathbf{0}$ for all $1\le i\le N$) usually does not occur for noiseless data. This condition is common in IRLS algorithms whose objective functions are l_1-type and are not twice differentiable at $\mathbf{0}$. For example, Weiszfeld's Algorithm (Weiszfeld, 1937) may not converge to the geometric median but to one of the data points (Kuhn, 1973, §3.4). On the other hand, regularized IRLS algorithms often converge linearly to the minimizer of the regularized function. We demonstrate this principle in our case as follows.

Theorem 12 *Let $\mathcal{X}=\{\mathbf{x}_i\}_{i=1}^N$ be a data set in $\mathbb{R}^D$ satisfying* (14)*, $\mathbf{Q}_0$ an arbitrary symmetric matrix with $\mathrm{tr}(\mathbf{Q}_0)=1$ and $\{\mathbf{Q}_i\}_{i\in\mathbb{N}}$ the sequence obtained by iteratively applying* (40) *(while initializing it with $\mathbf{Q}_0$). Then, the sequence $\{F_\delta(\mathbf{Q}_i)\}_{i\in\mathbb{N}}$ converges linearly to the unique minimum of $F_\delta(\mathbf{Q})$, and $\{\mathbf{Q}_i\}_{i\in\mathbb{N}}$ converges r-linearly to the unique minimizer of $F_\delta(\mathbf{Q})$.*

The convergence rate of the iterative application of (40) depends on δ. Following Theorem 6.1 of Chan and Mulet (1999), this rate is at most

$$r(\delta) = \sqrt{\max_{\mathbf{\Delta}=\mathbf{\Delta}^T,\mathrm{tr}(\mathbf{\Delta})=0} \frac{\sum_{i=1,\|\mathbf{Q}_*\mathbf{x}_i\|>\delta}^{N} \frac{(\mathbf{x}_i^T\mathbf{\Delta}\mathbf{Q}_*\mathbf{x}_i)^2}{\|\mathbf{Q}_*\mathbf{x}_i\|^3}}{\sum_{i=1}^{N} \frac{\|\mathbf{\Delta}\mathbf{x}_i\|^2}{\max(\|\mathbf{Q}_*\mathbf{x}_i,\delta\|)}}}.$$

That is, $\|\mathbf{Q}_k - \hat{\mathbf{Q}}\| < C \cdot r(\delta)^k$ for some constant $C > 0$. If (14) holds, then $r(\delta) < 1$ for all $\delta > 0$ and $r(\delta)$ is a non-increasing function. Furthermore, if $\{\mathbf{x}_i \in \mathcal{X} : \|\hat{\mathbf{Q}}\mathbf{x}_i\| \neq 0\}$ satisfies assumption (14), then $\lim_{\delta\to 0} r(\delta) < 1$.

4.3 The Practical Choices for the IRLS Algorithm

Following the theoretical discussion in §4.2 we prefer using the regularized version of the IRLS algorithm. We fix the regularization parameter to be smaller than the rounding error, that is, $\delta = 10^{-20}$, so that the regularization is very close to the original problem (even without regularization the iterative process is stable, but may have few warnings on badly scaled or close to singular matrices). The idea of the algorithm is to iteratively apply (40) with an arbitrary initialization (symmetric with trace 1). We note that in theory $\{F_\delta(\mathbf{Q}_k)\}_{k\in\mathbb{N}}$ is non-increasing (see, e.g., the proof of Theorem 12). However, empirically the sequence decreases when it is within the rounding error to the minimizer. Therefore, we check $F_\delta(\mathbf{Q}_k)$ every four iterations and stop our algorithm when we detect an increase (we noticed empirically that checking every four iterations, instead of every iteration, improves the accuracy of the algorithm). Algorithm 2 summarizes our practical procedure for minimizing (4).

Algorithm 2 Practical and Regularized Minimization of (4)

Input: $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \subseteq \mathbb{R}^D$: data
Output: $\hat{\mathbf{Q}}$: a symmetric matrix in $\mathbb{R}^{D\times D}$ with $\mathrm{tr}(\hat{\mathbf{Q}}) = 1$.
Steps:
- $\delta = 10^{-20}$
- Arbitrarily initialize $\mathbf{Q}_0$ to be a symmetric matrix with $\mathrm{tr}(\mathbf{Q}_0) = 1$
- $k = -1$

repeat
- k=k+1
- $\mathbf{Q}_{k+1} = \left(\sum_{i=1}^{N} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\max(\|\mathbf{Q}_k\mathbf{x}_i\|,\delta)}\right)^{-1} / \mathrm{tr}\left(\left(\sum_{i=1}^{N} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\max(\|\mathbf{Q}_k\mathbf{x}_i\|,\delta)}\right)^{-1}\right)$.

until $F(\mathbf{Q}_{k+1}) > F(\mathbf{Q}_{k-3})$ and $\mathrm{mod}\,(k+1, 4) = 0$
- Output $\hat{\mathbf{Q}} := \mathbf{Q}_k$

4.4 Complexity of Algorithm 2

Each update of Algorithm 2 requires a complexity of order $O(N \cdot D^2)$, due to the sum of N $D \times D$ matrices. Therefore, for n_s iterations the total running time of Algorithm 2 is of order $O(n_s \cdot N \cdot D^2)$. In most of our numerical experiments n_s was less than 40. The storage of this algorithm is $O(N \times D)$, which amounts to storing $\mathcal{X}$. Thus, Algorithm 2 has

the same order of storage and complexity as PCA. In practice, it might be a bit slower due to a larger constant for the actual complexity.

5. Subspace Recovery in Practice

We view the GMS algorithm as a prototype for various subspace recovery algorithms. We discuss here modifications and extensions of this procedure in order to make it even more practical. Sections 5.1 and 5.2 discuss the cases where d is unknown and known respectively; in particular, §5.2.1 proposes the EGMS algorithm when d is known. At last, §5.3 concludes with the computational complexity of the GMS and EGMS algorithms.

5.1 Subspace Recovery without Knowledge of d

In theory, the subspace recovery described here can work without knowing the dimension d. In the noiseless case, one may use (5) to estimate the subspace as guaranteed by Theorem 1. In the case of small noise one can estimate d from the eigenvalues of $\hat{\mathbf{Q}}$ and then apply the GMS algorithm. This strategy is theoretically justified by Theorems 1 and 6 as well as the discussion following (81). The problem is that condition (9) for guaranteeing exact recovery by GMS is restrictive; in particular, it requires the number of outliers to be larger than at least $D - d$ (according to our numerical experiments it is safe to use the lower bound $1.5\,(D-d)$). For practitioners, this is a failure mode of GMS, especially when the dimension of the data set is large (for example, $D > N$).

While this seems to be a strong restriction, we remark that the problem of exact subspace recovery without knowledge of the intrinsic dimension is rather hard and some assumptions on data sets or some knowledge of data parameters would be expected. Other algorithms for this problem, such as Chandrasekaran et al. (2011), Candès et al. (2011), Xu et al. (2010b) and McCoy and Tropp (2011), require estimates of unknown regularization parameters (which often depend on various properties of the data, in particular, the unknown intrinsic dimension) or strong assumptions on the underlying distribution of the outliers or corrupted elements.

We first note that if only conditions (6) and (7) hold, then Theorem 1 still guarantees that the GMS algorithm outputs a subspace containing the underlying subspace. Using some information on the data one may recover the underlying subspace from the outputted subspace containing it, even when dealing with the failure mode.

In the rest of this section we describe several practical solutions for dealing with the failure mode, in particular, with small number of outliers. We later demonstrate them numerically in §6.2 for artificial data and in §6.7 and §6.8 for real data.

Our first practical solution is to reduce the ambient dimension of the data. When the reduction is not too aggressive, it can be performed via PCA. In §5.2.1 we also propose a robust dimensionality reduction which can be used instead. There are two problems with this strategy. First of all, the reduced dimension is another parameter that requires tuning. Second of all, some information may be lost by the dimensionality reduction and thus exact recovery of the underlying subspace is generally impossible.

A second practical solution is to add artificial outliers. The number of added outliers should not be too large (otherwise (6) and (7) will be violated), but they should sufficiently permeate through $\mathbb{R}^D$ so that (9) holds. In practice, the number of outliers can be $2D$,

since empirically (9) holds with high probability when $N_0 = 2(D-d)$. To overcome the possible impact of outliers with arbitrarily large magnitude, we project the data with artificial outliers onto the sphere (following Lerman et al. 2012). Furthermore, if the original data matrix does not have full rank (in particular if $N < D$) we reduce the dimension of the data (by PCA) to be the rank of the data matrix. This dimensionality reduction clearly does not result in any loss of information. We refer to the whole process of initial "lossless dimensionality reduction" (if necessary), addition of $2D$ artificial Gaussian outliers, normalization onto the sphere and application of GMS (with optional estimation of d by the eigenvalues of $\hat{\mathbf{Q}}$) as the GMS2 algorithm. We believe that it is the best practical solution to avoid condition (9) when d is unknown.

A third solution is to regularize our M estimator, that is, to minimize the following objective function with the regularization parameter λ:

$$\hat{\mathbf{Q}} = \underset{\mathrm{tr}(\mathbf{Q})=1, \mathbf{Q}=\mathbf{Q}^T}{\arg\min} \sum_{i=1}^{N} \|\mathbf{Q}\mathbf{x}_i\| + \lambda \|\mathbf{Q}\|_F^2. \tag{42}$$

The IRLS algorithm then becomes

$$\mathbf{Q}_{k+1} = \left(\sum_{i=1}^{N} \frac{\mathbf{x}_i \mathbf{x}_i^T}{\max(\|\mathbf{Q}_k \mathbf{x}_i\|, \delta)} + 2\lambda \mathbf{I} \right)^{-1} / \mathrm{tr} \left(\left(\sum_{i=1}^{N} \frac{\mathbf{x}_i \mathbf{x}_i^T}{\max(\|\mathbf{Q}_k \mathbf{x}_i\|, \delta)} \right)^{-1} + 2\lambda \mathbf{I} \right).$$

We note that if $\lambda = 0$ and there are only few outliers, then in the noiseless case $\dim(\ker(\hat{\mathbf{Q}})) > d$ and in the small noise case the number of significantly small eigenvalues is bigger than d. On the other hand when $\lambda \to \infty$, $\hat{\mathbf{Q}} \to \mathbf{I}/D$, whose kernel is degenerate (similarly, it has no significantly small eigenvalues). Therefore, there exists an appropriate λ for which $\dim(\ker(\hat{\mathbf{Q}}))$ (or the number of significantly small eigenvalues of $\hat{\mathbf{Q}}$) is d. This formulation transforms the estimation of d into estimation of λ. This strategy is in line with other common regularized solutions to this problem (see, e.g., Chandrasekaran et al. 2011; Candès et al. 2011; Xu et al. 2010b; McCoy and Tropp 2011), however, we find it undesirable to estimate a regularization parameter that is hard to interpret in terms of the data.

5.2 Subspace Recovery with Knowledge of d

Knowledge of the intrinsic dimension d can help improve the performance of GMS or suggest completely new variants (especially as GMS always finds a subspace containing the underlying subspace). For example, knowledge of d can be used to carefully estimate the parameter λ of (42), for example, by finding λ yielding exactly a d-dimensional subspace via a bisection procedure.

Lerman et al. (2012) modified the strategy described in here by requiring an additional constraint on the maximal eigenvalue of $\mathbf{Q}$ in (28): $\lambda_{\max}(\mathbf{Q}) \leq \frac{1}{D-d}$ (where $\lambda_{\max}(\mathbf{Q})$ is the largest eigenvalue of $\mathbf{Q}$). This approach has theoretical guarantees, but it comes with the price of additional SVD in each iteration, which makes the algorithm slightly more expensive. Besides, in practice (i.e., noisy setting) this approach requires tuning the upper bound on $\lambda_{\max}(\mathbf{Q})$. Indeed, the solution $\mathbf{Q}'$ to their minimization problem (with $\lambda_{\max}(\mathbf{Q}') \leq 1/(D-d)$ and $\mathrm{tr}(\mathbf{Q}') = 1$) satisfies that $\dim(\ker(\mathbf{Q}')$ is at most d and equals d when $\mathbf{Q}'$ is a

scaled projector operator. They proved that $\dim(\ker(\mathbf{Q}') = d$ for the setting of pure inliers (lying exactly on a subspace) under some conditions avoiding the three types of enemies. However, in practice (especially in noisy cases) the actual subspace often has dimension smaller than d and thus the bound on $\lambda_{\max}(\mathbf{Q})$ has to be tuned as an additional parameter. In some cases, one may take $\lambda_{\max}(\mathbf{Q}) > \frac{1}{D-d}$ and find the subspace according to the bottom d eigenvectors. In other cases, a bisection method on the bound of $\lambda_{\max}(\mathbf{Q})$ provide more accurate results (see related discussion in Lerman et al. (2012, §6.1.6)).

5.2.1 The EGMS Algorithm

We formulate in Algorithm 3 the Extended Geometric Median Subspace (EGMS) algorithm for subspace recovery with known intrinsic dimension.

Algorithm 3 The Extended Geometric Median Subspace Algorithm

Input: $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N \subseteq \mathbb{R}^D$: data, d: dimension of L^*, an algorithm for minimizing (4)
Output: $\hat{\mathrm{L}}$: a d-dimensional linear subspace in $\mathbb{R}^D$.
Steps:
• $\hat{\mathrm{L}} = \mathbb{R}^D$
repeat
• $\hat{\mathbf{Q}} = \arg\min_{\mathbf{Q}\in\mathbb{H},\mathbf{Q}\mathbf{P}_{\hat{\mathrm{L}}^\perp}=\mathbf{0}} F(\mathbf{Q})$
• $\mathbf{u} =$ the top eigenvector of $\hat{\mathbf{Q}}$
• $\hat{\mathrm{L}} = \hat{\mathrm{L}} \cap \mathrm{Sp}(\mathbf{u}^\perp)$
until $\dim(\hat{\mathrm{L}}) = d$

We justify this basic procedure in the noiseless case without requiring (9) as follows.

Theorem 13 *Assume that $d, D \in \mathbb{N}$, $d < D$, $\mathcal{X}$ is a data set in $\mathbb{R}^D$ and $\mathrm{L}^* \in \mathrm{G}(D, d)$. If only conditions* (6) *and* (7) *hold, then the EGMS Algorithm exactly recovers L^*.*

In §6.5 we show how the vectors obtained by EGMS at each iteration can be used to form robust principal components (in reverse order), even when $\hat{\mathbf{Q}}$ is degenerate.

5.3 Computational Complexity of GMS and EGMS

The computational complexity of GMS is of the same order as that of Algorithm 2, that is, $O(n_s \cdot N \cdot D^2)$ (where n_s is the number of required iterations for Algorithm 2). Indeed, after obtaining $\hat{\mathbf{Q}}$, computing L^* by its smallest d eigenvectors takes an order of $O(d \cdot D^2)$ operations.

EGMS on the other hand repeats Algorithm 2 $D - d$ times; therefore it adds an order of $O((D-d) \cdot n_s \cdot N \cdot D^2)$ operations, where n_s denotes the total number of iterations for Algorithm 2. In implementation, we can speed up the EGMS algorithm by excluding the span of some of the top eigenvectors of $\hat{\mathbf{Q}}$ from $\hat{\mathrm{L}}$ (instead of excluding only the top eigenvector in the third step of Algorithm 3). We demonstrate this modified procedure on artificial setting in §6.2.

6. Numerical Experiments

We compare our proposed estimator to other algorithms, while using both synthetic and real data. We also demonstrate the effectiveness of some of our practical proposals. In §6.1 we describe a model for generating synthetic data. Using this model, we respectively demonstrate in §6.2-§6.4 the effectiveness of the following strategies: the practical solutions of §5.1 and §5.2, our estimation of the subspace dimension, and our regularization (more precisely, its effect on the recovery error). In §6.5 we demonstrate the use of our M estimator for robust estimation of eigenvectors of the covariance (or the inverse covariance) matrix. At last, actual comparisons are demonstrated in §6.6-§6.8 for synthetic data, face data and video surveillance data respectively.

6.1 Model for Synthetic Data

In §6.2-§6.4 and §6.6 we generate data from the following model. We randomly choose $\mathrm{L}^* \in \mathrm{G}(D,d)$, sample N_1 inliers from the d-dimensional Multivariate Normal distribution $N(\mathbf{0}, \mathbf{I}_{d\times d})$ on L^* and add N_0 outliers sampled from a uniform distribution on $[0,1]^D$. The outliers are strongly asymmetric around the subspace to make the subspace recovery problem more difficult (Lerman and Zhang, 2010). In some experiments below additional Gaussian noise is considered. When referring to this synthetic data we only need to specify its parameters N_1, N_0, D, d and possibly the standard deviation for the additive noise. For any subspace recovery algorithm (or heuristics), we denote by $\tilde{\mathrm{L}}$ its output (i.e., the estimator for L^*) and measure the corresponding recovery error by $e_{\tilde{\mathrm{L}}} = \|\mathbf{P}_{\tilde{\mathrm{L}}} - \mathbf{P}_{\mathrm{L}^*}\|_F$.

6.2 Demonstration of Practical Solutions of §5.1 and §5.2

We present two different artificial cases, where in one of them condition (9) holds and in the other one it does not hold and test the practical solutions of §5.1 and §5.2 in the second case.

The two cases are the following instances of the synthetic model of §6.1: (a) (N_1, N_0, D, d) = (100 , 100, 100, 20) and (b) $(N_1, N_0, D, d) = (100, 20, 100, 20)$. The GMS algorithm estimates the underlying subspace L^* given $d = 20$ with recovery errors 2.1×10^{-10} and 3.4 in cases (a) and (b) respectively. In case (a) there are sufficiently many outliers (with respect to $D - d$) and the GMS algorithm is successful. We later show in §6.3 that the underlying dimension ($d = 20$) can be easily estimated by the eigenvalues of $\hat{\mathbf{Q}}$. In case (b) $N_0 = 0.25 * (D - d)$, therefore, condition (9) is violated and the GMS algorithm completely fails.

We demonstrate the success of the practical solutions of §5.1 and §5.2 in case (b). We assume that the dimension d is known, though in §6.3 we estimate d correctly for the non-regularized solutions of §5.1. Therefore, these solutions can be also applied without knowing the dimension. If we reduce the dimension of the data set in case (b) from $D = 100$ to $D = 35$ (via PCA; though one can also use EGMS), then GMS (with $d = 20$) achieves a recovery error of 0.23, which indicates that GMS almost recovers the subspace correctly. We remark though that if we reduce the dimension to, for example, $D = 55$, then the GMS algorithm will still fail. We also note that the recovery error is not as attractive as the

ones below; this observation probably indicates that some information was lost during the dimension reduction.

The GMS2 algorithm with $d = 20$ recovers the underlying subspace in case (b) with error 1.2×10^{-10}. This is the method we advocated for when possibly not knowing the intrinsic dimension.

The regularized minimization of (42) with $\lambda = 100$ works well for case (b). In fact, it recovers the subspace as $\ker \hat{\mathbf{Q}}$ (without using its underlying dimension) with error 3.3×10^{-13}. The only issue is how to determine the value of λ. We claimed in §5.2 that if d is known, then λ can be carefully estimated by the bisection method. This is true for this example, in fact, we initially chose λ this way.

We remark that the REAPER algorithm of Lerman et al. (2012) did not perform well for this particular data, though in general it is a very successful solution. The recovery error of the direct REAPER algorithm was 3.725 (and 3.394 for S-REAPER) and the error for its modified version via bisection (relaxing the bound on the largest eigenvalue so that $\dim(\ker(\hat{\mathbf{Q}})) = 20$) was 3.734 (and 3.175 for S-REAPER).

At last we demonstrate the performance of EGMS and its faster heuristic with $d = 20$. The recovery error of the original EGMS for case (b) is only 0.095. We suggested in §5.3 a faster heuristic for EGMS, which can be reformulated as follows: In the third step of Algorithm 3, we replace $\mathbf{u}$ (the top eigenvector of $\hat{\mathbf{Q}}$) with $\mathbf{U}$, the subspace spanned by several top eigenvectors. In the noiseless case, we could let $\mathbf{U}$ be the span of the nonzero eigenvectors of $\hat{\mathbf{Q}}$. This modification of EGMS (for the noiseless case) required only two repetitions of Algorithm 2 and its recovery error was 2.2×10^{-13}. In real data sets with noise we need to determine the number of top eigenvectors spanning $\mathbf{U}$, which makes this modification of EGMS less automatic.

6.3 Demonstration of Dimension Estimation

We test dimension estimation by eigenvalues of $\hat{\mathbf{Q}}$ for cases (a) and (b) of §6.2. The eigenvalues of $\hat{\mathbf{Q}}$ obtained by Algorithm 2 for the two cases are shown in Figure 2. In case (a), the largest logarithmic eigengap (i.e., the largest gap in logarithms of eigenvalues) occurs at 80, so we can correctly estimate that $d = D - 80 = 20$ (the eigenvalues are not zero since Algorithm 2 uses the δ-regularized objective function). However, in case (b) the largest eigengap occurs at 60 and thus mistakenly predicts $d = 40$.

As we discussed in §6.2, the dimension estimation fails here since condition (9) is not satisfied. However, we have verified that if we try any of the solutions proposed in §5.1 then we can correctly recover that $d = 20$ by the logarithmic eigengap. For example, in Figure 2 we demonstrate the logarithms of eigenvalues of $\hat{\mathbf{Q}}$ in case (b) after dimensionality reduction (via PCA) onto dimension $D = 35$ and it is clear that the largest gap is at $d = 20$ (or $D - d = 80$). We obtained similar graphs when using $2D$ artificial outliers (more precisely, the GMS2 algorithm without the final application of the GMS algorithm) or the regularization of (42) with $\lambda = 100$.

6.4 The Effect of the Regularization Parameter δ

We assume a synthetic data set sampled according to the model of §6.1 with $(N_1, N_0, D, d) = (250, 250, 100, 10)$. We use the GMS algorithm with $d = 10$ and different values of the

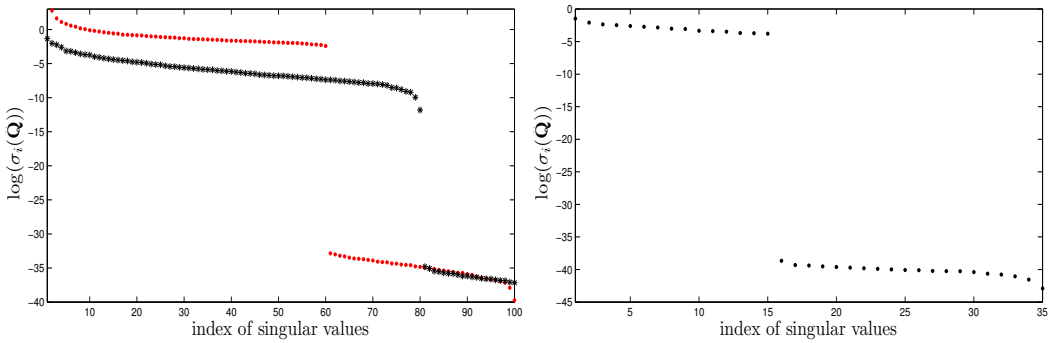


Figure 2: Dimension estimation: In the left figure, the starred points and the dotted point represent log-scaled eigenvalues of the output of Algorithm 2 for cases (a) and (b) respectively (see §6.3). The right figure corresponds to case (b) with dimension reduced to 35.

regularization parameter δ and record the recovery error in Figure 3. For $10^{-14} \leq \delta \leq 10^{-2}$, $\log(\text{error}) - \log(\delta)$ is constant. We thus empirically obtain that the error is of order $O(\delta)$ in this range. On the other hand, (27) only obtained an order of $O(\sqrt{\delta})$. It is possible that methods similar to those of Coudron and Lerman (2012) can obtain sharper error bounds. We also expect that for δ sufficiently small (here smaller than 10^{-14}), the rounding error becomes dominant. On the other hand, perturbation results are often not valid for sufficiently large δ (here this is the case for $\delta > 10^{-2}$).

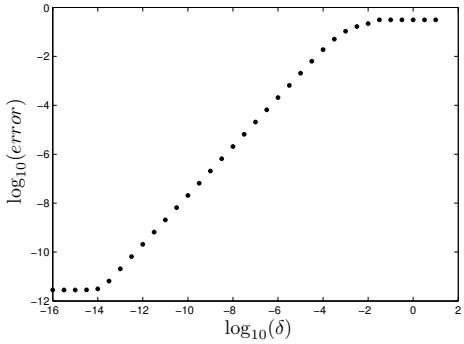


Figure 3: The recovery errors and the regularization parameters δ

6.5 Information Obtained from Eigenvectors

Throughout the paper we emphasized the subspace recovery problem, but did not discuss at all the information that can be inferred from the eigenvectors of our robust PCA strategy. Since in standard PCA these vectors have significant importance, we exemplify the information obtained from our robust PCA and compare it to that obtained from PCA and some other robust PCA algorithms.

We create a sample from a mixture of two Gaussian distributions with the same mean and same eigenvalues of the covariance matrices, but different eigenvectors of the covariance matrices. The mixture percentages are 25% and 75%. We expect the eigenvectors of any good robust PCA algorithm (robust to outliers as perceived in this paper) to be close to that of the covariance of the main component (with 75%).

More precisely, we sample 300 points from $N(\mathbf{0}, \mathbf{\Sigma}_1)$, where $\mathbf{\Sigma}_1$ is a 10×10 diagonal matrix with elements $1, 2^{-1}, 2^{-2}, \cdots, 2^{-9}$ and 100 points from $N(\mathbf{0}, \mathbf{\Sigma}_2)$, where $\mathbf{\Sigma}_2 = \mathbf{U}\mathbf{\Sigma}_1\mathbf{U}^T$, where $\mathbf{U}$ is randomly chosen from the set of all orthogonal matrices in $\mathbb{R}^{10\times 10}$. The goal is to estimate the eigenvectors of $\mathbf{\Sigma}_1$ (i.e., the standard basis vectors in $\mathbb{R}^{10}$) in the presence of 25% "outliers". Unlike the subspace recovery problem, where we can expect to exactly recover a linear structure among many outliers, here the covariance structure is more complex and we cannot exactly recover it with 25% outliers.

We estimated the eigenvectors of $\mathbf{\Sigma}_1$ by the the eigenvectors of $\hat{\mathbf{Q}}$ of Algorithm 2 in reverse order (recall that $\hat{\mathbf{Q}}$ is a scaled and robust version of the inverse covariance). We refer to this procedure as "EVs (eigenvectors) of $\hat{\mathbf{Q}}^{-1}$". We also estimated these eigenvectors by standard PCA, LLD (McCoy and Tropp, 2011) with $\lambda = 0.8\sqrt{D/N}$ and PCP (Candès et al., 2011) with $\lambda = 1/\sqrt{\max(D, N)}$. We repeated the random simulation (with different samples for the random orthogonal matrix $\mathbf{U}$) 100 times and reported in Table 2 the average angles between the estimated and actual top two eigenvectors of $\mathbf{\Sigma}_1$ according to the different methods. We note that the "EVs of $\hat{\mathbf{Q}}^{-1}$" outperforms PCA, LLD (or OP) and PCP in terms of estimation of the top two eigenvectors of $\mathbf{\Sigma}_1$. We remark though that PCP does not suit for robust estimation of the empirical covariance and thus the comparison is unfair for PCP.

	EVs of $\hat{\mathbf{Q}}^{-1}$	LLD	PCP	PCA
Eigenvector 1	3.0°	5.5°	45.7°	14.8°
Eigenvector 2	3.0°	5.5°	47.4°	40.3°

Table 2: Angles (in degrees) between the estimated and actual top two eigenvectors of $\mathbf{\Sigma}_1$.

When the covariance matrix $\mathbf{\Sigma}_1$ (and consequently also $\mathbf{\Sigma}_2$) is degenerate, $\hat{\mathbf{Q}}$ might be singular and therefore $\hat{\mathbf{Q}}$ cannot be directly used to robustly estimate eigenvectors of the covariance matrix. For this case, EGMS (Algorithm 3) can be used, where the vector $\mathbf{u}$ obtained in the ith iteration of Algorithm 3 can be considered as the $(D - i + 1)$st robust eigenvector (that is, we reverse the order again). To test the performance of this method, we modify $\mathbf{\Sigma}_1$ in the above model as follows: $\mathbf{\Sigma}_1$=diag$(1, 0.5, 0.25, 0, 0, \cdots, 0)$. We repeated the random simulations of this modified model 100 times and reported in Table 2 the average angles between the estimated and actual top two eigenvectors of $\mathbf{\Sigma}_1$ according to the different methods. Here LLD did slightly better than EGMS and they both outperformed PCA (and PCP).

	EGMS	LLD	PCP	PCA
Eigenvector 1	5.2°	3.4°	42.6°	8.2°
Eigenvector 2	5.2°	3.4°	47.3°	16.1°

Table 3: Angles (in degrees) between the estimated and actual top two eigenvectors of $\mathbf{\Sigma}_1$.

6.6 Detailed Comparison with Other Algorithms for Synthetic Data

Using the synthetic data of §6.1, we compared the GMS algorithm with the following algorithms: MDR (Mean Absolute Deviation Rounding) of McCoy and Tropp (2011), LLD (Low-Leverage Decomposition) of McCoy and Tropp (2011), OP (Outlier Pursuit) of Xu et al. (2010b), PCP (Principal Component Pursuit) of Candès et al. (2011), MKF (Median K-flats with $K = 1$) of Zhang et al. (2009), HR-PCA (High-dimensional Robust PCA) of Xu et al. (2010a), a common M-estimator (Huber and Ronchetti, 2009, see, e.g.,) and R_1-PCA of Ding et al. (2006). The codes of OP and HR-PCA were obtained from `http://guppy.mpe.nus.edu.sg/~mpexuh`, the code of MKF from `http://www.math.umn.edu/~zhang620/mkf`, the code of PCP from `http://perception.csl.illinois.edu/matrix-rank/sample_code.html` with the Accelerated Proximal Gradient and full SVD version, the codes of MDR and LLD from `http://www.acm.caltech.edu/~mccoy/code/` and the codes of the common M-estimator, R_1-PCA and GMS will appear in a supplemental webpage. We also record the output of standard PCA, where we recover the subspace by the span of the top d eigenvectors. We ran the experiments on a computer with Intel Core 2 CPU at 2.66GHz and 2 GB memory.

We remark that since the basic GMS algorithm already performed very well on these artificial instances, we did not test its extensions and modifications described in §5 (e.g., GMS2 and EGMS).

For all of our experiments with synthetic data, we could correctly estimate d by the largest logarithmic eigengap of the output of Algorithm 2. Nevertheless, we used the knowledge of d for all algorithms for the sake of fair comparison.

For LLD, OP and PCP we estimated L^* by the span of the top d eigenvectors of the low-rank matrix. Similarly, for the common M-estimator we used the span of the top d eigenvectors of the estimated covariance $\mathbf{A}$. For the HR-PCA algorithm we also used the true percentage of outliers (50% in our experiments). For LLD, OP and PCP we set the mixture parameter λ as $0.8\sqrt{D/N}, 0.8\sqrt{D/N}, 1/\sqrt{\max(D,N)}$ respectively (following the suggestions of McCoy and Tropp (2011) for LLD/OP and Candès et al. (2011) for PCP). These choices of parameters are also used in experiments with real data sets in §6.7 and §6.8.

For the common M-estimator, we used $u(x) = 2\max(\ln(x)/x, 10^{30})$ and the algorithm discussed by Kent and Tyler (1991). Considering the conditions in §3.1.1, we also tried other functions: $u(x) = \max(x^{-0.5}, 10^{30})$ had a significantly larger recovery error and $u(x) = \max(x^{-0.9}, 10^{30})$ resulted in a similar recovery error as $\max(\ln(x)/x, 10^{30})$ but a double running time.

We used the syntectic data with different values of (N_1, N_0, D, d). In some instances we also add noise from the Gaussian distribution $N(0, \eta^2 \mathbf{I})$ with $\eta = 0.1$ or 0.01. We repeated each experiment 20 times (due to the random generation of data). We record in Table 4 the mean running time, the mean recovery error and their standard deviations.

We remark that PCP is designed for uniformly corrupted coordinates of data, instead of corrupted data points (i.e., outliers), therefore, the comparison with PCP is somewhat unfair for this kind of data. On the other hand, the applications in §6.7 and §6.8 are tailored for the PCP model (though the other algorithms still apply successfully to them).

From Table 4 we can see that GMS is the fastest robust algorithm. Indeed, its running time is comparable to that of PCA. We note that this is due to its linear convergence rate (usually it converges in less than 40 iterations). The common M-estimator is the closest algorithm in terms of running time to GMS, since it also has the linear convergence rate. In contrast, PCP, OP and LLD need a longer running time since their convergence rates are much slower. Overall, GMS performs best in terms of exact recovery. The PCP, OP and LLD algorithms cannot approach exact recovery even by tuning the parameter λ. For example, in the case where $(N_1, N_0, D, d) = (125, 125, 10, 5)$ with $\eta = 0$, we checked a geometric sequence of 101 λ values from 0.01 to 1, and the smallest recovery errors for LLD, OP and PCP are 0.17, 0.16 and 0.22 respectively. The common M-estimator performed very well for many cases (sometimes slightly better than GMS), but its performance deteriorates as the density of outliers increases (e.g., poor performance for the case where $(N_1, N_0, D, d) = (125, 125, 10, 5)$). Indeed, Theorem 9 indicates problems with the exact recovery of the common M-estimator.

At last, we note that the empirical recovery error of the GMS algorithm for noisy data sets is in the order of $\sqrt{\eta}$, where η is the size of noise.

6.7 Yale Face data

Following Candès et al. (2011), we apply our algorithm to face images. It has been shown that face images from the same person lie in a low-dimensional linear subspace of dimension at most 9 (Basri and Jacobs, 2003). However, cast shadows, specular reflections and saturations could possibly distort this low-rank modeling. Therefore, one can use a good robust PCA algorithm to remove these errors if one has many images from the same face.

We used the images of the first two persons in the extended Yale face database B (Lee et al., 2005), where each of them has 65 images of size 192×168 under different illumination conditions. Therefore we represent each person by 65 vectors of length 32256. Following Basri and Jacobs (2003) we applied GMS, GMS2 and EGMS with $d = 9$ and we also reduced the 65×32256 matrix to 65×65 (in fact, we only reduced the representation of the column space) by rejecting left vectors with zero singular values. We also applied the GMS algorithm after initial dimensionality reduction (via PCA) to $D = 20$. The running times of EGMS and GMS (without dimensionality reduction) are 13 and 0.16 seconds respectively on average for each face (we used the same computer as in §6.6). On the other hand, the running times of PCP and LLD are 193 and 2.7 seconds respectively. Moreover, OP ran out of memory. The recovered images are shown in Figure 4, where the shadow of the nose and the parallel lines were removed best by EGMS. The GMS algorithm without dimension reduction did not perform well, due to the difficulty explained in §5 and demonstrated in

(N_1, N_0, D, d)		GMS	MDR	LLD	OP	PCP	HR-PCA	MKF	PCA	M-est.	R_1-PCA
	e	6e-11	0.275	1.277	0.880	0.605	0.210	0.054	0.193	0.102	0.121
$(125, 125, 10, 5)$	$std.e$	4e-11	0.052	0.344	0.561	0.106	0.049	0.030	0.050	0.037	0.048
$\eta = 0$	$t(s)$	0.008	0.371	0.052	0.300	0.056	0.378	0.514	0.001	0.035	0.020
	$std.t$	0.002	0.120	0.005	0.054	0.002	0.001	0.262	8e-06	4e-04	0.014
	e	0.011	0.292	1.260	1.061	0.567	0.233	0.069	0.213	0.115	0.139
$(125, 125, 10, 5)$	$std.e$	0.004	0.063	0.316	0.491	0.127	0.075	0.036	0.073	0.054	0.073
$\eta = 0.01$	$t(s)$	0.008	0.340	0.053	0.287	0.056	0.380	0.722	0.001	0.035	0.052
	$std.t$	0.001	0.075	0.007	0.033	0.001	0.009	0.364	1e-05	4e-04	0.069
	e	0.076	0.264	1.352	0.719	0.549	0.200	0.099	0.185	0.122	0.128
$(125, 125, 10, 5)$	$std.e$	0.023	0.035	0.161	0.522	0.102	0.051	0.033	0.048	0.041	0.050
$\eta = 0.1$	$t(s)$	0.007	0.332	0.055	0.301	0.056	0.378	0.614	0.001	0.035	0.032
	$std.t$	0.001	0.083	0.004	0.044	0.001	0.001	0.349	7e-06	4e-04	0.037
	e	2e-11	0.652	0.258	0.256	0.261	0.350	0.175	0.350	1e-12	0.307
$(125, 125, 50, 5)$	$std.e$	3e-11	0.042	0.030	0.032	0.033	0.023	0.028	0.025	5e-12	0.029
$\eta = 0$	$t(s)$	0.015	0.420	0.780	1.180	3.164	0.503	0.719	0.006	0.204	0.020
	$std.t$	0.001	0.128	0.978	0.047	0.008	0.055	0.356	9e-05	0.001	0.011
	e	0.061	0.655	0.274	0.271	0.273	0.355	0.196	0.359	0.007	0.321
$(125, 125, 50, 5)$	$std.e$	0.009	0.027	0.039	0.038	0.040	0.038	0.038	0.033	0.001	0.038
$\eta = 0.01$	$t(s)$	0.023	0.401	4.155	1.506	0.499	0.653	0.656	0.006	0.191	0.028
	$std.t$	0.002	0.079	0.065	0.197	0.006	0.044	0.377	8e-05	0.001	0.022
	e	0.252	0.658	0.292	0.290	0.296	0.358	0.264	0.363	0.106	0.326
$(125, 125, 50, 5)$	$std.e$	0.027	0.033	0.032	0.032	0.033	0.027	0.031	0.032	0.014	0.032
$\eta = 0.1$	$t(s)$	0.021	0.363	0.923	1.726	0.501	0.638	0.641	0.006	0.191	0.025
	$std.t$	0.001	0.063	0.033	0.470	0.009	0.051	0.240	1e-04	0.001	0.012
	e	3e-12	0.880	0.214	0.214	0.215	0.332	0.161	0.330	2e-12	0.259
$(250, 250, 100, 10)$	$std.e$	2e-12	0.018	0.019	0.019	0.019	0.014	0.024	0.012	9e-12	0.016
$\eta = 0$	$t(s)$	0.062	1.902	3.143	7.740	2.882	1.780	1.509	0.039	0.819	1.344
	$std.t$	0.006	0.354	4.300	0.038	0.014	0.041	1.041	3e-04	0.023	0.708
	e	0.077	0.885	0.217	0.216	0.219	0.334	0.164	0.335	0.009	0.263
$(250, 250, 100, 10)$	$std.e$	0.006	0.031	0.019	0.018	0.020	0.019	0.019	0.017	3e-04	0.018
$\eta = 0.01$	$t(s)$	0.084	1.907	21.768	11.319	2.923	1.785	1.412	0.039	0.400	1.086
	$std.t$	0.010	0.266	0.261	0.291	0.014	0.041	0.988	3e-04	0.002	0.738
	e	0.225	0.888	0.238	0.237	0.262	0.342	0.231	0.345	0.136	0.276
$(250, 250, 100, 10)$	$std.e$	0.016	0.020	0.019	0.019	0.019	0.019	0.018	0.015	0.010	0.019
$\eta = 0.1$	$t(s)$	0.076	1.917	4.430	16.649	2.876	1.781	1.555	0.039	0.413	1.135
	$std.t$	0.007	0.299	0.069	1.184	0.014	0.025	0.756	4e-04	0.011	0.817
	e	4e-11	1.246	0.162	0.164	0.167	0.381	0.136	0.381	3e-13	0.239
$(500, 500, 200, 20)$	$std.e$	1e-10	0.018	0.011	0.011	0.011	0.010	0.009	0.008	6e-14	0.009
$\eta = 0$	$t(s)$	0.464	23.332	16.778	89.090	16.604	8.602	5.557	0.347	6.517	15.300
	$std.t$	0.024	2.991	0.878	1.836	0.100	0.216	4.810	0.009	0.126	3.509
	e	0.082	1.247	0.160	0.162	0.166	0.374	0.139	0.378	0.012	0.236
$(500, 500, 200, 20)$	$std.e$	0.003	0.018	0.007	0.007	0.008	0.011	0.010	0.006	2e-04	0.007
$\eta = 0.01$	$t(s)$	0.592	23.214	128.51	122.61	16.823	8.541	6.134	0.354	2.361	15.165
	$std.t$	0.060	3.679	1.155	6.500	0.036	0.219	4.318	0.019	0.064	3.485
	e	0.203	1.262	0.204	0.204	0.250	0.391	0.275	0.398	0.166	0.270
$(500, 500, 200, 20)$	$std.e$	0.007	0.012	0.007	0.007	0.007	0.012	0.272	0.009	0.005	0.008
$\eta = 0.1$	$t(s)$	0.563	24.112	24.312	202.22	16.473	8.552	8.745	0.348	2.192	15.150
	$std.t$	0.061	2.362	0.226	8.362	0.050	0.155	3.408	0.010	0.064	3.420

Table 4: Mean running times, recovery errors and their standard deviations for synthetic data.

§6.2. The GMS2 algorithm turns out to work well, except for the second image of face 2. However, other algorithms such as PCP and GMS with dimension reduction ($D = 20$) performed even worse on this image and LLD did not remove any shadow at all; the only good algorithm for this image is EGMS.

Figure 4: Recovering faces: (a) given images, (b)-(f) the recovered images by EGMS, GMS without dimension reduction, GMS2, GMS with dimension reduced to 20, PCP and LLD respectively

6.8 Video Surveillance

For background subtraction in surveillance videos (Li et al., 2004), we consider the following two videos used by Candès et al. (2011): "Lobby in an office building with switching on / off lights" and "Shopping center" from `http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html`. In the first video, the resolution is 160×128 and we used 1546 frames from 'SwitchLight1000.bmp' to 'SwitchLight2545.bmp'. In the second video, the resolution is 320×256 and we use 1000 frames from 'ShoppingMall1001.bmp' to 'ShoppingMall2000.bmp'. Therefore, the data matrices are of size 1546×20480 and 1001×81920. We used a computer with Intel Core 2 Quad Q6600 2.4GHz and 8 GB memory due to the large size of these data.

We applied GMS, GMS2 and EGMS with $d = 3$ and with initial dimensionality reduction to 200 to reduce running time. For this data we are unaware of a standard choice of d; though we noticed empirically that the outputs of our algorithms as well as other algorithms are very stable to changes in d within the range $2 \leq d \leq 5$. We obtain the foreground by the orthogonal projection to the recovered 3-dimensional subspace. Figure 5 demonstrates foregrounds detected by EGMS, GMS, GMS2, PCP and LLD, where PCP and LLD used $\lambda = 1/\sqrt{\max(D, N)}, 0.8\sqrt{D/N}$. We remark that OP ran out of memory. Using truth labels provided in the data, we also form ROC curves for GMS, GMS2, EGMS and PCP in Figure 6 (LLD is not included since it performed poorly for any value of λ we tried). We note that PCP performs better than both GMS and EGMS in the 'Shoppingmall' video, whereas the latter algorithms perform better than PCP in the 'SwitchLight' video. Furthermore, GMS is significantly faster than EGMS and PCP. Indeed, the running times (on average) of GMS, EGMS and PCP are 91.2, 1018.8 and 1209.4 seconds respectively.

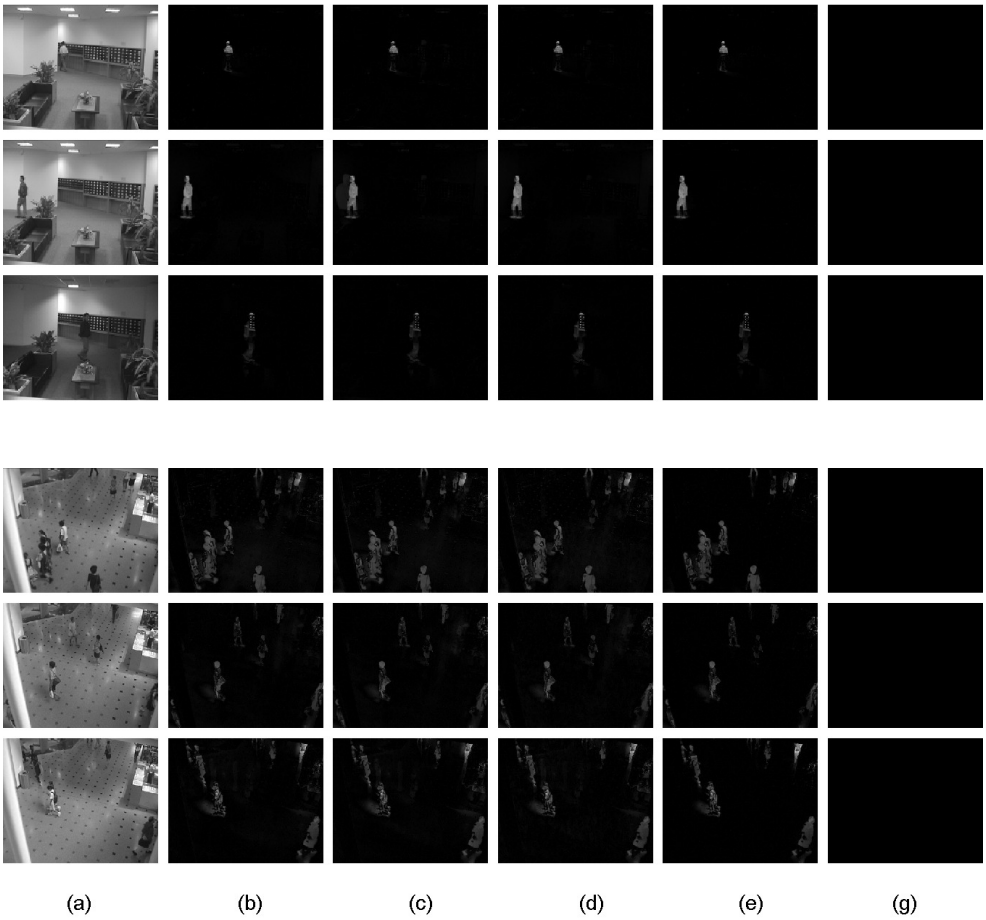


Figure 5: Video surveillance: (a) the given frames (b)-(e) the detected foreground by EGMS, GMS, GMS2, PCP, LLD respectively

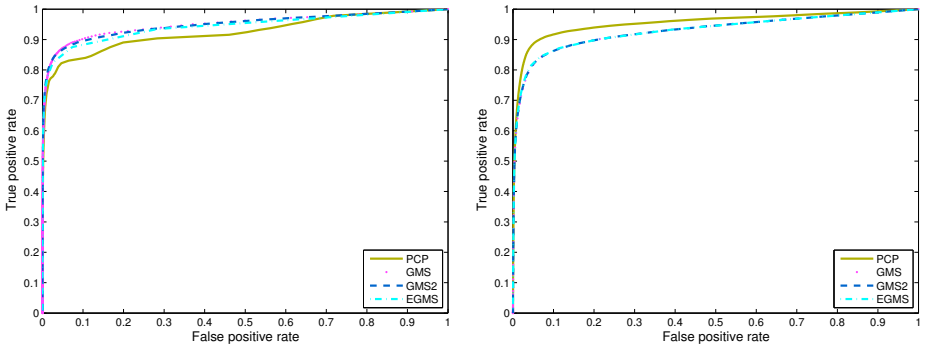


Figure 6: ROC curves for EGMS, GMS, GMS2 and PCP in the 'SwitchLight' video (the left figure) and the 'Shoppingmall' video (the right figure)

7. Proofs of Theorems

We present the technical proofs of the theoretical statements of this paper according to their order of appearance.

7.1 Proof of Theorem 1

We will prove that if conditions (6) and (7) hold, then the set of all minimizers satisfying (4) coincides with the set of all minimizers satisfying (8). This clearly implies that if conditions (6) and (7) hold, then any minimizer $\hat{\mathbf{Q}}$ of (4) satisfies $\ker(\hat{\mathbf{Q}}) \supseteq \mathrm{L}^*$ (indeed, this condition is equivalent with the condition $\mathbf{Q}\mathbf{P}_{\mathrm{L}^*} = \mathbf{0}$, which appears in the formulation of (8)). If condition (9) also holds, then $\ker(\hat{\mathbf{Q}}) = \mathrm{L}^*$ and the theorem is concluded.

We assume that conditions (6) and (7) hold and arbitrarily fix a minimizer $\hat{\mathbf{Q}}_0$ of the oracle problem (8). We claim that in order to establish the equivalence of the sets of solutions of (4) and (8), it is sufficient to prove that

$$F(\hat{\mathbf{Q}}_0 + \mathbf{\Delta}) - F(\hat{\mathbf{Q}}_0) > 0 \text{ for any symmetric } \mathbf{\Delta} \text{ with } \mathrm{tr}(\mathbf{\Delta}) = 0 \text{ and } \mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*} \neq \mathbf{0}. \quad (43)$$

Indeed, we first note that (43) implies that $\hat{\mathbf{Q}}_0$ is also a minimizer of (4). This observation follows from combining (43) with the following equation:

$$F(\hat{\mathbf{Q}}_0 + \mathbf{\Delta}) - F(\hat{\mathbf{Q}}_0) \geq 0 \text{ for any symmetric } \mathbf{\Delta} \text{ with } \mathrm{tr}(\mathbf{\Delta}) = 0 \text{ and } \mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*} = \mathbf{0},$$

which is an immediate consequence of the definition of (8). To conclude the equivalence, we assume on the contrary that there exists $\hat{\mathbf{Q}}_0$, which is a minimizer of (8) but not a minimizer of (4). We denote by $\hat{\mathbf{Q}}_0'$ a minimizer of (8), which is also a minimizer of (4) and let $\mathbf{\Delta} := \hat{\mathbf{Q}}_0' - \hat{\mathbf{Q}}_0$. Then by the definitions of $\hat{\mathbf{Q}}_0$, $\hat{\mathbf{Q}}_0'$ and $\mathbf{\Delta}$: $\mathrm{tr}(\mathbf{\Delta}) = 0$, $\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*} \neq \mathbf{0}$ and $F(\hat{\mathbf{Q}}_0') = F(\hat{\mathbf{Q}}_0)$. This contradicts (43) and thus concludes the proof.

In order to conclude (43) (and thus the theorem) we first differentiate $\|\mathbf{Q}\mathbf{x}\|$ at $\mathbf{Q} = \mathbf{Q}_0$ when $\mathbf{x} \in \ker(\mathbf{Q}_0)^{\perp}$ as follows:

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{Q}}\|\mathbf{Q}\mathbf{x}\|\Big|_{\mathbf{Q}=\mathbf{Q}_0} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{Q}}\sqrt{\|\mathbf{Q}\mathbf{x}\|^2}\Big|_{\mathbf{Q}=\mathbf{Q}_0} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{Q}}\frac{\mathbf{Q}\mathbf{x}\mathbf{x}^T\mathbf{Q}^T}{2\|\mathbf{Q}_0\mathbf{x}\|}\Big|_{\mathbf{Q}=\mathbf{Q}_0} = \frac{\mathbf{Q}_0\mathbf{x}\mathbf{x}^T + \mathbf{x}\mathbf{x}^T\mathbf{Q}_0}{2\|\mathbf{Q}_0\mathbf{x}\|}. \quad (44)$$

We note that for any $\mathbf{x} \in \mathbb{R}^D \setminus \{\mathbf{0}\}$ satisfying $\hat{\mathbf{Q}}_0\mathbf{x} \neq \mathbf{0}$ and $\mathbf{\Delta} \in \mathbb{R}^{D\times D}$ symmetric:

$$\|(\hat{\mathbf{Q}}_0 + \mathbf{\Delta})\mathbf{x}\| - \|\hat{\mathbf{Q}}_0\mathbf{x}\| \geq \left\langle \mathbf{\Delta}, (\hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T + \mathbf{x}\mathbf{x}^T\hat{\mathbf{Q}}_0)/2\|\hat{\mathbf{Q}}_0\mathbf{x}\| \right\rangle_F = \left\langle \mathbf{\Delta}, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T/\|\hat{\mathbf{Q}}_0\mathbf{x}\| \right\rangle_F. \tag{45}$$

Indeed, the first equality follows from (44) and the convexity of $\|\mathbf{Q}\mathbf{x}\|$ in $\mathbf{Q}$ and the second equality follows from the symmetry of $\mathbf{\Delta}$ and $\hat{\mathbf{Q}}_0$ as well as the definition of the Frobenius dot product.

If on the other hand $\hat{\mathbf{Q}}_0\mathbf{x} = \mathbf{0}$, then clearly

$$\|(\hat{\mathbf{Q}}_0 + \mathbf{\Delta})\mathbf{x}\| - \|\hat{\mathbf{Q}}_0\mathbf{x}\| = \|\mathbf{\Delta}\mathbf{x}\|. \tag{46}$$

For simplicity of our presentation, we use (46) only for $\mathbf{x} \in \mathcal{X}_1$ (where obviously $\hat{\mathbf{Q}}_0\mathbf{x} = \mathbf{0}$ since $\hat{\mathbf{Q}}_0\mathbf{P}_{\mathrm{L}^*} = 0$). On the other hand, we use (45) for all $\mathbf{x} \in \mathcal{X}_0$. One can easily check that if $\mathbf{x} \in \mathcal{X}_0$ and $\hat{\mathbf{Q}}_0\mathbf{x} = \mathbf{0}$, then replacing (45) with (46) does not change the analysis below. Using these observations we note that

$$F(\hat{\mathbf{Q}}_0 + \mathbf{\Delta}) - F(\hat{\mathbf{Q}}_0) \geq \sum_{\mathbf{x}\in\mathcal{X}_1} \|\mathbf{\Delta}\mathbf{x}\| + \sum_{\mathbf{x}\in\mathcal{X}_0} \left\langle \mathbf{\Delta}, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T/\|\hat{\mathbf{Q}}_0\mathbf{x}\| \right\rangle_F. \tag{47}$$

We assume first that $\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*} = \mathbf{0}$. In this case, $\hat{\mathbf{Q}}_0 + \mathbf{\Delta} \in \mathbb{H}$ and $(\hat{\mathbf{Q}}_0 + \mathbf{\Delta})\mathbf{P}_{\mathrm{L}^*} = \mathbf{0}$. Since $\hat{\mathbf{Q}}_0$ is the minimizer of (8), we obtain the following identity (which is analogous to (34)):

$$\sum_{\mathbf{x}\in\mathcal{X}_0} \left\langle \mathbf{\Delta}, \frac{\hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T}{\|\hat{\mathbf{Q}}_0\mathbf{x}\|} \right\rangle_F \geq 0 \quad \forall\ \mathbf{\Delta} \in \mathbb{R}^{D\times D} \text{ s.t. } \mathrm{tr}(\mathbf{\Delta}) = 0\ , \mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*} = \mathbf{0}. \tag{48}$$

We will prove (43) by showing that the RHS of (47) is positive for any symmetric $\mathbf{\Delta}$ with $\mathrm{tr}(\mathbf{\Delta}) = 0$ and $\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*} \neq \mathbf{0}$. Using (47) and the facts that $\mathcal{X}_1 \subset \mathrm{L}^*$ and $\hat{\mathbf{Q}}_0 = \mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_0$ (since $\mathbf{P}_{\mathrm{L}^*}\hat{\mathbf{Q}}_0 = \hat{\mathbf{Q}}_0\mathbf{P}_{\mathrm{L}^*} = \mathbf{0}$), we establish the following inequality:

$$\begin{aligned}
&F(\hat{\mathbf{Q}}_0 + \mathbf{\Delta}) - F(\hat{\mathbf{Q}}_0) \geq \sum_{\mathbf{x}\in\mathcal{X}_1} \|\mathbf{\Delta}\mathbf{x}\| + \sum_{\mathbf{x}\in\mathcal{X}_0} \left\langle \mathbf{\Delta}, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T/\|\hat{\mathbf{Q}}_0\mathbf{x}\| \right\rangle_F \\
&= \sum_{\mathbf{x}\in\mathcal{X}_1} \|\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}\mathbf{x}\| + \sum_{\mathbf{x}\in\mathcal{X}_0} \left\langle (\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*} + \mathbf{\Delta}\mathbf{P}_{\mathrm{L}^{*\perp}}), \mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T/\|\hat{\mathbf{Q}}_0\mathbf{x}\| \right\rangle_F \\
&\geq \sum_{\mathbf{x}\in\mathcal{X}_1} \left(\|\mathbf{P}_{\mathrm{L}^*}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}\mathbf{x}\| + \|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}\mathbf{x}\|\right)/\sqrt{2} \\
&+ \sum_{\mathbf{x}\in\mathcal{X}_0} \left\langle (\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*} + \mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^{*\perp}}), \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T/\|\hat{\mathbf{Q}}_0\mathbf{x}\| \right\rangle_F.
\end{aligned} \tag{49}$$

For ease of notation we denote $\mathbf{\Delta}_0 = \mathrm{tr}(\mathbf{P}_{\mathrm{L}^*}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*})\mathbf{v}_0\mathbf{v}_0^T$, where $\mathbf{v}_0$ is the minimizer of the RHS of (6). Combining the following two facts: $\mathrm{tr}(\mathbf{\Delta}_0) - \mathrm{tr}(\mathbf{P}_{\mathrm{L}^*}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}) = 0$ and $\mathrm{tr}(\mathbf{P}_{\mathrm{L}^*}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}) + \mathrm{tr}(\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^{*\perp}}) = \mathrm{tr}(\mathbf{\Delta}) = 0$, we obtain that

$$\mathrm{tr}(\mathbf{\Delta}_0 + \mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^{*\perp}}) = 0.$$

Further application of (48) implies that

$$\sum_{\mathbf{x}\in\mathcal{X}_0} \left\langle \mathbf{\Delta}_0 + \mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^{*\perp}}, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T/\|\hat{\mathbf{Q}}_0\mathbf{x}\| \right\rangle_F \geq 0. \tag{50}$$

We note that

$$\begin{aligned}&\left\langle \mathbf{P}_{\mathrm{L}^{*\perp}}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^{*\perp}}, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\rangle_F = \left\langle \mathbf{P}_{\mathrm{L}^{*\perp}}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{P}_{\mathrm{L}^{*\perp}}, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\rangle_F\\ =&\left\langle \mathbf{P}_{\mathrm{L}^{*\perp}}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^{*\perp}}, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\rangle_F. \end{aligned} \tag{51}$$

Combining (50) and (51) we conclude that

$$\begin{aligned}&-\sum_{\mathbf{x}\in\mathcal{X}_0}\left\langle \mathbf{P}_{\mathrm{L}^{*\perp}}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^{*\perp}}, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\rangle_F \le \sum_{\mathbf{x}\in\mathcal{X}_0}\left\langle \boldsymbol{\Delta}_0, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\rangle_F\\ =&\sum_{\mathbf{x}\in\mathcal{X}_0}\operatorname{tr}(\mathbf{P}_{\mathrm{L}^*}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*})(\mathbf{v}_0^T\hat{\mathbf{Q}}_0\mathbf{x}/\|\hat{\mathbf{Q}}_0\mathbf{x}\|)(\mathbf{v}_0^T\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}) \le |\operatorname{tr}(\mathbf{P}_{\mathrm{L}^*}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*})|\sum_{\mathbf{x}\in\mathcal{X}_0}|\mathbf{v}_0^T\mathbf{x}|. \end{aligned} \tag{52}$$

We apply (52) and then use (6) with $\mathbf{Q}=\mathbf{P}_{\mathrm{L}^*}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*}/\operatorname{tr}(\mathbf{P}_{\mathrm{L}^*}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*})$ to obtain the inequality:

$$\begin{aligned}&\sum_{\mathbf{x}\in\mathcal{X}_1}\|\mathbf{P}_{\mathrm{L}^*}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*}\mathbf{x}\|/\sqrt{2}+\sum_{\mathbf{x}\in\mathcal{X}_0}\left\langle \mathbf{P}_{\mathrm{L}^{*\perp}}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^{*\perp}}, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\rangle_F\\ \ge&\sum_{\mathbf{x}\in\mathcal{X}_1}\|\mathbf{P}_{\mathrm{L}^*}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*}\mathbf{x}\|/\sqrt{2}-|\operatorname{tr}(\mathbf{P}_{\mathrm{L}^*}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*})|\sum_{\mathbf{x}\in\mathcal{X}_0}|\mathbf{v}_0^T\mathbf{x}|>0. \end{aligned} \tag{53}$$

We define $\mathbb{H}_1=\{\mathbf{Q}\in\mathbb{H}:\ \mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}=\mathbf{0}\}$ and claim that (7) leads to the following inequality:

$$\sum_{\mathbf{x}\in\mathcal{X}_1}\|\mathbf{Q}(\mathbf{P}_{\mathrm{L}^*}\mathbf{x})\|>\sqrt{2}\sum_{\mathbf{x}\in\mathcal{X}_0}\|\mathbf{Q}(\mathbf{P}_{\mathrm{L}^*}\mathbf{x})\|\quad\forall\mathbf{Q}\in\mathbb{H}_1. \tag{54}$$

Indeed, since the RHS of (54) is a convex function of $\mathbf{Q}$, its maximum is achieved at the set of all extreme points of $\mathbb{H}_1$, which is $\{\mathbf{Q}\in\mathbb{R}^{D\times D}:\mathbf{Q}=\mathbf{v}\mathbf{v}^T,\text{where }\mathbf{v}\in\mathrm{L}^*,\|\mathbf{v}\|=1\}$. Therefore the maximum of the RHS of (54) is the RHS of (7). Since the minimum of the LHS of (54) is also the LHS of (7), (54) is proved.

We also claim that (54) can be extended from $\mathbb{H}_1$ to all $\mathbf{Q}\in\mathbb{R}^{D\times D}$ such that $\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}=\mathbf{0}$. Indeed, for any $\mathbf{Q}\in\mathbb{R}^{D\times D}$ satisfying $\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}=\mathbf{0}$ and having the SVD decomposition $\mathbf{Q}=\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, we can assign the following matrix $\mathbf{Q}'=\mathbf{Q}'(\mathbf{Q})\in\mathbb{H}_1$: $\mathbf{Q}':=\mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T/\operatorname{tr}(\mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T)$. It is not hard to note that the inequality in (54) holds for $\mathbf{Q}$ if and only if it holds for $\mathbf{Q}'$.

By first applying Cauchy's inequality, then using the defining property of projections and at last applying (54) with $\mathbf{Q}=\mathbf{P}_{\mathrm{L}^{*\perp}}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*}$ (while using its latter extension beyond $\mathbb{H}_1$), we obtain the inequality:

$$\begin{aligned}&\sum_{\mathbf{x}\in\mathcal{X}_1}\|\mathbf{P}_{\mathrm{L}^{*\perp}}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*}\mathbf{x}\|/\sqrt{2}+\sum_{\mathbf{x}\in\mathcal{X}_0}\left\langle \mathbf{P}_{\mathrm{L}^{*\perp}}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*}, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\rangle_F\\ \ge&\sum_{\mathbf{x}\in\mathcal{X}_1}\|\mathbf{P}_{\mathrm{L}^{*\perp}}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*}\mathbf{x}\|/\sqrt{2}-\sum_{\mathbf{x}\in\mathcal{X}_0}\|\mathbf{P}_{\mathrm{L}^{*\perp}}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*}\mathbf{x}\|\\ =&\sum_{\mathbf{x}\in\mathcal{X}_1}\|\mathbf{P}_{\mathrm{L}^{*\perp}}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*}(\mathbf{P}_{\mathrm{L}^*}\mathbf{x})\|/\sqrt{2}-\sum_{\mathbf{x}\in\mathcal{X}_0}\|\mathbf{P}_{\mathrm{L}^{*\perp}}\boldsymbol{\Delta}\mathbf{P}_{\mathrm{L}^*}(\mathbf{P}_{\mathrm{L}^*}\mathbf{x})\|>0. \end{aligned} \tag{55}$$

Finally, we combine (53) and (55) and conclude that the RHS of (49) is nonnegative and consequently (43) holds.

7.2 Proof of Theorem 2

Assume on the contrary that F is not strictly convex, in particular, there exists $0 < t_0 < 1$ such that

$$t_0 \cdot F(\mathbf{Q}_1) + (1-t_0) \cdot F(\mathbf{Q}_2) = F(t_0 \cdot \mathbf{Q}_1 + (1-t_0) \cdot \mathbf{Q}_2) \;\text{ for }\; \mathbf{Q}_1 \neq \mathbf{Q}_2,$$

or equivalently,

$$t_0 \cdot \sum_{i=1}^{N} \|\mathbf{Q}_1 \mathbf{x}_i\| + (1-t_0) \cdot \sum_{i=1}^{N} \|\mathbf{Q}_2 \mathbf{x}_i\| = \sum_{i=1}^{N} \|(t_0 \cdot \mathbf{Q}_1 + (1-t_0) \cdot \mathbf{Q}_2)\mathbf{x}_i\|. \tag{56}$$

Combining (56) with the fact that $\|\mathbf{Q}_1 \mathbf{x}_i\| + \|\mathbf{Q}_2 \mathbf{x}_i\| \geq \|(\mathbf{Q}_1 + \mathbf{Q}_2)\mathbf{x}_i\|$, we obtain that $t_0 \cdot \|\mathbf{Q}_1 \mathbf{x}_i\| + (1-t_0) \cdot \|\mathbf{Q}_2 \mathbf{x}_i\| = \|(t_0 \cdot \mathbf{Q}_1 + (1-t_0) \cdot \mathbf{Q}_2)\mathbf{x}_i\|$ for any $1 \leq i \leq N$ and therefore there exists a sequence $\{c_i\}_{i=1}^{N} \subset \mathbb{R}$ such that

$$\mathbf{Q}_2 \mathbf{x}_i = \mathbf{0} \;\text{ or }\; \mathbf{Q}_1 \mathbf{x}_i = c_i \, \mathbf{Q}_2 \mathbf{x}_i \;\text{ for all } 1 \leq i \leq N. \tag{57}$$

We conclude Theorem 2 by considering two different cases. We first assume that $\ker(\mathbf{Q}_1) = \ker(\mathbf{Q}_2)$. We denote

$$\tilde{\mathbf{Q}}_1 = \mathbf{P}_{\ker(\mathbf{Q}_1)^\perp} \mathbf{Q}_1 \mathbf{P}_{\ker(\mathbf{Q}_1)^\perp} \text{ and } \tilde{\mathbf{Q}}_2 = \mathbf{P}_{\ker(\mathbf{Q}_1)^\perp} \mathbf{Q}_2 \mathbf{P}_{\ker(\mathbf{Q}_1)^\perp}.$$

It follows from (57) that

$$\tilde{\mathbf{Q}}_1(\mathbf{P}_{\ker(\mathbf{Q}_1)^\perp} \mathbf{x}_i) = c_i \, \tilde{\mathbf{Q}}_2(\mathbf{P}_{\ker(\mathbf{Q}_1)^\perp} \mathbf{x}_i)$$

and consequently that $\mathbf{P}_{\ker(\mathbf{Q}_1)^\perp} \mathbf{x}_i$ lies in one of the eigenspaces of $\tilde{\mathbf{Q}}_1^{-1} \tilde{\mathbf{Q}}_2$. We claim that $\tilde{\mathbf{Q}}_1^{-1} \tilde{\mathbf{Q}}_2$ is a scalar matrix. Indeed, if on the contrary $\tilde{\mathbf{Q}}_1^{-1} \tilde{\mathbf{Q}}_2$ is not a scalar matrix, then $\{\mathbf{P}_{\ker(\mathbf{Q}_1)^\perp} \mathbf{x}_i\}_{i=1}^{N}$ lies in a union of several eigenspaces with dimensions summing to $\dim(\mathbf{P}_{\ker(\mathbf{Q}_1)^\perp})$ and this contradicts (14). In view of this property of $\tilde{\mathbf{Q}}_1^{-1} \tilde{\mathbf{Q}}_2$ and the fact that $\mathrm{tr}(\tilde{\mathbf{Q}}_1) = \mathrm{tr}(\hat{\mathbf{Q}}_1) = 1$ we have that $\tilde{\mathbf{Q}}_1 = \tilde{\mathbf{Q}}_2$ and $\mathbf{Q}_1 = \mathbf{Q}_2$, which contradicts our current assumption.

Next, assume that $\ker(\mathbf{Q}_1) \neq \ker(\mathbf{Q}_2)$. We will first show that if $1 \leq i \leq N$ is arbitrarily fixed, then $\mathbf{x}_i \in \ker(\mathbf{Q}_2) \cup \ker(\mathbf{P}_{\ker(\mathbf{Q}_1)} \mathbf{Q}_2)$. Indeed, if $\mathbf{x}_i \notin \ker(\mathbf{Q}_2)$, then using (57) we have $\mathbf{Q}_1 \mathbf{x}_i = c_i \, \mathbf{Q}_2 \mathbf{x}_i$. This implies that $c_i \mathbf{P}_{\ker(\mathbf{Q}_1)} \mathbf{Q}_2 \mathbf{x}_i = \mathbf{P}_{\ker(\mathbf{Q}_1)} \mathbf{Q}_1 \mathbf{x}_i = \mathbf{0}$ and thus $\mathbf{x}_i \in \ker(\mathbf{P}_{\ker(\mathbf{Q}_1)} \mathbf{Q}_2)$. That is, $\mathcal{X}$ is contained in the union of the 2 subspaces $\ker(\mathbf{Q}_2)$ and $\ker(\mathbf{P}_{\ker(\mathbf{Q}_1)} \mathbf{Q}_2)$. The dimensions of both spaces are less than D. This obvious for $\ker(\mathbf{Q}_2)$, since $\mathrm{tr}(\mathbf{Q}_2) = 1$. For $\ker(\mathbf{P}_{\ker(\mathbf{Q}_1)} \mathbf{Q}_2)$ it follows from the fact that $\ker(\mathbf{Q}_1) \neq \ker(\mathbf{Q}_2)$ and thus $\mathbf{P}_{\ker(\mathbf{Q}_1)} \mathbf{Q}_2 \neq \mathbf{0}$. We thus obtained a contradiction to (14).

7.3 Verification of (10) and (11) as Sufficient Conditions and (12) and (13) as Necessary Ones

We revisit the proof of Theorem 1 and first show that (10) and (11) can replace (6) and (7) in the first part of Theorem 1. We only deal with the first part of Theorem 1, which assumes that (9) holds, since (9) guarantees that (10) and (11) are well-defined (see the discussion in §2.4.1).

To show that (11) can replace (7), we prove the inequality in (55) using (11) as follows. Assuming that the SVD of $\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}$ is $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, then $\mathbf{Q}' := \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T/\operatorname{tr}(\mathbf{\Sigma})$ satisfies $\mathbf{Q}' \in \mathbb{H}, \mathbf{Q}'\mathbf{P}_{\mathrm{L}^{*\perp}} = \mathbf{0}$ and $\|\mathbf{Q}'\mathbf{x}\| = \|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}\mathbf{x}\|/\operatorname{tr}(\mathbf{\Sigma}) = \|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}\mathbf{x}\|/\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}\|_*$. Using this fact, we obtain that

$$\sum_{\mathbf{x}\in\mathcal{X}_1}\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}\mathbf{x}\| \geq \|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}\|_* \min_{\mathbf{Q}'\in\mathbb{H},\mathbf{Q}'\mathbf{P}_{\mathrm{L}^{*\perp}}=\mathbf{0}} \sum_{\mathbf{x}\in\mathcal{X}_1}\|\mathbf{Q}'\mathbf{x}\|. \tag{58}$$

We also note that

$$\begin{aligned}\sum_{\mathbf{x}\in\mathcal{X}_0}\left\langle \mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\rangle_F &= \sum_{\mathbf{x}\in\mathcal{X}_0}\left\langle \mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}, \hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^*}/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\rangle_F \\ \geq -\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{\Delta}\mathbf{P}_{\mathrm{L}^*}\|_* &\left\|\sum_{\mathbf{x}\in\mathcal{X}_0}\hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^*}/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\|.\end{aligned} \tag{59}$$

Therefore (55) follows from (11), (58) and (59). Similarly, one can show that (10) may replace (6).

One can also verify that (12) and (13) are necessary conditions for exact recovery by revisiting the proof of Theorem 1 and reversing inequalities.

7.4 Proof of Lemma 3

We first note by symmetry that the minimizer of the LHS of (16) for the needle-haystack model is $\mathbf{Q} = \mathbf{P}_{\mathrm{L}^*}/d$. We can thus rewrite (16) in this case as $\alpha_1 \mathbb{E}\, r_1/d > 2\sqrt{2}\alpha_0 \mathbb{E}\, r_0/(D-d)$, where the "radii" r_1 and r_0 are the norms of the normal distributions with covariances $\sigma_1^2 d^{-1}\mathbf{P}_{\mathrm{L}^*}$ and $\sigma_0^2 D^{-1}\mathbf{P}_{\mathrm{L}^{*\perp}}$ respectively. Let $\tilde{r}_1$ and $\tilde{r}_2$ be the χ-distributed random variables with d and $D-d$ degrees of freedoms, then (16) obtains the form

$$\frac{\alpha_1\sigma_1}{d\sqrt{d}}\mathbb{E}\,\tilde{r}_1 > \frac{2\sqrt{2}\alpha_0\sigma_0}{(D-d)\sqrt{D}}\mathbb{E}\,\tilde{r}_0.$$

Applying (B.7) of Lerman et al. (2012), $\mathbb{E}\,\tilde{r}_1 \geq \sqrt{d/2}$ and $\mathbb{E}\,\tilde{r}_0 \leq \sqrt{D-d}$. Therefore (16) follows from (15).

7.5 Proof of Theorem 4

For simplicity of the proof we first assume that the supports of μ_0 and μ_1 are contained in a ball centered at the origin of radius M.

We start with the proof of (9) "in expectation" and then extend it to hold with high probability. We use the notation $F_I(\mathbf{Q})$ and $\hat{\mathbf{Q}}_I$ defined in (19) and (20) respectively. The spherical symmetry of $\mu_{0,\mathrm{L}^{*\perp}}$ implies that

$$\hat{\mathbf{Q}}_I = \frac{1}{D-d}\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{P}_{\mathrm{L}^{*\perp}}^T \tag{60}$$

is the unique minimizer of (20). To see this formally, we first note that $\mu_{0,\mathrm{L}^{*\perp}}$ satisfies the two-subspaces criterion of Coudron and Lerman (2012) for any $0 < \gamma \leq 1$ (this criterion

generalizes (14) of this paper to continuous measures) and thus by Theorem 2.1 of Coudron and Lerman (2012) (whose proof follows directly the one of Theorem 2 here) the solution of this minimization must be unique. On the other hand, any application of an arbitrary rotation of L^* (within $\mathbb{R}^D$) to the minimizer expressed in the RHS of (20) should also be a minimizer of the RHS of (20). We note that $\frac{1}{D-d}\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{P}_{\mathrm{L}^{*\perp}}^T$ is the only element in the domain of this minimization that is preserved under any rotation of L^*. Therefore, due to uniqueness, this can be the only solution of this minimization problem.

Let

$$\mathbb{H}_2 = \{\mathbf{Q} \in \mathbb{H} : \ \mathbf{Q}\mathbf{P}_{\mathrm{L}^*} = \mathbf{0}, \ \mathbf{Q} \succeq \mathbf{0} \text{ and } \mathrm{cond}(\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}) \geq 2\}, \tag{61}$$

where $\mathbf{Q} \succeq \mathbf{0}$ denotes the positive semidefiniteness of $\mathbf{Q}$ and $\mathrm{cond}(\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}})$ denotes the condition number of this matrix, that is, the ratio between the largest and lowest eigenvalues of $\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}$, or equivalently, the ratio between the top eigenvalue and the $(D-d)$th eigenvalue of $\mathbf{Q}$. Since $\hat{\mathbf{Q}}_I$ is the unique minimizer of (20) and $\hat{\mathbf{Q}}_I \notin \mathbb{H}_2$, then

$$c_1 := \min_{\mathbf{Q}\in\mathbb{H}_2} (F_I(\mathbf{Q}) - F_I(\hat{\mathbf{Q}}_I)) > 0. \tag{62}$$

We note that if $\mathbf{x}$ is a random variable sampled from μ and $\mathbf{Q} \in \mathbb{H}$ (so that $\|\mathbf{Q}\| \leq \|\mathbf{Q}\|_* = 1$), then $\|\mathbf{Q}\mathbf{x}\| \leq M$. Applying this fact, (62) and Hoeffding's inequality, we conclude that for any fixed $\mathbf{Q} \in \mathbb{H}_2$

$$F(\mathbf{Q}) - F(\hat{\mathbf{Q}}_I) > c_1 N/2 \ \text{ w.p. } 1 - \exp(-c_1^2 N/2M^2). \tag{63}$$

We also observe that

$$F(\mathbf{Q}_1) - F(\mathbf{Q}_2) \leq \|\mathbf{Q}_1 - \mathbf{Q}_2\| \sum_{i=1}^{N} \|\mathbf{x}\| \leq \|\mathbf{Q}_1 - \mathbf{Q}_2\| N\, M\,. \tag{64}$$

Combining (63) and (64), we obtain that for all $\mathbf{Q}$ in a ball of radius $r_1 := c_1/2M$ centered around a fixed element in $\mathbb{H}_2$: $F(\mathbf{Q}) - F(\hat{\mathbf{Q}}_I) > 0$ w.p. $1 - \exp(-c_1^2 N/2M^2)$.

We thus cover the compact space $\mathbb{H}_2$ by an r_1-net. Denoting the corresponding covering number by $N(\mathbb{H}_2, r_1)$ and using the above observation we note that w.p. $1 - N(\mathbb{H}_2, r_1)\exp(-c_1^2 N/2M^2)$

$$F(\mathbf{Q}) - F(\hat{\mathbf{Q}}_I) > 0 \quad \text{for all } \mathbf{Q} \in \mathbb{H}_2. \tag{65}$$

The definition of $\hat{\mathbf{Q}}_0$ (that is, (8)) implies that $F(\hat{\mathbf{Q}}_0) \leq F(\hat{\mathbf{Q}}_I)$. Combining this observation with (65), we conclude that w.h.p. $\hat{\mathbf{Q}}_0 \notin \mathbb{H}_2$. We also claim that $\hat{\mathbf{Q}}_0 \succeq \mathbf{0}$ (see, e.g., the proof of Lemma 14, which appears later). Since $\hat{\mathbf{Q}}_0 \notin \mathbb{H}_2$ and $\hat{\mathbf{Q}}_0 \succeq \mathbf{0}$, $\hat{\mathbf{Q}}_0$ satisfies the following property w.h.p.:

$$\mathrm{cond}(\mathbf{P}_{\mathrm{L}^{*\perp}}^T \hat{\mathbf{Q}}_0 \mathbf{P}_{\mathrm{L}^{*\perp}}^T) < 2. \tag{66}$$

Consequently, (9) holds w.h.p. (more precisely, w.p. $1 - N(\mathbb{H}_2, c_1/2M)\exp(-c_1^2 N/2M^2)$).

Next, we verify (10) w.h.p. as follows. Since $\hat{\mathbf{Q}}_0$ is symmetric and $\hat{\mathbf{Q}}_0\mathbf{P}_{\mathrm{L}^*} = \mathbf{0}$ (see (8)), then

$$\hat{\mathbf{Q}}_0 = \mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_0\mathbf{P}_{\mathrm{L}^{*\perp}}. \tag{67}$$

Applying (67), basic inequalities of operators' norms and (66), we bound the RHS of (10) from above as follows:

$$\begin{aligned}
&\sqrt{2}\left\|\sum_{\mathbf{x}\in\mathcal{X}_0}\hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\| = \sqrt{2}\left\|\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_0\mathbf{P}_{\mathrm{L}^{*\perp}}\cdot\sum_{\mathbf{x}\in\mathcal{X}_0}\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\| \\
\le&\sqrt{2}\cdot\left\|\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_0\mathbf{P}_{\mathrm{L}^{*\perp}}\right\|\cdot\left\|\sum_{\mathbf{x}\in\mathcal{X}_0}\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\hat{\mathbf{Q}}_0\mathbf{x}\|\right\| \qquad (68)\\
\le&\sqrt{2}\cdot\lambda_{\max}(\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_0\mathbf{P}_{\mathrm{L}^{*\perp}})\cdot\|\sum_{\mathbf{x}\in\mathcal{X}_0}\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\lambda_{\min}(\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_0\mathbf{P}_{\mathrm{L}^{*\perp}})\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|\| \\
<&\sqrt{8}\left\|\sum_{\mathbf{x}\in\mathcal{X}_0}\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|\right\| = \max_{\mathbf{u}\in S^{D-1}\cap\mathrm{L}^{*\perp}}\sqrt{8}\mathbf{u}^T(\sum_{\mathbf{x}\in\mathcal{X}_0}\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|)\mathbf{u}.
\end{aligned}$$

Therefore to prove (10), we only need to prove that with high probability

$$\min_{\mathbf{Q}\in\mathbb{H},\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}=\mathbf{0}}\sum_{\mathbf{x}\in\mathcal{X}_1}\|\mathbf{Q}\mathbf{x}\| > \max_{\mathbf{u}\in S^{D-1}\cap\mathrm{L}^{*\perp}}\sqrt{8}\mathbf{u}^T(\sum_{\mathbf{x}\in\mathcal{X}_0}\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|)\mathbf{u}. \qquad (69)$$

We will prove that the LHS and RHS of (69) concentrates w.h.p. around the LHS and RHS of (16) respectively and consequently verify (69) w.h.p. Let ϵ_1 be the difference between the RHS and LHS of (69). Theorem 1 of Coudron and Lerman (2012) implies that the LHS of (69) is within distance $\epsilon_1/4$ to the RHS of (16) with probability $1-C\exp(-N/C)$ (where C is a constant depending on ϵ_1, μ and its parameters).

The concentration of the RHS of (16) can be concluded as follows. The spherical symmetry of $\mu_{0,\mathrm{L}^{*\perp}}$ implies that the expectation (w.r.t. μ_0) of $\sum_{\mathbf{x}\in\mathcal{X}_0}\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|$ is a scalar matrix within $\mathrm{L}^{*\perp}$, that is, it equals $\rho_\mu\,\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|$ for some $\rho_\mu\in\mathbb{R}$. We observe that

$$\mathbb{E}_{\mu_0}\operatorname{tr}(\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|) = \mathbb{E}_{\mu_0}\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|$$

and thus conclude that $\rho_\mu = \mathbb{E}_{\mu_0}\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|/(D-d)$. Therefore, for any $\mathbf{u}\in S^{D-1}\cap\mathrm{L}^{*\perp}$

$$\mathbb{E}_{\mu_0}\mathbf{u}^T(\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|)\mathbf{u} = \mathbb{E}_{\mu_0}\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|/(D-d) = \int\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|\,\mathrm{d}\mu_0(\mathbf{x})/(D-d). \qquad (70)$$

We thus conclude from (70) and Hoeffding's inequality that for any fixed $\mathbf{u}\in S^{D-1}\cap\mathrm{L}^{*\perp}$ the function $\sqrt{8}\mathbf{u}^T(\sum_{\mathbf{x}\in\mathcal{X}_0}\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|)\mathbf{u}$ is within distance $\epsilon_1/4$ to the RHS of (16) with probability $1-C\exp(-N/C)$ (where C is a constant depending on ϵ_1, μ and its parameters). Furthermore, applying ϵ-nets and covering (i.e., union bounds) arguments with regards to $S^{D-1}\cap\mathrm{L}^{*\perp}$, we obtain that for all $\mathbf{u}\in S^{D-1}\cap\mathrm{L}^{*\perp}$, $\sqrt{8}\mathbf{u}^T(\sum_{\mathbf{x}\in\mathcal{X}_0}\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^{*\perp}}/\|\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{x}\|)\mathbf{u}$ is within distance $\epsilon_1/2$ to the RHS of (16) with probability $1-C\exp(-N/C)$ (where C is a constant depending on ϵ_1, μ and its parameters). In particular, the RHS of (69) is within distance $\epsilon_1/2$ to the RHS of (16) with the same probability. We thus conclude (69) with probability $1-C'\exp(-N/C')$.

Similarly we can also prove (11), noting that the expectation (w.r.t. μ_0) of $\hat{\mathbf{Q}}_0\mathbf{x}\mathbf{x}^T\mathbf{P}_{\mathrm{L}^*}/\|\hat{\mathbf{Q}}_0\mathbf{x}\|$ is $\mathbf{0}$, since $\hat{\mathbf{Q}}_0\mathbf{x}/\|\hat{\mathbf{Q}}_0\mathbf{x}\|$ and $\mathbf{x}^T\mathbf{P}_{\mathrm{L}^*}$ are independent when $\mathbf{x}$ is restricted to lie in the complement of L^* (that is, $\mathbf{x}\in\mathcal{X}_0$).

If we remove the assumption of bounded supports (with radius M), then we need to replace Hoeffding's inequality with the Hoeffding-type inequality for sub-Gaussian measures of Proposition 5.10 of Vershynin (2012), where in this proposition $a_i = 1$ for all $1 \leq i \leq n$.

We emphasize that our probabilistic estimates are rather loose and can be interpreted as near-asymptotic; we thus did not fully specify their constants. We clarify this point for the probability estimate we have for (9), that is, $1 - N(\mathbb{H}_2, c_1/2M)\exp(-c_1^2 N/2M^2)$. Its constant $N(\mathbb{H}_2, r_1)$ can be bounded from above by the covering number $N(\mathbb{H}_0, r_1)$ of the larger set $\mathbb{H}_0 = \{\mathbf{Q} \in \mathbb{R}^{D\times D} : |\mathbf{Q}_{i,i}| \leq 1\}$, which is bounded from above by $(8/r_1)^{D(D-1)/2}$ (see, e.g., Lemma 5.2 of Vershynin, 2012). This is clearly a very loose estimate that cannot reveal interesting information, such as, the right dependence of N on D and d in order to obtain a sufficiently small probability.

At last, we explain why (14) holds with probability 1 if there are at least $2D-1$ outliers. We denote the set of outliers by $\{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_{N_0}\}$, where $N_0 \geq 2D-1$, and assume on the contrary that (14) holds with probability smaller than 1. Then, there exists a sequence $\{i_j\}_{j=1}^{D-1} \subset \{1, 2, 3, \cdots, N_0\}$ such that the subspace spanned by the $D-1$ points $\mathbf{y}_{i_1}, \mathbf{y}_{i_2}, \cdots, \mathbf{y}_{i_{D-1}}$ contains another outlier with positive probability. However, this is not true for haystack model and thus our claim is proved.

7.5.1 Proof of the Extension of Theorem 4 to the Asymmetric Case

We recall our assumptions that μ_0 is a sub-Gaussian distribution with covariance $\mathbf{\Sigma}_0$ and that $\hat{\mathbf{Q}}_I$ is unique. We follow the proof of Theorem 4 in §7.5 with the following changes. First of all, we replace the requirement

$$\mathrm{cond}(\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}) \geq 2. \tag{71}$$

in (61) with the following one:

$$\mathrm{cond}(\mathbf{P}_{\mathrm{L}^{*\perp}}\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}) \geq 2 \cdot \mathrm{cond}(\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_I\mathbf{P}_{\mathrm{L}^{*\perp}}). \tag{72}$$

We note that (71) follows from (72) in the symmetric case. Indeed, in this case the expression of $\hat{\mathbf{Q}}_I$ in (60) implies that the RHS of (72) is 2. Similarly, instead of (66) we prove that

$$\mathrm{cond}(\mathbf{P}^T_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_0\mathbf{P}^T_{\mathrm{L}^{*\perp}}) < 2 \cdot \mathrm{cond}(\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_I\mathbf{P}_{\mathrm{L}^{*\perp}}).$$

Second of all, in the third inequality of (68) the term

$$\sqrt{2}\,\lambda_{\max}(\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_0\mathbf{P}_{\mathrm{L}^{*\perp}})/\lambda_{\max}(\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_0\mathbf{P}_{\mathrm{L}^{*\perp}})$$

needs to be bounded above by $\sqrt{8}\,\mathrm{cond}(\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_I\mathbf{P}_{\mathrm{L}^{*\perp}})$, instead of $\sqrt{8}$. We can thus conclude the revised theorem, in particular, the last modification in the proof clarifies why we need to multiply the RHS of (16) by $\mathrm{cond}(\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_I\mathbf{P}_{\mathrm{L}^{*\perp}})$, which is the ratio between the largest eigenvalue of $\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_I\mathbf{P}_{\mathrm{L}^{*\perp}}$ and the $(D-d)$th eigenvalue of $\mathbf{P}_{\mathrm{L}^{*\perp}}\hat{\mathbf{Q}}_I\mathbf{P}_{\mathrm{L}^{*\perp}}$.

7.6 Proof of Theorem 5

This proof follows ideas of Lerman et al. (2012). We bound from below the LHS of (7) by applying (A.15) of Lerman et al. (2012) as follows

$$\min_{\mathbf{Q}\in\mathbb{H},\mathbf{Q}\mathbf{P}_{\mathrm{L}^{*\perp}}=\mathbf{0}} \sum_{\mathbf{x}\in\mathcal{X}_1} \|\mathbf{Q}\mathbf{x}\| \geq \frac{1}{\sqrt{d}} \min_{\mathbf{v}\in\mathrm{L}^*,\|\mathbf{v}\|=1} \sum_{\mathbf{x}\in\mathcal{X}_1} |\mathbf{v}^T\mathbf{x}|. \tag{73}$$

We denote the number of inliers sampled from μ_1 by N_1 and the number of outliers sampled from μ_0 by $N_0(=N-N_1)$. We bound from below w.h.p. the RHS of (73) by applying Lemma B.2 of Lerman et al. (2012) in the following way:

$$\frac{1}{\sqrt{d}} \min_{\mathbf{v}\in \mathrm{L}^*, \|\mathbf{v}\|=1} \sum_{\mathbf{x}\in\mathcal{X}_1} |\mathbf{v}^T\mathbf{x}| \geq \frac{\sigma_1}{d}\Big(\sqrt{2/\pi}N_1 - 2\sqrt{N_1 d} - t\sqrt{N_1}\Big) \text{ w.p. } 1-e^{-t^2/2}. \tag{74}$$

By following the proof of Lemma B.2 of Lerman et al. (2012) we bound from above w.h.p. the RHS of (7) as follows

$$\max_{\mathbf{v}\in \mathrm{L}^*, \|\mathbf{v}\|=1} \sum_{\mathbf{x}\in\mathcal{X}_0} |\mathbf{v}^T\mathbf{x}| \leq \frac{\sigma_0}{\sqrt{D}}\Big(\sqrt{2/\pi}N_0 + 2\sqrt{N_0 d} + t\sqrt{N_0}\Big) \text{ w.p. } 1-e^{-t^2/2}. \tag{75}$$

We need to show w.h.p. that the RHS of (75) is strictly less than the RHS of (74). We note that Hoeffding's inequality implies that

$$N_1 > \alpha_1 N/2 \text{ w.p. } 1-e^{-\alpha_1^2 N/2} \text{ and } |N_0 - \alpha_0 N| < \alpha_0 N/2 \text{ w.p. } 1-2e^{-\alpha_0^2 N/2}. \tag{76}$$

Furthermore, (18) and (76) imply that

$$d < N_1/4 \text{ w.p. } 1-e^{-\alpha_1^2 N/2} \text{ and } d < N_0/4 \text{ w.p. } 1-e^{-\alpha_0^2 N/2}. \tag{77}$$

Substituting $t = \sqrt{N_1}/10$ ($> \sqrt{\alpha_1 N}/20$ w.p. $1-e^{-\alpha_1^2 N/2}$) in (74) and $t = \sqrt{N_0}/10$ ($> \sqrt{\alpha_0 N}/20$ w.p. $1-2e^{-\alpha_0^2 N/2}$) in (75) and combining (17) and (73)-(77), we obtain that (7) holds w.p. $1-e^{-\alpha_1^2 N/2} - 2e^{-\alpha_0^2 N/2} - e^{-\alpha_1 N/800} - e^{-\alpha_0 N/800}$. We can similarly obtain that (6) holds with the same probability.

7.6.1 Proof of the Extension of Theorem 5 to the Asymmetric Case

We assume the generalized needle-haystack model of §2.6.2. The proof of Theorem 5 in §7.6 immediately extends to this model, where σ_0 in the RHS of (75) needs to be replaced with $\sqrt{\lambda_{\max}(\mathbf{\Sigma}_0)}$ (recall that $\lambda_{\max}(\mathbf{\Sigma}_0)$ denotes the largest eigenvalue of $\mathbf{\Sigma}_0$). Consequently, Theorem 5 still holds in this case when replacing σ_0 in the RHS of (17) with $\sqrt{\lambda_{\max}(\mathbf{\Sigma}_0)}$.

7.7 Proof of Theorem 6

We first establish the following lemma.

Lemma 14 *The minimizer of $F(\mathbf{Q})$, $\hat{\mathbf{Q}}$, is a semi-definite positive matrix.*

Proof We assume that $\hat{\mathbf{Q}}$ has some negative eigenvalues and show that this assumption contradicts the defining property of $\hat{\mathbf{Q}}$, that is, being the minimizer of $F(\mathbf{Q})$. We denote the eigenvalue decomposition of $\hat{\mathbf{Q}}$ by $\hat{\mathbf{Q}} = \mathbf{V}_{\hat{\mathbf{Q}}}\mathbf{\Sigma}_{\hat{\mathbf{Q}}}\mathbf{V}_{\hat{\mathbf{Q}}}^T$ and define $\mathbf{\Sigma}_{\hat{\mathbf{Q}}}^+ = \max(\mathbf{\Sigma}_{\hat{\mathbf{Q}}}, 0)$ and $\hat{\mathbf{Q}}^+ = \mathbf{V}_{\hat{\mathbf{Q}}}\mathbf{\Sigma}_{\hat{\mathbf{Q}}}^+\mathbf{V}_{\hat{\mathbf{Q}}}^T / \operatorname{tr}(\mathbf{\Sigma}_{\hat{\mathbf{Q}}}^+) \in \mathbb{H}$. Then $\operatorname{tr}(\mathbf{\Sigma}_{\hat{\mathbf{Q}}}^+) > \operatorname{tr}(\mathbf{\Sigma}_{\hat{\mathbf{Q}}}) = \operatorname{tr}(\hat{\mathbf{Q}}) = 1$ and for any $\mathbf{x}\in\mathbb{R}^D$ we have

$$\|\hat{\mathbf{Q}}^+\mathbf{x}\| < \operatorname{tr}(\mathbf{\Sigma}_{\hat{\mathbf{Q}}}^+)\|\hat{\mathbf{Q}}^+\mathbf{x}\| = \|\mathbf{\Sigma}_{\hat{\mathbf{Q}}}^+(\mathbf{V}_{\hat{\mathbf{Q}}}^T\mathbf{x})\| \leq \|\mathbf{\Sigma}_{\hat{\mathbf{Q}}}(\mathbf{V}_{\hat{\mathbf{Q}}}^T\mathbf{x})\| = \|\hat{\mathbf{Q}}\mathbf{x}\|.$$

Summing it over all $\mathbf{x}\in\mathcal{X}$, we conclude the contradiction $F(\hat{\mathbf{Q}}^+) < F(\hat{\mathbf{Q}})$.

■

In order to prove Theorem 6 we first notice that by definition and the connection of γ_0, γ_0 with second derivative of $F(\mathbf{Q})$

$$F_{\mathcal{X}}(\tilde{\mathbf{Q}}) - F_{\mathcal{X}}(\tilde{\mathbf{Q}}) \geq N\gamma_0 \|\tilde{\mathbf{Q}} - \hat{\mathbf{Q}}\|_F^2, \tag{78}$$

and

$$F_{\mathcal{X}}(\tilde{\mathbf{Q}}) - F_{\mathcal{X}}(\tilde{\mathbf{Q}}) \geq N\gamma_0' \|\tilde{\mathbf{Q}} - \hat{\mathbf{Q}}\|^2. \tag{79}$$

Next, we observe that

$$|F_{\mathcal{X}}(\hat{\mathbf{Q}}) - F_{\tilde{\mathcal{X}}}(\hat{\mathbf{Q}})| \leq \sum_{i=1}^{N} \left| \|\hat{\mathbf{Q}}\tilde{\mathbf{x}}_i\| - \|\hat{\mathbf{Q}}\mathbf{x}_i\| \right| \leq \sum_{i=1}^{N} \|\hat{\mathbf{Q}}(\tilde{\mathbf{x}}_i - \mathbf{x}_i)\| \leq \sum_{i=1}^{N} \|\tilde{\mathbf{x}}_i - \mathbf{x}_i\| \leq \sum_{i=1}^{N} \epsilon_i$$

and similarly $|F_{\mathcal{X}}(\tilde{\mathbf{Q}}) - F_{\tilde{\mathcal{X}}}(\tilde{\mathbf{Q}})| \leq \sum_{i=1}^{N} \epsilon_i$. Therefore,

$$\begin{aligned} F_{\mathcal{X}}(\tilde{\mathbf{Q}}) - F_{\mathcal{X}}(\hat{\mathbf{Q}}) &= (F_{\tilde{\mathcal{X}}}(\tilde{\mathbf{Q}}) - F_{\tilde{\mathcal{X}}}(\hat{\mathbf{Q}})) + (F_{\mathcal{X}}(\tilde{\mathbf{Q}}) - F_{\tilde{\mathcal{X}}}(\tilde{\mathbf{Q}})) + (F_{\tilde{\mathcal{X}}}(\hat{\mathbf{Q}}) \\ -F_{\mathcal{X}}(\hat{\mathbf{Q}})) &\leq 0 + |F_{\mathcal{X}}(\tilde{\mathbf{Q}}) - F_{\tilde{\mathcal{X}}}(\tilde{\mathbf{Q}})| + |F_{\tilde{\mathcal{X}}}(\hat{\mathbf{Q}}) - F_{\mathcal{X}}(\hat{\mathbf{Q}})| \leq 2\sum_{i=1}^{N} \epsilon_i. \end{aligned} \tag{80}$$

Therefore (23) follows from (78), (79) and (80). Applying the Davis-Kahan perturbation Theorem (Davis and Kahan, 1970) to (23), we conclude (24).

7.7.1 Implication of Theorem 6 to Dimension Estimation

Theorem 6 implies that we may properly estimate the dimension of the underlying subspace for low-dimensional data with sufficiently small perturbation. We make this statement more precise by assuming the setting of Theorem 6 and further assuming that $\hat{\mathbf{Q}}$ is a low-rank matrix with $\ker(\hat{\mathbf{Q}}) = \mathrm{L}^*$. We note that the $(D-d+1)$st eigenvalue of $\hat{\mathbf{Q}}$ is 0. Thus applying the following eigenvalue stability inequality (Tao, 2012, (1.63)):

$$|\lambda_i(\mathbf{A}+\mathbf{B}) - \lambda_i(\mathbf{A})| \leq \|\mathbf{B}\|, \tag{81}$$

we obtain that the $(D-d+1)$st eigenvalue of $\tilde{\mathbf{Q}}$ is smaller than $\sqrt{2\sum_{i=1}^{N} \epsilon_i/\gamma_0}$, and the $(D-d)$th eigengap of $\tilde{\mathbf{Q}}$ is larger than $\nu_{D-d} - 2\sqrt{2\sum_{i=1}^{N} \epsilon_i/\gamma_0}$ (recall that ν_{D-d} is the $(D-d)$th eigengap of $\hat{\mathbf{Q}}$). This means that when the noise is small and the conditions of Theorem 1 hold, then we can estimate the dimension of the underlying subspace for $\tilde{\mathcal{X}}$ from the number of small eigenvalues.

7.7.2 Improved Bounds in a Restricted Setting

We assume that $\epsilon_i = O(\epsilon)$ for all $1 \leq i \leq N$, where ϵ is sufficiently small, and further assume that $\mathrm{rank}(\hat{\mathbf{Q}}) = D$. We show that in this special case the norm of $\hat{\mathbf{Q}} - \tilde{\mathbf{Q}}$ is of order $O(\epsilon)$ instead of order $O(\sqrt{\epsilon})$ that is specified in Theorem 6.

We note that since $\hat{\mathbf{Q}}$ is of full rank, then the first and second directional derivative of F are well-defined in a sufficiently small neighborhood around $\hat{\mathbf{Q}}$. Therefore, if $\mathbf{\Delta} \in \mathbb{R}^{D\times D}$ and $\|\mathbf{\Delta}\|$ is sufficiently small then

$$F'_{\mathcal{X}}(\hat{\mathbf{Q}}) - F'_{\mathcal{X}}(\hat{\mathbf{Q}} + \mathbf{\Delta}) = O(\|\mathbf{\Delta}\|). \tag{82}$$

Furthermore, we note by basic calculations that

$$F'_{\mathcal{X}}(\mathbf{Q}) - F'_{\tilde{\mathcal{X}}}(\mathbf{Q}) = O(\epsilon). \tag{83}$$

Combining (83) with the following facts: $F'_{\mathcal{X}}(\hat{\mathbf{Q}}) = 0$ and $F'_{\tilde{\mathcal{X}}}(\tilde{\mathbf{Q}}) = 0$, we obtain that

$$F'_{\mathcal{X}}(\hat{\mathbf{Q}}) - F'_{\mathcal{X}}(\tilde{\mathbf{Q}}) = F'_{\tilde{\mathcal{X}}}(\tilde{\mathbf{Q}}) - F'_{\mathcal{X}}(\tilde{\mathbf{Q}}) = O(\epsilon). \tag{84}$$

At last, the combination of (82) and (84) implies that $\|\hat{\mathbf{Q}} - \tilde{\mathbf{Q}}\| = O(\epsilon)$. Clearly, the spectral norm of $\hat{\mathbf{Q}} - \tilde{\mathbf{Q}}$ can be replaced with any other norm, in particular, the Frobenius norm.

7.8 Proof of Proposition 7

We recall the function F_I, which was defined in (19), and the notation ${F_I}_{,1}''(\mathbf{Q}, \mathbf{\Delta})$ should be clear, where now F_I replaces F.

The law of large numbers implies that ${F_1}''(\mathbf{Q}, \mathbf{\Delta})/N \to {F_I}''(\mathbf{Q}, \mathbf{\Delta})$ almost surely for any $\mathbf{\Delta}$ and $\mathbf{Q}$ (see also related bounds in Coudron and Lerman 2012). Since $\mathbf{Q}$ and $\mathbf{\Delta}$ lie in compact space, we conclude (26) for γ_0 and c_0; the proof is identical for γ'_0 and c'_0.

7.9 Proof of Theorem 8

The theorem follows from the observation that $0 \le F(\mathbf{Q}) - F_\delta(\mathbf{Q}) \le N\delta/2$ for all $\mathbf{Q} \in \mathbb{H}$ and the proof of Theorem 6.

7.10 Proof of Theorem 9

It is sufficient to verify that

$$\text{If } \tilde{\mathbf{A}} \in \mathbb{R}^{D\times D} \text{ with } \mathrm{Im}(\tilde{\mathbf{A}}) = \mathrm{L}^*, \text{ then } L(\tilde{\mathbf{A}} + \eta\mathbf{I}) \to \infty \text{ as } \eta \to 0. \tag{85}$$

Indeed, since $L(\mathbf{A})$ is a continuous function, (85) implies that $L(\tilde{\mathbf{A}})$ is undefined (or infinite) and therefore $\tilde{\mathbf{A}}$ is not the minimizer of (28) as stated in Theorem 9.

We fix $a_1 < \lim_{x\to\infty} xu(x)$ and note that Condition D_0 (w.r.t. L^*) implies that

$$|\mathcal{X}_0|/N > (D-d)/a_1. \tag{86}$$

Condition M implies that there exists x_1 such that for any $x > x_1$: $xu(x) \ge a_1$ and therefore (recalling that $u = \rho'$) $\rho(x) \ge a_1 \ln(x - x_1)/2 + u(x_1)/2$. Thus for any $\mathbf{x}_i \in \mathcal{X}_0$, we have

$$\rho(\mathbf{x}_i^T(\tilde{\mathbf{A}} + \eta\mathbf{I})^{-1}\mathbf{x}_i) \ge a_1 \ln(1/\eta - x_1)/2 + C_i \text{ for some constant } C_i \equiv C_i(\mathbf{x}_i, \tilde{\mathbf{A}}) \tag{87}$$

and

$$\frac{N}{2}\log(\det(\mathbf{A})) \le NC_0 + (D-d)/2\ln(\eta) \text{ for some } C_0 \equiv C_0(\tilde{\mathbf{A}}). \tag{88}$$

Equation (85) thus follows from (86)-(88) and the theorem is concluded.

7.11 Proof of Theorem 10

The derivative of the energy function in the RHS of (32) is $\mathbf{Q}\mathbf{X}^T\mathbf{X} + \mathbf{X}^T\mathbf{X}\mathbf{Q}$. Using the argument establishing (36) and the fact that $\hat{\mathbf{Q}}_2$ is the minimizer of (32), we conclude that $\mathbf{Q}\mathbf{X}^T\mathbf{X} + \mathbf{X}^T\mathbf{X}\mathbf{Q}$ is a scalar matrix. We then conclude (33) by using the argument establishing (37) as well as the following two facts: $\mathrm{tr}(\hat{\mathbf{Q}}_2) = 1$ and $\mathbf{X}$ is full rank (so the inverse of $\mathbf{X}^T\mathbf{X}$ exists).

7.12 Proof of Theorem 11

We frequently use here some of the notation introduced in §4.1, in particular, $I(\mathbf{Q})$, $\mathrm{L}(\mathbf{Q})$ and $T(\mathbf{Q})$. We will first prove that $F(\mathbf{Q}_k) \geq F(\mathbf{Q}_{k+1})$ for all $k \geq 1$. For this purpose, we use the convex quadratic function:

$$G(\mathbf{Q}, \mathbf{Q}^*) = \frac{1}{2} \sum_{\substack{i=1 \\ i \notin I(\mathbf{Q}^*)}}^{N} \left(\|\mathbf{Q}\mathbf{x}_i\|^2 / \|\mathbf{Q}^*\mathbf{x}_i\| + \|\mathbf{Q}^*\mathbf{x}_i\| \right).$$

Following the same derivation of (44) and (36), we obtain that

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{Q}} G(\mathbf{Q}, \mathbf{Q}_k)\big|_{\mathbf{Q}=\mathbf{Q}_{k+1}} = \left(\mathbf{Q}_{k+1} \left(\sum_{\substack{i=1 \\ i \notin I(\mathbf{Q}_k)}}^{N} \frac{\mathbf{x}_i \mathbf{x}_i^T}{\|\mathbf{Q}_k \mathbf{x}_i\|} \right) + \left(\sum_{\substack{i=1 \\ i \notin I(\mathbf{Q}_k)}}^{N} \frac{\mathbf{x}_i \mathbf{x}_i^T}{\|\mathbf{Q}_k \mathbf{x}_i\|} \right) \mathbf{Q}_{k+1} \right) /2.$$

We let $\mathbf{A}_k = \sum_{i=1, i \notin I(\mathbf{Q}_k)}^{N} \frac{\mathbf{x}_i \mathbf{x}_i^T}{\|\mathbf{Q}_k \mathbf{x}_i\|}$, $c_k = \mathbf{P}_{\mathrm{L}(\mathbf{Q}_k)^\perp} \mathbf{A}_k^{-1} \mathbf{P}_{\mathrm{L}(\mathbf{Q}_k)^\perp}$ and for any symmetric $\mathbf{\Delta} \in \mathbb{R}^{D \times D}$ with $\mathrm{tr}(\mathbf{\Delta}) = 0$ and $\mathbf{P}_{\mathrm{L}(\mathbf{Q}_k)} \mathbf{\Delta} = \mathbf{0}$ we let $\mathbf{\Delta}_0 = \tilde{\mathbf{P}}^T_{\mathrm{L}(\mathbf{Q}_k)^\perp} \mathbf{\Delta} \tilde{\mathbf{P}}_{\mathrm{L}(\mathbf{Q}_k)^\perp}$. We note that

$$\begin{aligned} \mathrm{tr}(\mathbf{\Delta}_0) &= \langle \mathbf{\Delta}_0, \mathbf{I} \rangle_F = \left\langle \tilde{\mathbf{P}}^T_{\mathrm{L}(\mathbf{Q}_k)^\perp} \mathbf{\Delta} \tilde{\mathbf{P}}_{\mathrm{L}(\mathbf{Q}_k)^\perp}, \mathbf{I} \right\rangle_F = \left\langle \mathbf{\Delta}, \tilde{\mathbf{P}}_{\mathrm{L}(\mathbf{Q}_k)^\perp} \tilde{\mathbf{P}}^T_{\mathrm{L}(\mathbf{Q}_k)^\perp} \right\rangle_F \\ &= \left\langle \mathbf{\Delta}, \mathbf{I} - \mathbf{P}_{\mathrm{L}(\mathbf{Q}_k)} \right\rangle_F = \langle \mathbf{\Delta}, \mathbf{I} \rangle_F - \left\langle \mathbf{\Delta}, \mathbf{P}_{\mathrm{L}(\mathbf{Q}_k)} \right\rangle_F = \langle \mathbf{\Delta}, \mathbf{I} \rangle_F = \mathrm{tr}(\mathbf{\Delta}) = 0. \end{aligned}$$

Consequently, we establish that the derivative of $G(\mathbf{Q}, \mathbf{Q}_k)$ at $\mathbf{Q}_{k+1}$ in the direction $\mathbf{\Delta}$ is zero as follows.

$$\begin{aligned} &\langle (\mathbf{Q}_{k+1}\mathbf{A}_k + \mathbf{A}_k\mathbf{Q}_{k+1})/2, \mathbf{\Delta} \rangle_F = \langle \mathbf{Q}_{k+1}\mathbf{A}_k, \mathbf{\Delta} \rangle_F = c_k \left\langle \mathbf{P}_{\mathrm{L}(\mathbf{Q}_k)^\perp} \mathbf{A}_k^{-1} \mathbf{P}_{\mathrm{L}(\mathbf{Q}_k)^\perp} \mathbf{A}_k, \mathbf{\Delta} \right\rangle_F \\ =& c_k \left\langle \mathbf{P}_{\mathrm{L}(\mathbf{Q}_k)^\perp} \mathbf{A}_k^{-1} \mathbf{P}_{\mathrm{L}(\mathbf{Q}_k)^\perp} \mathbf{A}_k, \tilde{\mathbf{P}}_{\mathrm{L}(\mathbf{Q}_k)^\perp} \mathbf{\Delta}_0 \tilde{\mathbf{P}}^T_{\mathrm{L}(\mathbf{Q}_k)^\perp} \right\rangle_F \\ =& c_k \left\langle (\tilde{\mathbf{P}}^T_{\mathrm{L}(\mathbf{Q}_k)^\perp} \mathbf{A}_k^{-1} \tilde{\mathbf{P}}_{\mathrm{L}(\mathbf{Q}_k)^\perp})(\tilde{\mathbf{P}}^T_{\mathrm{L}(\mathbf{Q}_k)^\perp} \mathbf{A}_k \tilde{\mathbf{P}}_{\mathrm{L}(\mathbf{Q}_k)^\perp}), \mathbf{\Delta}_0 \right\rangle_F = c_k \langle \mathbf{I}, \mathbf{\Delta}_0 \rangle_F = 0. \end{aligned}$$

This and the strict convexity of $G(\mathbf{Q}, \mathbf{Q}_k)$ (which follows from $\mathrm{Sp}(\{\mathbf{x}_i\}_{i \notin I(\mathbf{Q}_k)}) = \mathbb{R}^D$ using (14)) imply that $\mathbf{Q}_{k+1}$ is the unique minimizer of $G(\mathbf{Q}, \mathbf{Q}_k)$ among all $\mathbf{Q} \in \mathbb{H}$ such that $\mathbf{P}_{\mathrm{L}(\mathbf{Q}_k)}\mathbf{Q} = \mathbf{0}$.

Combining this with the following two facts: $\mathbf{Q}_{k+1}\mathbf{x}_i = 0$ for any $i \in I(\mathbf{Q}_k)$ and $G(\mathbf{Q}_k, \mathbf{Q}_k) = F(\mathbf{Q}_k)$, we conclude that

$$F(\mathbf{Q}_{k+1}) = \sum_{i \notin I(\mathbf{Q}_k)} \|\mathbf{Q}_{k+1}\mathbf{x}_i\| = \sum_{i \notin I(\mathbf{Q}_k)} \frac{\|\mathbf{Q}_{k+1}\mathbf{x}_i\|\|\mathbf{Q}_k\mathbf{x}_i\|}{\|\mathbf{Q}_k\mathbf{x}_i\|}$$
$$\leq \sum_{i \notin I(\mathbf{Q}_k)} \frac{\|\mathbf{Q}_{k+1}\mathbf{x}_i\|^2 + \|\mathbf{Q}_k\mathbf{x}_i\|^2}{2\|\mathbf{Q}_k\mathbf{x}_i\|} = G(\mathbf{Q}_{k+1}, \mathbf{Q}_k) \leq G(\mathbf{Q}_k, \mathbf{Q}_k) = F(\mathbf{Q}_k). \tag{89}$$

Since F is positive, $F(\mathbf{Q}_k)$ converges and

$$F(\mathbf{Q}_k) - F(\mathbf{Q}_{k+1}) \to 0 \text{ as } k \to \infty. \tag{90}$$

Applying (89) we also have that

$$F(\mathbf{Q}_k) - F(\mathbf{Q}_{k+1}) \geq G(\mathbf{Q}_k, \mathbf{Q}_k) - G(\mathbf{Q}_{k+1}, \mathbf{Q}_k) = \frac{1}{2} \sum_{i \notin I(\mathbf{Q}_k)} \|(\mathbf{Q}_k - \mathbf{Q}_{k+1})\mathbf{x}_i\|^2 / \|\mathbf{Q}_k\mathbf{x}_i\|. \tag{91}$$

We note that if $\mathbf{Q}_k \neq \mathbf{Q}_{k+1}$, then $\mathrm{Sp}(\{\mathbf{x}_i\}_{i \notin I(\mathbf{Q}_k)}) = \mathbb{R}^D \supset \ker(\mathbf{Q}_k - \mathbf{Q}_{k+1})$ and $1/\|\mathbf{Q}_k\mathbf{x}_i\| \geq 1/\max_i \|\mathbf{x}_i\|$. Combining this observation with (90) and (91) we obtain that

$$\|\mathbf{Q}_k - \mathbf{Q}_{k+1}\|_2 \to 0 \text{ as } k \to \infty. \tag{92}$$

Since for all $k \in \mathbb{N}$, $\mathbf{Q}_k$ is nonnegative (this follows from its defining formula (39)) and $\mathrm{tr}(\mathbf{Q}_k) = 1$, the sequence $\{\mathbf{Q}_k\}_{k\in\mathbb{N}}$ lies in a compact space (of nonnegative matrices) and it thus has a converging subsequence. Assume a subsequence of $\{\mathbf{Q}_k\}_{k\in\mathbb{N}}$, which converges to $\tilde{\mathbf{Q}}$. We claim the following property of $\tilde{\mathbf{Q}}$:

$$\tilde{\mathbf{Q}} = \operatorname*{arg\,min}_{\mathbf{Q} \in \mathbb{H}_0} F(\mathbf{Q}), \text{ where } \mathbb{H}_0 := \{\mathbf{Q} \in \mathbb{H} : \ker \mathbf{Q} \supseteq \mathrm{L}(\tilde{\mathbf{Q}})\}. \tag{93}$$

In order to prove (93), we note that (89) and the convergence of the subsequence imply that $F(\tilde{\mathbf{Q}}) = F(T(\tilde{\mathbf{Q}}))$. Combining this with (89) (though replacing $\mathbf{Q}_k$ and $\mathbf{Q}_{k+1}$ in (89) with $\tilde{\mathbf{Q}}$ and $T(\tilde{\mathbf{Q}})$ respectively) we get that $G(T(\tilde{\mathbf{Q}}), \tilde{\mathbf{Q}}) = G(\tilde{\mathbf{Q}}, \tilde{\mathbf{Q}})$. We conclude that $T(\tilde{\mathbf{Q}}) = \tilde{\mathbf{Q}}$ from this observation and the following three facts: 1) $\mathbf{Q} = \tilde{\mathbf{Q}}$ is the unique minimizer of $G(\mathbf{Q}, \tilde{\mathbf{Q}})$ among all $\mathbf{Q} \in \mathbb{H}$, 2) $\mathbf{P}_{\mathrm{L}(\tilde{\mathbf{Q}})}\tilde{\mathbf{Q}} = \mathbf{0}$, 3) $\mathbf{Q} = T(\tilde{\mathbf{Q}})$ is the unique minimizer of $G(\mathbf{Q}, \tilde{\mathbf{Q}})$ among all $\mathbf{Q} \in \mathbb{H}$ such that $\mathbf{P}_{\mathrm{L}(\tilde{\mathbf{Q}})}\mathbf{Q} = \mathbf{0}$ (we remark that $F(\mathbf{Q})$ is strictly convex in $\mathbb{H}$ and consequently also in $\mathbb{H}_0$ by Theorem 2). Therefore, for any symmetric $\mathbf{\Delta} \in \mathbb{R}^{D\times D}$ with $\mathrm{tr}(\mathbf{\Delta}) = 0$ and $\mathbf{P}_{\mathrm{L}(\tilde{\mathbf{Q}})}\mathbf{\Delta} = \mathbf{0}$, the directional derivative at $\tilde{\mathbf{Q}}$ is 0:

$$0 = \left\langle \mathbf{\Delta}, \frac{\mathrm{d}}{\mathrm{d}\mathbf{Q}} G(\mathbf{Q}, \tilde{\mathbf{Q}})\big|_{\mathbf{Q}=\tilde{\mathbf{Q}}} \right\rangle_F = \left\langle \mathbf{\Delta}, \tilde{\mathbf{Q}} \sum_{i \notin I(\tilde{\mathbf{Q}})} \frac{\mathbf{x}_i\mathbf{x}_i^T}{\|\tilde{\mathbf{Q}}\mathbf{x}_i\|} \right\rangle_F. \tag{94}$$

We note that (94) is the corresponding directional derivative of $F(\mathbf{Q})$ when restricted to $\mathbf{Q} \in \mathbb{H}_0$ and we thus conclude (93).

Next, we will prove that $\{\mathbf{Q}_k\}_{k\in\mathbb{N}}$ converge to $\tilde{\mathbf{Q}}$ by proving that there are only finite choices for $\tilde{\mathbf{Q}}$. In view of (93) and the strict convexity of $F(\mathbf{Q})$ in $\mathbb{H}_0$, any limit $\tilde{\mathbf{Q}}$ (of

a subsequence as above) is uniquely determined by $I(\tilde{\mathbf{Q}})$. Since the number of choices for $I(\tilde{\mathbf{Q}})$ is finite (independently of $\tilde{\mathbf{Q}}$), the number of choices for $\tilde{\mathbf{Q}}$ is finite. That is, $\mathcal{Y} := \{\mathbf{Q} \in \mathbb{H} : F(\mathbf{Q}) = F(T(\mathbf{Q}))\}$ is a finite set. Combining this with (92) and the convergence analysis of the sequence $\{\mathbf{Q}_k\}_{k\in\mathbb{N}}$ (see Ostrowski, 1966, Theorem 28.1), we conclude that $\{\mathbf{Q}_k\}_{k\in\mathbb{N}}$ converges to $\tilde{\mathbf{Q}}$.

At last, we assume that $\tilde{\mathbf{Q}}\mathbf{x}_i \neq \mathbf{0}$ for all $1 \leq i \leq N$. We note that $I(\tilde{\mathbf{Q}}) = \emptyset$ and thus $\tilde{\mathbf{Q}} = \hat{\mathbf{Q}}$ by (93). The proof for the rate of convergence follows the analysis of generalized Weiszfeld's method by Chan and Mulet (1999) (in particular see §6 of that work). We practically need to verify Hypotheses 4.1 and 4.2 (see §4 of that work) and replace the functions F and G in that work by $F(\mathbf{Q})$ and

$$\tilde{G}(\mathbf{Q}, \mathbf{Q}^*) = \sum_{i=1}^{N} \left(\|\mathbf{Q}\mathbf{x}_i\|^2 / \|\mathbf{Q}^*\mathbf{x}_i\| + \|\mathbf{Q}^*\mathbf{x}_i\| \right)$$

respectively. We note that the functions $\tilde{G}$ and G (defined earlier in this work) coincide in the following way: $\tilde{G}(\mathbf{Q}, \mathbf{Q}_k) = G(\mathbf{Q}, \mathbf{Q}_k)$ for any $k \in \mathbb{N}$ (this follows from the fact that $\mathbf{Q}_k\mathbf{x}_i \neq \mathbf{0}$ for all $k \in \mathbb{N}$ and $1 \leq i \leq N$; indeed, otherwise for some i, $\mathbf{Q}_j\mathbf{x}_i = \mathbf{0}$ for $j \geq k$ by (39) and this leads to the contradiction $\hat{\mathbf{Q}}\mathbf{x}_i = \mathbf{0}$). We remark that even though Chan and Mulet (1999) consider vector-valued functions, their proof generalizes to matrix-valued functions as here. Furthermore, we can replace the global properties of Hypotheses 4.1 and 4.2 of Chan and Mulet (1999) by the local properties in $B(\hat{\mathbf{Q}}, \delta_0)$ for any $\delta_0 > 0$, since the convergence of $\mathbf{Q}_k$ implies the existence of $K_0 > 0$ such that $\mathbf{Q}_k \in B(\hat{\mathbf{Q}}, \delta_0)$ for all $k > K_0$. In particular, there is no need to check condition 2 in Hypothesis 4.1. Condition 1 in Hypothesis 4.1 holds since $F(\mathbf{Q})$ is twice differentiable in $B(\hat{\mathbf{Q}}, \delta_0)$ (which follows from the assumption on the limit $\tilde{\mathbf{Q}} \equiv \hat{\mathbf{Q}}$ and the continuity of the derivative). Conditions 1-3 in Hypothesis 4.2 are verified by the fact that C of Hypothesis 4.2 satisfies $C(\mathbf{Q}^*) = \sum_{i=1}^{N} \mathbf{x}_i\mathbf{x}_i^T / \|\mathbf{Q}^*\mathbf{x}_i\|$ and $\mathbf{Q}^*\mathbf{x}_i \neq \mathbf{0}$ when $\mathbf{Q}^* \in B(\hat{\mathbf{Q}}, \delta_0)$. Condition 3 in Hypothesis 3.1 and condition 4 in Hypothesis 4.2 are easy to check.

7.13 Proof of Theorem 12

The proof follows from the second part of the proof of Theorem 11, while using instead of $\tilde{G}(\mathbf{Q}, \mathbf{Q}^*)$ the function

$$G_\delta(\mathbf{Q}, \mathbf{Q}^*) = \frac{1}{2} \sum_{i=1, \|\mathbf{Q}^*\mathbf{x}_i\| \geq \delta}^{N} \left(\|\mathbf{Q}\mathbf{x}_i\|^2 / \|\mathbf{Q}^*\mathbf{x}_i\| + \|\mathbf{Q}^*\mathbf{x}_i\| \right) + \sum_{i=1, \|\mathbf{Q}^*\mathbf{x}_i\| < \delta}^{N} (\|\mathbf{Q}\mathbf{x}_i\|^2 / 2\delta + \delta/2).$$

7.14 Proof of Theorem 13

We note that the minimization of $F(\mathbf{Q})$ over all $\mathbf{Q} \in \mathbb{H}$ such that $\mathbf{Q}\mathbf{P}_{\hat{\mathrm{L}}^\perp} = \mathbf{0}$ in Algorithm 3 can be performed at each iteration with respect to the projected data: $\tilde{\mathbf{P}}_{\hat{\mathrm{L}}}(\mathcal{X}) = \{\tilde{\mathbf{P}}_{\hat{\mathrm{L}}}\mathbf{x}_1, \tilde{\mathbf{P}}_{\hat{\mathrm{L}}}\mathbf{x}_2, \cdots, \tilde{\mathbf{P}}_{\hat{\mathrm{L}}}\mathbf{x}_N\}$.

We note that conditions (6) and (7) hold for $\tilde{\mathbf{P}}_{\hat{\mathrm{L}}}(\mathcal{X})$ with any $\hat{\mathrm{L}} \supseteq \mathrm{L}^*$. Therefore, Theorem 1 implies that $\mathbf{u} \perp \mathrm{L}^*$ and $\hat{\mathrm{L}} \supseteq \mathrm{L}^*$ in each iteration. Since $\dim(\hat{\mathrm{L}})$ decreases by one in each iteration, $\dim(\hat{\mathrm{L}}) = d$ in $D - d$ iterations and thus $\hat{\mathrm{L}} = \mathrm{L}^*$.

8. Conclusion

We proposed an M-estimator for the problems of exact and near subspace recovery. Substantial theory has been developed to quantify the recovery obtained by this estimator as well as its numerical approximation. Numerical experiments demonstrated state-of-the-art speed and accuracy for our corresponding implementation on both synthetic and real data sets.

This work broadens the perspective of two recent ground-breaking theoretical works for subspace recovery by Candès et al. (2011) and Xu et al. (2012). We hope that it will motivate additional approaches to this problem.

There are many interesting open problems that stem from our work. We believe that by modifying or extending the framework described in here, one can even yield better results in various scenarios. For example, we have discussed in §1.2 the modification by Lerman et al. (2012) suggesting tighter convex relaxation of orthogonal projectors when d is known. We also discussed in §1.2 adaptation by Wang and Singer (2013) of the basic ideas in here to the different synchronization problem. Another direction was recently followed up by Coudron and Lerman (2012), where they established exact asymptotic subspace recovery under specific sampling assumptions, which may allow relatively large magnitude of noise. It is interesting to follow this direction and establish exact recovery when using in theory a sequence of IRLS regularization parameters $\{\delta_i\}_{i\in\mathbb{N}}$ approaching zero (in analogy to the work of Daubechies et al. 2010).

An interesting generalization that was not pursued so far is robust data modeling by multiple subspaces or by locally-linear structures. It is also interesting to know whether one can adapt the current framework so that it can detect linear structure in the presence of both sparse elementwise corruption (as in Candès et al. 2011) and the type of outliers addressed in here.

Acknowledgments

This work was supported by NSF grants DMS-09-15064 and DMS-09-56072, GL was also partially supported by the IMA (during 2010-2012). Arthur Szlam has inspired our extended research on robust l_1-type subspace recovery. We thank John Wright for referring us to Xu et al. (2010b) shortly after it appeared online and for some guidance with the real data sets. GL thanks Emmanuel Candès for inviting him to visit Stanford university in May 2010 and for his constructive criticism on the lack of a theoretically guaranteed algorithm for the l_1 subspace recovery of Lerman and Zhang (2010).

Supp. webpage: `http://www.math.umn.edu/~lerman/gms`.

References

A. Agarwal, S. Negahban, and M. J. Wainwright. Noisy matrix decomposition via convex relaxation: optimal rates in high dimensions. *Ann. Statist.*, 40(2):1171–1197, 2012a. ISSN 0090-5364. doi: 10.1214/12-AOS1000.

A. Agarwal, S. Negahban, and M. J. Wainwright. Fast global convergence of gradient methods for high-dimensional statistical recovery. *The Annals of Statistics*, 40(5):2452–2482, 2012b.

L. P. Ammann. Robust singular value decompositions: A new approach to projection pursuit. *Journal of the American Statistical Association*, 88(422):pp. 505–514, 1993. ISSN 01621459.

E. Arias-Castro, D. L. Donoho, X. Huo, and C. A. Tovey. Connect the dots: how many random points can a regular curve pass through? *Adv. in Appl. Probab.*, 37(3):571–603, 2005.

E. Arias-Castro, G. Chen, and G. Lerman. Spectral clustering based on local linear approximations. *Electron. J. Statist.*, 5:1537–1587, 2011.

O. Arslan. Convergence behavior of an iterative reweighting algorithm to compute multivariate M-estimates for location and scatter. *Journal of Statistical Planning and Inference*, 118(1-2):115 – 128, 2004. ISSN 0378-3758. doi: 10.1016/S0378-3758(02)00402-0.

A. Bargiela and J. K. Hartley. Orthogonal linear regression algorithm based on augmented matrix formulation. *Comput. Oper. Res.*, 20:829–836, October 1993. ISSN 0305-0548. doi: 10.1016/0305-0548(93)90104-Q.

R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):218–233, February 2003.

R. Basri, T. Hassner, and L. Zelnik-Manor. Approximate nearest subspace search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):266–278, 2011. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2010.110.

R. Bhatia and D. Drissi. Generalized Lyapunov equations and positive definite functions. *SIAM J. Matrix Anal. Appl.*, 27(1):103–114, May 2005. ISSN 0895-4798. doi: 10.1137/040608970.

P. Bradley and O. Mangasarian. k-plane clustering. *J. Global optim.*, 16(1):23–32, 2000.

S. C. Brubaker. Robust PCA and clustering in noisy mixtures. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 1078–1087, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

E. J. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006. doi: 10.1002/cpa.20124.

E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

T. F. Chan and P. Mulet. On the convergence of the lagged diffusivity fixed point method in total variation image restoration. *SIAM J. Numer. Anal.*, 36:354–367, 1999. ISSN 0036-1429. doi: 10.1137/S0036142997327075.

V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky. Rank-sparsity incoherence for matrix decomposition. *SIAM J. Optim.*, 21(2):572–596, 2011. ISSN 1052-6234. doi: 10.1137/090761793.

T.-J. Chin, J. Yu, and D. Suter. Accelerated hypothesis generation for multistructure data via preference analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(4):625–638, April 2012. ISSN 0162-8828. doi: 10.1109/TPAMI.2011.169.

A. K. Cline. Rate of convergence of Lawson's algorithm. *Mathematics of Computation*, 26 (117):pp. 167–176, 1972. ISSN 00255718.

M. Coudron and G. Lerman. On the sample complexity of robust PCA. In *NIPS*, pages 3230–3238, 2012.

C. Croux and G. Haesbroeck. Principal component analysis based on robust estimators of the covariance or correlation matrix: Influence functions and efficiencies. *Biometrika*, 87: 603–618, 2000.

C. Croux, P. Filzmoser, and M. Oliveira. Algorithms for projectionc pursuit robust principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 87(2):218–225, 2007.

A. d'Aspremont, L. El Ghaoui, M. Jordan, and G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448, 2007. doi: 10.1137/050645506.

I. Daubechies, R. DeVore, M. Fornasier, and C. S. Gunturk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63:1–38, 2010. doi: 10.1002/cpa.20303.

G. David and S. Semmes. Singular integrals and rectifiable sets in $\mathbb{R}^n$: au-delà des graphes Lipschitziens. *Astérisque*, 193:1–145, 1991.

P. L. Davies. Asymptotic behaviour of s-estimates of multivariate location parameters and dispersion matrices. *The Annals of Statistics*, 15(3):pp. 1269–1292, 1987. ISSN 00905364.

C. Davis and W. M. Kahan. The rotation of eigenvectors by a perturbation. iii. *SIAM J. on Numerical Analysis*, 7:1–46, 1970.

S. J. Devlin, R. Gnandesikan, and J. R. Kettenring. Robust estimation of dispersion matrices and principal components. *Journal of the American Statistical Association*, 76(374):pp. 354–362, 1981. ISSN 01621459.

C. Ding, D. Zhou, X. He, and H. Zha. R1-PCA: rotational invariant L_1-norm principal component analysis for robust subspace factorization. In *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, pages 281–288, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143880.

M. Fornasier, H. Rauhut, and R. Ward. Low-rank matrix recovery via iteratively reweighted least squares minimization. *SIAM J. Optim.*, 21(4):1614–1640, 2011. ISSN 1052-6234. doi: 10.1137/100811404.

M. Hardt and A. Moitra. Algorithms and hardness for robust subspace recovery. In *COLT*, pages 354–375, 2013.

J. Ho, M. Yang, J. Lim, K. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. In *Proceedings of International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 11–18, 2003.

D. Hsu, S.M. Kakade, and Tong Zhang. Robust matrix decomposition with sparse corruptions. *Information Theory, IEEE Transactions on*, 57(11):7221 –7234, nov. 2011. ISSN 0018-9448. doi: 10.1109/TIT.2011.2158250.

P. J. Huber and E. M. Ronchetti. *Robust Statistics*. Wiley Series in Probability and Statistics. Wiley, Hoboken, NJ, 2nd edition, 2009. ISBN 978-0-470-12990-6. doi: 10.1002/9780470434697.

Q. Ke and T. Kanade. Robust subspace computation using L_1 norm. Technical report, Carnegie Mellon, 2003.

J. T. Kent and D. E. Tyler. Redescending M-estimates of multivariate location and scatter. *The Annals of Statistics*, 19(4):pp. 2102–2119, 1991. ISSN 00905364.

H. W. Kuhn. A note on Fermat's problem. *Mathematical Programming*, 4:98–107, 1973. ISSN 0025-5610. 10.1007/BF01584648.

N. Kwak. Principal component analysis based on L_1-norm maximization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(9):1672–1680, 2008. doi: 10.1109/TPAMI.2008.114.

C. L. Lawson. *Contributions to the Theory of Linear Least Maximum Approximation*. PhD thesis, University of California, Los Angeles, 1961.

K.-C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27 (5):684–698, 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.92.

G. Lerman and T. Zhang. l_p-Recovery of the most significant subspace among multiple subspaces with outliers. *ArXiv e-prints*, December 2010. To Appear in *Constructive Approximation*.

G. Lerman and T. Zhang. Robust recovery of multiple subspaces by geometric l_p minimization. *Ann. Statist.*, 39(5):2686–2715, 2011. ISSN 0090-5364. doi: 10.1214/11-AOS914.

G. Lerman, M. McCoy, J. A. Tropp, and T. Zhang. Robust computation of linear models, or how to find a needle in a haystack. *ArXiv e-prints*, February 2012.

G. Li and Z. Chen. Projection-pursuit approach to robust dispersion matrices and principal components: Primary theory and monte carlo. *Journal of the American Statistical Association*, 80(391):759–766, 1985. ISSN 01621459. doi: 10.2307/2288497.

L. Li, W. Huang, I. Gu, and Q. Tian. Statistical modeling of complex backgrounds for foreground object detection. *Image Processing, IEEE Transactions on*, 13(11):1459 – 1472, nov. 2004. ISSN 1057-7149. doi: 10.1109/TIP.2004.836169.

Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. In *In Intl. Workshop on Comp. Adv. in Multi-Sensor Adapt. Processing, Aruba, Dutch Antilles*, 2009.

G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *ICML*, 2010.

G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):171 –184, 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2012.88.

H. P. Lopuhaä and P. J. Rousseeuw. Breakdown points of affine equivariant estimators of multivariate location and covariance matrices. *Ann. Statist.*, 19(1):229–248, 1991. ISSN 0090-5364.

R. A. Maronna. Robust M-estimators of multivariate location and scatter. *The Annals of Statistics*, 4(1):pp. 51–67, 1976. ISSN 00905364.

R. A. Maronna, R. D. Martin, and V. J. Yohai. *Robust Statistics: Theory and methods*. Wiley Series in Probability and Statistics. John Wiley & Sons Ltd., Chichester, 2006. ISBN 978-0-470-01092-1; 0-470-01092-4.

M. McCoy and J. Tropp. Two proposals for robust PCA using semidefinite programming. *Elec. J. Stat.*, 5:1123–1160, 2011.

S. Mendelson. A few notes on statistical learning theory. In *Lecture Notes in Computer Science*, volume 2600, pages 1–40. Springer-Verlag, 2003.

H. Nyquist. Least orthogonal absolute deviations. *Computational Statistics & Data Analysis*, 6(4):361 – 367, 1988. ISSN 0167-9473. doi: 10.1016/0167-9473(88)90076-X.

M. R. Osborne and G. A. Watson. An analysis of the total approximation problem in separable norms, and an algorithm for the total l_1 problem. *SIAM Journal on Scientific and Statistical Computing*, 6(2):410–424, 1985. doi: 10.1137/0906029.

A. M. Ostrowski. *Solution of Equations and Systems of Equations*. Academic Press, Second edition, September 1966. ISBN 0471889873.

M. Soltanolkotabi and E. J. Candès. A geometric analysis of subspace clustering with outliers. *Ann. Stat.*, 40(4):2195–2238, 2012. doi: 10.1214/12-AOS1034.

H. Späth and G. A. Watson. On orthogonal linear approximation. *Numer. Math.*, 51: 531–543, October 1987. ISSN 0029-599X. doi: 10.1007/BF01400354.

C. V. Stewart. Robust parameter estimation in computer vision. *SIAM Reviews*, 41:513–537, 1999.

T. Tao. *Topics in Random Matrix Theory*, volume 132 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2012. ISBN 978-0-8218-7430-1.

M. Tipping and C. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

F. De La Torre and M. J. Black. Robust principal component analysis for computer vision. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 362 –369 vol.1, 2001. doi: 10.1109/ICCV.2001.937541.

F. De La Torre and M. J. Black. A framework for robust subspace learning. *International Journal of Computer Vision*, 54:117–142, 2003. ISSN 0920-5691. doi: 10.1023/A:1023709501986.

P. Tseng. Nearest q-flat to m points. *Journal of Optimization Theory and Applications*, 105:249–252, 2000. ISSN 0022-3239. 10.1023/A:1004678431677.

D. E. Tyler. A distribution-free M-estimator of multivariate scatter. *Ann. Statist.*, 15(1): 234–251, 1987. ISSN 0090-5364. doi: 10.1214/aos/1176350263.

R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. In *Compressed sensing*, pages 210–268. Cambridge Univ. Press, Cambridge, 2012.

H. Voss and U. Eckhardt. Linear convergence of generalized weiszfeld's method. *Computing*, 25:243–251, 1980. ISSN 0010-485X. doi: 10.1007/BF02242002.

L. Wang and A. Singer. Exact and stable recovery of rotations for robust synchronization. *Information and Inference*, 2013. doi: 10.1093/imaiai/iat005.

G. A. Watson. *Some Problems in Orthogonal Distance and Non-Orthogonal Distance Regression*. Defense Technical Information Center, 2001. URL `http://books.google.com/books?id=WKKWGwAACAAJ`.

G. A. Watson. On the gauss-newton method for l_1 orthogonal distance regression. *IMA Journal of Numerical Analysis*, 22(3):345–357, 2002. doi: 10.1093/imanum/22.3.345.

E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donne's est minimum. *Tohoku Math. J.*, 43:35–386, 1937.

H. Xu, C. Caramanis, and S. Mannor. Principal component analysis with contaminated data: The high dimensional case. In *COLT*, pages 490–502, 2010a.

H. Xu, C. Caramanis, and S. Sanghavi. Robust PCA via outlier pursuit. In *NIPS*, pages 2496–2504, 2010b.

H. Xu, C. Caramanis, and S. Sanghavi. Robust PCA via outlier pursuit. *Information Theory, IEEE Transactions on*, PP(99):1, 2012. ISSN 0018-9448. doi: 10.1109/TIT.2011.2173156.

L. Xu and A.L. Yuille. Robust principal component analysis by self-organizing rules based on statistical physics approach. *Neural Networks, IEEE Transactions on*, 6(1):131–143, 1995. ISSN 1045-9227. doi: 10.1109/72.363442.

T. Zhang. Robust subspace recovery by geodesically convex optimization. *ArXiv e-prints*, 2012.

T. Zhang, A. Szlam, and G. Lerman. Median K-flats for hybrid linear modeling with many outliers. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on Computer Vision*, pages 234–241, Kyoto, Japan, 2009. doi: 10.1109/ICCVW.2009.5457695.

T. Zhang, A. Szlam, Y. Wang, and G. Lerman. Randomized hybrid linear modeling by local best-fit flats. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1927 –1934, jun. 2010. doi: 10.1109/CVPR.2010.5539866.

T. Zhang, A. Szlam, Y. Wang, and G. Lerman. Hybrid linear modeling via local best-fit flats. *International Journal of Computer Vision*, 100:217–240, 2012. ISSN 0920-5691. doi: 10.1007/s11263-012-0535-6.

Journal of Machine Learning Research 15 (2014) 809-883 Submitted 5/13; Revised 11/13; Published 3/14

Policy Evaluation with Temporal Differences: A Survey and Comparison

Christoph Dann CDANN@CDANN.DE
Gerhard Neumann GERI@ROBOT-LEARNING.DE
Technische Universität Darmstadt
Karolinenplatz 5
64289 Darmstadt, Germany

Jan Peters* MAIL@JAN-PETERS.NET
Max Planck Institute for Intelligent Systems
Spemannstraße 38
72076 Tübingen, Germany

Editor: Peter Dayan

Abstract

Policy evaluation is an essential step in most reinforcement learning approaches. It yields a value function, the quality assessment of states for a given policy, which can be used in a policy improvement step. Since the late 1980s, this research area has been dominated by temporal-difference (TD) methods due to their data-efficiency. However, core issues such as stability guarantees in the off-policy scenario, improved sample efficiency and probabilistic treatment of the uncertainty in the estimates have only been tackled recently, which has led to a large number of new approaches.

This paper aims at making these new developments accessible in a concise overview, with foci on underlying cost functions, the off-policy scenario as well as on regularization in high dimensional feature spaces. By presenting the first extensive, systematic comparative evaluations comparing TD, LSTD, LSPE, FPKF, the residual-gradient algorithm, Bellman residual minimization, GTD, GTD2 and TDC, we shed light on the strengths and weaknesses of the methods. Moreover, we present alternative versions of LSTD and LSPE with drastically improved off-policy performance.

Keywords: temporal differences, policy evaluation, value function estimation, reinforcement learning

1. Introduction

Policy evaluation estimates a value function that predicts the accumulated rewards an agent following a fixed policy will receive after being in a particular state. A policy prescribes the agent's action in each state. As value functions point to future success, they are important in many applications. For example, they can provide failure probabilities in large telecommunication networks (Frank et al., 2008), taxi-out times at big airports (Balakrishna et al., 2010) or the importance of different board configurations in the game Go (Silver et al., 2007). Such value functions are particularly crucial in many *reinforcement*

∗. Also at *Technische Universität Darmstadt, Karolinenplatz 5, Darmstadt, Germany.*

learning methods for learning control policies as one of the two building blocks constituting *policy iteration*. In policy iteration, an optimal policy is obtained by iterating between the value prediction for states (and sometimes actions) given the agent's current policy, that is, *policy evaluation*, and improving the policy such that it maximizes the value of all states predicted by the current value function, that is, *policy improvement*. Policy-iteration-based reinforcement learning has yielded impressive applications in robot soccer (Riedmiller and Gabel, 2007), elevator control (Crites and Barto, 1998) and game-playing such as Checkers (Samuel, 1959), Backgammon (Tesauro, 1994) and Go (Gelly and Silver, 2008). For sufficiently accurate value function estimates, policy iteration frequently converges to the optimal policy. Hence, a reliable and precise estimator of the value function for a given policy is essential in reinforcement learning and helpful in many applications.

However, obtaining accurate value function estimates is not a straightforward supervised learning problem. Creating sufficient data for obtaining the value function by regression would require a large number of roll-outs (state-action-reward sequences) in order to acquire the accumulated reward for each considered state. As the variance of the accumulated reward frequently grows with the time horizon, exhaustive number of data points would be required. To circumvent this issue, the idea of bootstrapping has been proposed, that is, the current estimate of the value function is used to generate the target values for learning a new estimate of the value function. In expectation, the sum of the current reward and the discounted value of the next state should match the value of the current state. Hence, their difference becomes an error signal for the value function estimator. The resulting approaches are called temporal-difference methods from their introduction in Sutton (1988). Temporal-difference methods have received a tremendous attention in the last three decades and had a number of important successes including the ones mentioned in the previous paragraph.

While temporal-difference methods have been successful, they have not been understood well for a long time (Tsitsiklis and van Roy, 1997; Schoknecht, 2002), they were data-inefficient (Bradtke and Barto, 1996) , and were not stable if used with function approximation in the off-policy case (Baird, 1995). In the off-policy scenario, the value function is estimated from a data set that was generated from another policy than the one we want to evaluate, which is crucial for data re-use in policy iteration. Recently, there has been a large number of substantial advances both in our understanding of temporal-difference methods as well as in the design of novel estimators that can deal with the problems above. These methods are currently scattered over the literature and usually only compared to the most similar methods. In this survey paper, we attempt at presenting the state of the art combined with a more comprehensive comparison.

This survey has two major contributions. First, we are going to present a principled and structured overview on the classic as well as the recent temporal-difference methods derived from general insights. Second, we are comparing these methods in several meaningful scenarios. This comprehensive experimental study reveals the strengths and weaknesses of temporal-difference methods in different problem settings. These insights on the behavior of current methods can be used to design improvements which overcome previous limitations as exemplified by the alternative off-policy reweighting strategy for LSTD and LSPE proposed in this paper. The remainder of this paper is structured as follows: Sections 1.1 and 1.2 introduce the required background for this paper on Markov decision processes and value functions. As the paper aims at complementing the literature, especially the book

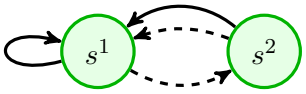


Figure 1: Stationary Distributions for Different Policies. The MDP has deterministic transitions depending on the state (1 or 2) and the action (solid or dashed) illustrated by the arrows. Taking for example the dashed action in state 1 moves the agent always to state 2. A policy which always chooses the solid action leaves the agent always in state 1, that is, $d_{\text{solid}} = [1, 0]^T$, while the dashed counterpart makes the agent alternate between the two states ($d_{\text{dashed}} = [\frac{1}{2}, \frac{1}{2}]^T$). If the agent takes the solid action with probability α, the steady state distribution is given by $d_\alpha = [\frac{1}{2} + \frac{1}{2}\alpha, \frac{1}{2} - \frac{1}{2}\alpha]^T$.

by Sutton and Barto (1998), we illustrate the concept of temporal-difference methods for policy evaluation already in Section 1.2. In Section 2, we present a structured overview of current policy evaluation methods based on temporal differences. This overview starts out by presenting the core objective functions that underlie the various different temporal-difference-based value function methods. We show how different algorithms can be designed by using different optimization techniques, such as stochastic gradient descent, least-squares methods or even Bayesian formulation, resulting in a large variety of algorithms. Furthermore, we illustrate how these objectives can be augmented by regularization to cope with overabundant features. We present important extensions of temporal-difference learning including eligibility traces and importance reweighting for more data-efficiency and estimation from off-policy samples. As Section 2 characterizes methods in terms of design decisions, it also sheds light on new combinations not yet contributed to the literature. In Section 3, we first present a series of benchmark tasks that are being used for comparative evaluations. We focus particularly on the robustness of the different methods in different scenarios (e.g., on-policy vs. off-policy, continuous vs. discrete states, number of features) and for different parameter settings (i.e., the open parameters of the algorithms such as learning rates, eligibility traces, etc). Subsequently, a series of important insights gained from the experimental evaluation is presented including experimental validation of known results as well as new ones which are important for applying value-function estimation techniques in practice. The paper is concluded in Section 4 with a short summary and an outlook on the potential future developments in value-function estimation with temporal differences.

1.1 Notation and Background on Markov Decision Processes

The learning agent's task is modeled as a *Markov decision process* (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R)$. At each discrete time step $t = 0, 1, 2 \ldots$, the system is in a state $s_t \in S$ and the agent chooses an action $a_t \in \mathcal{A}$. The state of the next time step is then determined by the transition model $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, that is, $\mathcal{P}(s_{t+1}|a_t, s_t)$ is the conditional probability (density) for transitioning from s_t to s_{t+1} with action a_t. After each transition, the agent receives a reward $r_t = R(s_t, a_t)$ specified by the deterministic *reward function* $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. We distinguish between discrete systems and continuous systems. While continuous systems

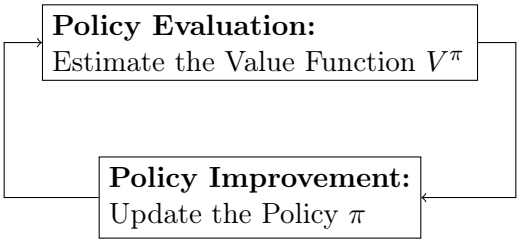


Figure 2: Policy Iteration Algorithm

have infinitely many states (and actions), discrete systems are usually restricted to a finite number of states. For notational simplicity, we mostly treat $\mathcal{S}$ and $\mathcal{A}$ to be finite sets in the remainder of this paper. Nevertheless, the analysis presented in this paper often generalizes to continuous/infinite state-spaces.

The behavior of the learning agent within the environment, that is, the action-selection strategy given the current state, is denoted by a *policy* π. A stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defines a probability distribution over actions given a state s_t. The agent samples from π to select its actions. Stochasticity of the policy promotes state exploration, a key component of robust policy learning. However, in certain cases a deterministic policy can be easier to handle. Such a policy can be treated as a deterministic function $\pi : \mathcal{S} \to \mathcal{A}$ with $a_t = \pi(s_t)$.

While we also consider episodic Markov decision processes in examples and experiments, we concentrate on ergodic MDPs for the formal part to keep the theoretical analysis concise. Ergodic Markov decision processes do not terminate and the agent can interact with its environment for an infinite time. Their underlying stochastic processes have to be ergodic, which, in highly simplified terms, means that every state can be reached from all others within a finite amount of time steps (for details and exact definitions see for example the work of Rosenblatt, 1971). If these assumptions hold, there exists a *stationary distribution* d^π over $\mathcal{S}$ with $d^\pi(s') = \sum_{s,a} \mathcal{P}(s'|s,a)\pi(a|s)d^\pi(s)$. This distribution yields the probability of the process being in state s when following policy π, that is, sampled states from the MDP with policy π are identically distributed samples from d^π. While they are not (necessarily) independently drawn, ergodicity ensures that the strong law of large numbers still holds. Formally, MDPs do not need to have unique limiting distributions. Instead, a distribution defined as $d^\pi(s) = \mathbb{E}_{\pi,\mathcal{P}}\left[\sum_{t=0}^{\infty} 1_{\{s_t=s\}}\right]$ would suffice in most cases. For fixed policies π, we can rewrite the definition d^π more concisely as $d^\pi(s) = \mathbb{E}_{\mathcal{P}^\pi}\left[d^\pi(s)\right]$, where $\mathcal{P}^\pi$ denotes the state transition distribution

$$\mathcal{P}^\pi(s_{t+1}|s_t) = \sum_{a_t} \mathcal{P}(s_{t+1}|a_t, s_t)\pi(a_t|s_t).$$

Marginalizing out the action reduces the MDP to a Markov chain. Even though the actions are marginalized out, the policy affects $\mathcal{P}^\pi$ and, thus, the stationary distribution is highly dependent on π. See Figure 1 for an example.

Reinforcement learning aims at finding a policy that maximizes the *expected (total discounted) future reward*

$$J(\pi) = \mathbb{E}_{\mathcal{P},\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right].$$

The discount factor $\gamma \in [0, 1)$ controls the considered timespan or planning horizon. Small discount factors emphasize earlier rewards while rewards in the future are becoming less relevant with time.

A common family of iterative reinforcement learning algorithms for finding the optimal policy is policy iteration. Policy iteration algorithms alternate between a *policy evaluation* and a *policy improvement* step (see Figure 2). In the policy evaluation step, the value function $V^\pi : \mathcal{S} \to \mathbb{R}$ for the current policy is estimated. The value function corresponds to the expected accumulated future reward

$$V^\pi(s) = \mathbb{E}_{\mathcal{P},\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \;\middle|\; s_0 = s\right], \tag{1}$$

given that the process started in state s, and that the actions are chosen according to policy π. Hence, the value function evaluates the policy in each state. It allows the subsequent policy improvement step to obtain a policy which chooses actions that move the agent most likely in states with the highest values.

1.2 Problem Statement: Efficient Value Function Estimation

This paper discusses different approaches to estimate the value function in Equation (1) while observing the agent interacting with its environment. More formally, the problem of *value-function estimation* can be defined as follows:

The value function of a target policy π_G and a given MDP $\mathcal{M}$ is estimated based on the data set $\mathcal{D} = \{(s_t, a_t, r_t; t = 1 \dots t_f), (s_t, a_t, r_t; t = 1 \dots t_f), \dots\}$ sampled from the MDP $\mathcal{M}$ and a behavior policy π_B. The data set $\mathcal{D}$ may consist of one or more roll-outs $(s_t, a_t, r_t; t = 1 \dots t_f)$. We distinguish between *on-policy* estimation ($\pi_B = \pi_G$) and *off-policy* estimation ($\pi_B \neq \pi_G$). The latter scenario is particularly appealing for policy iteration, as we can re-use samples from previous policy evaluation iterations for the current value function.

To illustrate major challenges of value-function estimation we consider a classic 2D grid-world example shown in Figure 3, a simple benchmark task often used in reinforcement learning. To estimate the value of the agent's position, we have to compute the expectation in Equation (1). We can approximate this value directly with Monte-Carlo methods, that is, take the average of the accumulated reward computed for several roll-outs starting from this position (Sutton and Barto, 1998, Chapter 5). However, the variance of the accumulated reward will be huge as the stochasticity of each time-step often adds up in the accumulated rewards. For example one roll-out may yield a high reward as the agent always moves in the directions prescribed by the policy, while another roll-out may yield very low reward as the agent basically performs a random walk due to the transition probabilities of the MDP. Hence, even if we have a model of the MDP to simulate the agent until future rewards are sufficiently discounted, the value estimate of Monte-Carlo methods is typically highly inaccurate in any reasonable limit of roll-outs.

The crux is the dependency of the state value on future rewards, and subsequently on the state after many time-steps. The problem simplifies with decreasing discount factor γ and reduces to standard supervised learning for $\gamma = 0$ (estimate immediate reward $\mathbb{E}[r_t | s_t = s]$). Bootstrapping is an approach to circumvent the problems of long-time dependencies using

a recursive formulation of the value function. This recursion can be obtained by comparing Equation (1) for two successive time-steps t and $t+1$

$$V^\pi(s) = \mathbb{E}_{\mathcal{P},\pi}\Big[r(s_t, a_t) + \gamma V^\pi(s_{t+1})\Big| s_t = s\Big]. \tag{2}$$

This so-called *Bellman equation*[1] holds true for arbitrary MDPs $\mathcal{M}$, discount factors γ and policies π. This basic insight allows us to update the value estimate of the current state based on the observed reward r_t and the value estimate for the successor state s_{t+1}. In the long run, the estimate is changed such that the difference of the values of temporally subsequent states (temporal difference) matches the observed rewards in expectation. This bootstrapping approach is the foundation for efficient value-function estimation with temporal-difference methods, as it drastically reduces the variance of the estimator. Yet, it may also introduce a bias (cf. Section 2.1).

To simplify notation we will write V^π as a $m = |\mathcal{S}|$ dimensional vector $\boldsymbol{V}^\pi$ which contains $V^\pi(s^i)$ at position i for a fixed order $s^1, s^2, \dots s^m$ of the states. Using the same notation for the rewards, that is, $\boldsymbol{R}^\pi \in \mathbb{R}^m$ with $\boldsymbol{R}^\pi_i = \mathbb{E}_\pi\big[r(s^i, a)\big]$, the Bellman equation can be rewritten as

$$\boldsymbol{V}^{\boldsymbol{\pi}} = \boldsymbol{R}^{\boldsymbol{\pi}} + \gamma \boldsymbol{P}^{\boldsymbol{\pi}} \boldsymbol{V}^{\boldsymbol{\pi}} =: T^\pi \boldsymbol{V}^{\boldsymbol{\pi}}. \tag{3}$$

Here, the transition matrix $\boldsymbol{P}^{\boldsymbol{\pi}} \in \mathbb{R}^{m\times m}$ of policy π contains the state transitions probabilities $P^\pi_{ij} = \sum_a \mathcal{P}(s^i|s_j, a)\pi(a|s_j)$. As we can see, the Bellman equation specifies a fixpoint of an affine transformation $T^\pi : \mathbb{R}^m \to \mathbb{R}^m$ of $\boldsymbol{V}^\pi$ (*Bellman operator*). We will omit the policy superscripts in unambiguous cases for notational simplicity.

The depicted world in Figure 3 consists of $15 \times 15 = 225$ states, that is, we have to estimate 225 values. Yet, such a small world can only be used for highly simplified tasks. More realistic settings (such as street navigation) require much finer and larger grids or have to allow arbitrary continuous positions $s \in \mathbb{R}^2$. This requirement illustrates another inherent problem of value estimation, the curse of dimensionality, that is, the number of states $|\mathcal{S}|$ increases exponentially with the number of state variables. For example, if there are several moving objects in our grid world, the number of states $|\mathcal{S}|$ explodes. In a 15×15 grid world with one agent and 9 moving objects, we have to estimate $(15 \times 15)^{10} \approx 3^{23}$ values. Thus, we almost always need to resort to approximation techniques for the value function. The simplest and most common approach is a linear parametrization with parameter vector $\boldsymbol{\theta} \in \mathbb{R}^n$, that is,

$$V(s) \approx V_\theta(s) = \boldsymbol{\theta}^T \boldsymbol{\phi}(s),$$

where $\boldsymbol{\phi}(s)$ defines features of the state s.

The feature function $\boldsymbol{\phi} : \mathcal{S} \to \mathbb{R}^n$ reduces the number of parameters which we need to estimate from m to n with $n \ll m$ but comes at the price of less precision. Hence, the choice of a feature representation is always a trade-off between compactness and expressiveness, where the latter means that there exists a V_θ that is close to V for all states.

Estimation techniques for alternative parametrizations of V exist, such as non-linear function approximation (e.g.,, see non-linear versions of GTD and TDC, Maei, 2011, Chapter 6) or automatically built representations (cf. Section 2.3). However, defining non-linear

1. Bellman would not have claimed this equation but rather the principle of optimality (source: personal correspondence with Bellman's former collaborators).

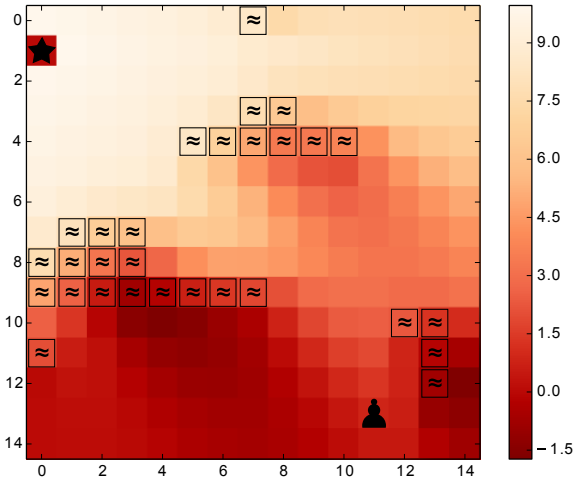


Figure 3: Classic 2D Grid-World Example: The agent ♟ obtains a positive reward (10) when it reaches the goal and negative (-2) ones when it goes through water ≈. The agent always chooses a direction (up, right, down left) that points towards the goal ★. With probability 0.8 the agent moves in that direction and with 0.2 in a random direction. We are interested in the value $V(s)$ of each possible position (state) of the agent with a discount of $\gamma = 0.99$. The value $V(s)$ is shown as an overlay.

function approximations by hand requires domain knowledge to an extent that is usually not available and the learning problem typically becomes non-convex, that is, the estimator may get stuck in local, but not global, optima. Therefore, this paper focuses on more commonly used linear function approximation.

After having identified temporal differences and function approximation as the key ingredients for efficient policy evaluation, we can concentrate on important properties of value function estimators. In robotics and many other fields, the data gathering process is very costly and time consuming. In such cases, algorithms need to be data efficient and yield accurate estimates already after few observations. In other applications, accurate models of the learning environment are available and observations can be generated efficiently by simulation. Hence, the focus is shifted to efficiency in terms of computation time. Computation time is also a limiting factor in online and real-time learning, which require to update the value estimations after each observation within a limited time frame.

2. Overview of Temporal-Difference Methods

Value estimation can be cast as an optimization problem, and, in fact, most temporal-difference methods are direct applications of optimization techniques. Hence, their characteristics are largely determined by (1) the chosen objective or cost function, and (2) how this function is optimized. We start by discussing different optimization objectives in Section 2.1 that build the basis of most theoretical results and define the quality of the value estimates

after enough observations. In Section 2.2, we introduce temporal-difference methods by grouping them according to the employed optimization technique as algorithms within a group share similar convergence speed and computational demands. To avoid cluttered notation and to put the focus on their intrinsic characteristics, we present the algorithms in their basic form and omit eligibility traces and importance weights for the off-policy case here (these are discussed in Section 2.4). Complete versions of the algorithms with available extensions can be found in Appendix C.

Reliable value estimates are the result of fruitful interplay between expressive feature descriptors $\boldsymbol{\phi}$ and suitable estimation algorithms. Yet, choosing appropriate feature functions poses a hard problem when insufficient domain knowledge is available. Several approaches have been proposed to simplify the generation and handling of features for temporal-difference methods. We review these recent efforts in Section 2.3. Finally, in Section 2.4, we discuss two important extensions applicable to most methods. The first extension are eligibility traces, which reduce bootstrapping and may increase convergence speed. Subsequently, we discuss importance reweighting for off-policy value-function estimation.

2.1 Objective Functions

We are interested in estimating parameters $\boldsymbol{\theta}$ that yield a value function $\boldsymbol{V}_\theta$ as close as possible to the true value function $\boldsymbol{V}^\pi$. This goal directly corresponds to minimizing the *mean squared error* (MSE)

$$\begin{aligned}\mathrm{MSE}(\boldsymbol{\theta}) &= \|\boldsymbol{V_\theta} - \boldsymbol{V^\pi}\|^2_{\boldsymbol{D}} = \sum_{i=1}^{m} d^\pi(s^i)[V_{\boldsymbol{\theta}}(s^i) - V^\pi(s^i)]^2 \\ &= [\boldsymbol{V_\theta} - \boldsymbol{V^\pi}]^T \boldsymbol{D}[\boldsymbol{V_\theta} - \boldsymbol{V^\pi}]. \end{aligned} \tag{4}$$

The weight matrix $\boldsymbol{D} = \mathrm{diag}[d^\pi(s^1), d^\pi(s^2), \ldots, d^\pi(s^m)]$ has the entries of the stationary distribution d^π on the diagonal and weights each error according to its probability. The true value function $\boldsymbol{V}^\pi$ can be obtained by Monte-Carlo estimates, that is, performing roll-outs with the current policy and collecting the long-term reward. However, the Monte-Carlo estimate of $\boldsymbol{V}^\pi$ requires a lot of samples as it suffers from a high variance.

The high variance can be reduced by eliminating the true value function $\boldsymbol{V}^\pi$ from the cost function. To do so, we can use *bootstrapping* (Sutton and Barto, 1998), where $\boldsymbol{V}^\pi$ is approximated by a one-step prediction based on the approximated value-function. Hence, we minimize the squared difference between the two sides of the Bellman equation (3) which corresponds to minimizing

$$\mathrm{MSBE}(\boldsymbol{\theta}) = \|\boldsymbol{V_\theta} - T\boldsymbol{V_\theta}\|^2_{\boldsymbol{D}}. \tag{5}$$

This objective, called the *mean squared Bellman error*, can be reformulated in terms of expectations using Equation (2)

$$\begin{aligned}\mathrm{MSBE}(\boldsymbol{\theta}) &= \sum_{i=1}^{n} d^\pi(s^i)[V_{\boldsymbol{\theta}}(s^i) - \mathbb{E}_{\mathcal{P},\pi}[r(s_t, a_t) + \gamma V_{\boldsymbol{\theta}}(s_{t+1})|\pi, s_t = s^i]]^2 \\ &= \mathbb{E}_d\big[\big(V_{\boldsymbol{\theta}}(s) - \mathbb{E}_{\mathcal{P},\pi}[r(s_t, a_t) + \gamma V_{\boldsymbol{\theta}}(s_{t+1})|\pi, s_t = s]\big)^2\big], \end{aligned} \tag{6}$$

where the outer expectation is taken with respect to the stationary distribution d^π and the inner one with respect to the state dynamics $\mathcal{P}$ of the MDP and the policy π. Let δ_t denote

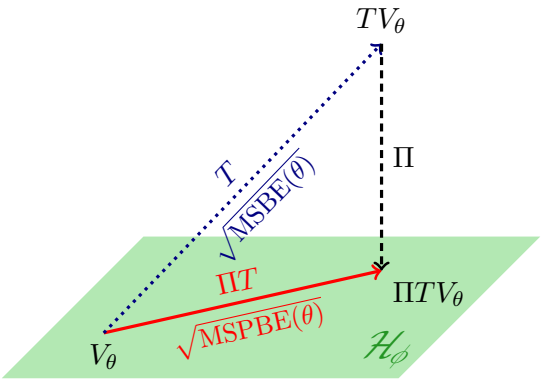


Figure 4: MSBE compares the current value estimate $\boldsymbol{V_\theta}$ to the T-transformed one $T\boldsymbol{V_\theta}$. In contrast, MSPBE is a distance in the space of parameterized functions $\mathcal{H}_{\boldsymbol{\phi}}$ and always smaller or equal than MSBE, as Π is an orthogonal projection. Figure adopted from Lagoudakis and Parr (2003).

the temporal-difference (TD) error which is given by the error in the Bellman equation for time step t

$$\delta_t = r(s_t, a_t) + \gamma V_{\boldsymbol{\theta}}(s_{t+1}) - V_{\boldsymbol{\theta}}(s_t) = r_t + (\gamma \boldsymbol{\phi}_{t+1} - \boldsymbol{\phi}_t)^T \boldsymbol{\theta}, \tag{7}$$

with $\boldsymbol{\phi}_t = \boldsymbol{\phi}(s_t)$. Equation (6) can then be written concisely as $\text{MSBE}(\boldsymbol{\theta}) = \mathbb{E}_d[\mathbb{E}_{\mathcal{P},\pi}[\delta_t|s_t]^2]$. Using the second form of δ_t from Equation (7) to formulate the MSBE error as

$$\text{MSBE}(\boldsymbol{\theta}) = \mathbb{E}_d\left[\left(\left(\mathbb{E}_{\mathcal{P},\pi}[\gamma\boldsymbol{\phi}_{t+1}|s_t] - \boldsymbol{\phi}_t\right)^T \boldsymbol{\theta} + \mathbb{E}_{\mathcal{P},\pi}[r_t|s_t]\right)^2\right]$$

makes apparent that the MSBE objective corresponds to a linear least-squares regression model with inputs $\gamma\mathbb{E}_{\mathcal{P},\pi}[\boldsymbol{\phi}_{t+1}|s_t] - \boldsymbol{\phi}_t$ and outputs $-\mathbb{E}_{\mathcal{P},\pi}[r_t|s_t]$. However, we actually cannot observe the inputs but only noisy samples $\gamma\boldsymbol{\phi}_{t+1} - \boldsymbol{\phi}_t$. The least-squares regression model does not account for this noise in the input variables, known as *error-in-variables* situation (Bradtke and Barto, 1996). As we will discuss in Section 2.2.1, this deficiency requires that two independent samples of s_{t+1} need to be drawn when being in state s_t, also known as the *double-sampling problem* (Baird, 1995). Hence, samples generated by a single roll-out cannot be used directly without introducing a bias. This bias corresponds to minimizing the *mean squared temporal-difference error* (Maei, 2011)

$$\text{MSTDE}(\boldsymbol{\theta}) = \mathbb{E}_{d,\mathcal{P},\pi}[\delta_t^2] = \mathbb{E}_d[\mathbb{E}_{\mathcal{P},\pi}[\delta_t^2|s_t]]. \tag{8}$$

The square is now inside the expectation. This cost function has a different optimum than the MSBE and, hence, minimizing it results in a different value function estimate.

Another possibility to avoid the optimization problems connected to the MSBE is instead to minimize the distance of the projected Bellman operator, also called the *mean squared projected Bellman error* (MSPBE). The Bellman operator in Equation (3) is independent of the feature representation, and, hence, $T\boldsymbol{V}_\theta$ may not be representable using the given features. The MSPBE only yields the error which is representable with the given features

and neglects the error orthogonal to the feature representation. Most prominent temporal-difference methods such as TD learning (Sutton, 1988), LSTD (Bradtke and Barto, 1996), GTD (Sutton et al., 2008) and TDC (Sutton et al., 2009) either directly minimize the MSPBE or converge to the same fixpoint (Tsitsiklis and van Roy, 1997; Sutton et al., 2008, 2009). The MSPBE is given by the squared distance between $\boldsymbol{V_\theta}$ and the representable function $\Pi T \boldsymbol{V_\theta}$ that is closest to $T\boldsymbol{V_\theta}$

$$\text{MSPBE}(\boldsymbol{\theta}) = \|\boldsymbol{V_\theta} - \Pi T \boldsymbol{V_\theta}\|^2_{\boldsymbol{D}}, \tag{9}$$

where Π is a projection operator which projects arbitrary value functions onto the space of representable functions $\mathcal{H}_{\boldsymbol{\phi}}$. For linear function approximation, the projection Π has the closed form

$$\Pi \boldsymbol{V} = \min_{V_\theta \in \mathcal{H}_\phi} \|\boldsymbol{V_\theta} - \boldsymbol{V}\|^2_{\boldsymbol{D}} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T D \boldsymbol{V},$$

where $\boldsymbol{\Phi} = [\boldsymbol{\phi}(s^1), \boldsymbol{\phi}(s^2), \dots \boldsymbol{\phi}(s^m)]^T \in \mathbb{R}^{m \times n}$ is the feature matrix consisting of rows with features of every state.

An illustration of the differences between the cost function can be found in Figure 4. The MSBE compares a parameterized value function $\boldsymbol{V_\theta}$ against $T\boldsymbol{V_\theta}$ (see the dotted distance in Figure 4), which may lie outside the space of parameterized functions $\mathcal{H}_{\boldsymbol{\phi}}$. The MSPBE first projects $T\boldsymbol{V_\theta}$ on the set of representable functions and, subsequently, calculates the error (solid distance).

Example 1 *For a simple example of the different distance functions, consider an MDP of two states and one action. The transition probabilities of the MDP and policy are uniform, that is,*

$$\boldsymbol{P}^\pi = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

The agent receives reward $r^1 = -0.8$ in the first state and $r^2 = 1.2$ in the second state. With a discount factor of $\gamma = 0.8$ the true value function is then given by $\boldsymbol{V}^\pi = (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1}[-0.8\ 1.2]^T = [0\ 2]^T$. If we only use a single constant feature $\phi(s) = 1, \forall s \in \mathcal{S}$, the feature matrix is $\Phi = [1\ 1]^T$ and the parametrization $\boldsymbol{V_\theta} = [1\ 1]^T \boldsymbol{\theta}$ assigns the same value to all states. Hence, the true value function $\boldsymbol{V}^\pi$ cannot be represented by any parameter, that is, $\text{MSE}(\boldsymbol{\theta}) > 0\ \forall \boldsymbol{\theta}$. In addition, $\gamma \boldsymbol{P}^\pi \boldsymbol{V}_\theta$ is always a vector with equal components and subsequently $T\boldsymbol{V}_\theta = [-0.8\ 1.2]^T + \gamma \boldsymbol{P}^\pi \boldsymbol{V}_\theta$ has entries different from each other and the MSBE is always greater 0. One can easily verify that the projection is a simple average-operator $\boldsymbol{\Pi} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ and that $\boldsymbol{\theta} = 1$ satisfies $\boldsymbol{V}_\theta - \Pi T \boldsymbol{V_\theta} = 0$, that is, $\text{MSPBE}(\boldsymbol{\theta}) = 0$.

The MSPBE circumvents the optimization problems connected to the MSBE, but instead loses the direct connection to the original MSE, the quantity we truly want to minimize. As shown by Sutton et al. (2009), the MSPBE can also be written as

$$\text{MSPBE}(\boldsymbol{\theta}) = \|\boldsymbol{V_\theta} - T\boldsymbol{V_\theta}\|^2_{\boldsymbol{U}} = \|\Phi^T \boldsymbol{D}(\boldsymbol{V_\theta} - T\boldsymbol{V_\theta})\|^2_{(\Phi^T \boldsymbol{D} \Phi)^{-1}}, \tag{10}$$

with $\boldsymbol{U} = \boldsymbol{D}\boldsymbol{\Phi}(\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D}$. A derivation of this formulation is given in Appendix A. This formulation reveals two important insights for understanding the MSPBE. First, the

MSPBE still measures the MSBE, just the metric is now defined as $\boldsymbol{U}$ instead of $\boldsymbol{D}$. Second, the minimum of the MSPBE is reached if and only if

$$\boldsymbol{\Phi}^T \boldsymbol{D}(\boldsymbol{V}_{\boldsymbol{\theta}} - T\boldsymbol{V}_{\boldsymbol{\theta}}) = \mathbb{E}_{d,\mathcal{P},\pi}[\delta_t \boldsymbol{\phi}_t] = \mathbf{0}. \tag{11}$$

This condition means that there is no correlation between the temporal-difference error δ_t and the feature vector $\boldsymbol{\phi}(s_t)$. Many algorithms, such as TD learning and LSTD have been shown to minimize the MSPBE as their fixpoints satisfy $\mathbb{E}_{d,\mathcal{P},\pi}[\delta_t \boldsymbol{\phi}_t] = \mathbf{0}$.

The insight that the fixpoint of TD learning has the property $\mathbb{E}_{d,\mathcal{P},\pi}[\delta_t \boldsymbol{\phi}_t] = \mathbf{0}$ has also motivated the *norm of the expected TD update*

$$\mathrm{NEU}(\boldsymbol{\theta}) = \|\boldsymbol{\Phi}^T \boldsymbol{D}(\boldsymbol{V}_{\boldsymbol{\theta}} - T\boldsymbol{V}_{\boldsymbol{\theta}})\|_2^2 = \mathbb{E}_{d,\mathcal{P},\pi}[\delta_t \boldsymbol{\phi}_t]^T \mathbb{E}_{d,\mathcal{P},\pi}[\delta_t \boldsymbol{\phi}_t] \tag{12}$$

as an alternative objective function. It shares the same minimum as the MSPBE but has a different shape, and therefore yields different optimization properties such as speed of convergence.

Many algorithms solve the problem of finding the minimum of the MSPBE indirectly by solving a nested optimization problem (Antos et al., 2008; Farahmand et al., 2008) consisting of the minimization of the *operator error (OPE)* and the *fixed-point error (FPE)*. The problem is given by

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}'} \mathrm{OPE}(\boldsymbol{\theta}', \boldsymbol{\omega}) = \arg\min_{\boldsymbol{\theta}'} \|\boldsymbol{V}_{\boldsymbol{\theta}'} - T\boldsymbol{V}_{\boldsymbol{\omega}}\|_{\boldsymbol{D}}^2 \quad \text{and} \tag{13}$$

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega}'} \mathrm{FPE}(\boldsymbol{\theta}, \boldsymbol{\omega}') = \arg\min_{\boldsymbol{\omega}'} \|\boldsymbol{V}_{\boldsymbol{\theta}} - \boldsymbol{V}_{\boldsymbol{\omega}'}\|_{\boldsymbol{D}}^2 = \arg\min_{\boldsymbol{\omega}'} \|\boldsymbol{\Phi}(\boldsymbol{\theta} - \boldsymbol{\omega}')\|_{\boldsymbol{D}}^2. \tag{14}$$

Minimizing the MSPBE is split into two problems where we maintain two estimates of the parameters $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$. In the operator problem, we try to approximate the Bellman operator applied to the value function $V_{\boldsymbol{\omega}}$ with $V_{\boldsymbol{\theta}}$. In the fixpoint problem, we reduce the distance between both parameter estimates $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$. Many algorithms solve this problem by alternating between improving the operator and fixed-point error.

To see that the FPE-OPE solution indeed minimizes the MSPBE, we first look at the optimality conditions of the error functions. By considering the first order optimality criterion of the OPE

$$\mathbf{0} = \nabla_{\boldsymbol{\theta}} \mathrm{OPE}(\boldsymbol{\theta}, \boldsymbol{\omega}) = \boldsymbol{\Phi}^T \boldsymbol{D}(\boldsymbol{\Phi}\boldsymbol{\theta} - \gamma \boldsymbol{\Phi}'\boldsymbol{\omega} - \boldsymbol{R}) \underset{\boldsymbol{\omega}=\boldsymbol{\theta}}{=} \boldsymbol{\Phi}^T \boldsymbol{D}(\boldsymbol{V}_{\boldsymbol{\theta}} - T\boldsymbol{V}_{\boldsymbol{\theta}})$$

and using optimality in the fixpoint problem ($\boldsymbol{\omega} = \boldsymbol{\theta}$), we see that solving the nested OPE-FPE problem (13) – (14) indeed corresponds to minimizing the MSPBE from Equation (11). Note that the optimal value of the operator error is equal to the MSBE value due to $\boldsymbol{\omega} = \boldsymbol{\theta}$ and $\mathrm{OPE}(\boldsymbol{\omega}, \boldsymbol{\omega}) = \|\boldsymbol{V}_{\boldsymbol{\omega}} - T\boldsymbol{V}_{\boldsymbol{\omega}}\|_{\boldsymbol{D}}^2 = \mathrm{MSBE}(\boldsymbol{\omega})$. Yet, the problem does not corresponds to minimizing the MSBE as only one of the parameter vectors can change at a time. The OPE-FPE formulation is particularly appealing as it does not suffer from the double-sampling problem.

2.1.1 Fixpoint Discussion

Most temporal-difference methods for value estimation converge either to the minimum of MSBE or MSPBE. Thus, the properties of both functions as well as their relation to each

other and the mean squared error have been examined thoroughly by Schoknecht (2002) and Scherrer (2010) and in parts by Bach and Moulines (2011), Lazaric et al. (2010), Sutton et al. (2009) and Li (2008).

In the following, we summarize the most important results for both cost functions. First, MSBE and MSPBE are quadratic functions that are strongly convex if the features are linearly independent, that is, $\text{rank}(\boldsymbol{\Phi}) = m$. Linearly independent features are a necessary assumption for the convergence of most temporal-difference methods. Convexity of the cost function guarantees that optimization techniques such as gradient descent do not get stuck in a non-global minimum. Second, the MSBE is larger than the MSPBE for any fixed $\boldsymbol{\theta}$ as

$$\text{MSBE}(\boldsymbol{\theta}) = \text{MSPBE}(\boldsymbol{\theta}) + \|TV_{\boldsymbol{\theta}} - \Pi TV_{\boldsymbol{\theta}}\|^2_{\boldsymbol{D}},$$

where $\|TV_{\boldsymbol{\theta}} - \Pi TV_{\boldsymbol{\theta}}\|^2_{\boldsymbol{D}}$ is the projection error (dashed distance in Figure 4). As Π is an orthogonal projection, this insight follows directly from the Pythagorean theorem (cf. Figure 4).

In addition, Williams and Baird (1993) as well as Scherrer (2010) have derived a bound on the MSE by the MSBE

$$\text{MSE}(\boldsymbol{\theta}) \leq \frac{\sqrt{C(d)}}{1-\gamma} \text{MSBE}(\boldsymbol{\theta}), \text{ with } C(d) = \max_{s^i, s^j} \frac{\sum_a P(s^j|s^i, a)\pi(a|s^i)}{d^\pi(s^i)} \tag{15}$$

where $C(d)$ is a constant concentration coefficient depending on π and $\mathcal{P}$. The numerator contains the average probability of transitioning from s^i to s^j while the denominator is the probability of the stationary distribution at state s^i. The term $C(d)$ becomes minimal if the transitions of the MDP are uniform. For the MSPBE no such general bound exists, as the MSPBE only considers part of the MSBE and ignores the projection error. Under mild conditions, the optimal value of MSPBE is always 0, while optima of MSE and MSBE may have larger values (see Example 1). Bertsekas and Tsitsiklis (1996) and Scherrer (2010) provide an example, where the projection error is arbitrarily large, and the MSE value of the MSPBE optimum is therefore unbounded as well.

MSBE and MSPBE solutions (also referred to as fixpoints, as they satisfy $\boldsymbol{V}_{\boldsymbol{\theta}} = T\boldsymbol{V}_{\boldsymbol{\theta}}$ and $\boldsymbol{V}_{\boldsymbol{\theta}} = \Pi T\boldsymbol{V}_{\boldsymbol{\theta}}$ respectively) have been characterized as different projections of the true value function V onto the space of representable functions $\mathcal{H}_\phi$ by Schoknecht (2002) and Scherrer (2010). These results imply that, if $V^\pi \in \mathcal{H}_\phi$, that is, there exist parameters for the true value function, algorithms optimizing MSPBE or MSBE converge to the true solution. If V^π cannot be represented, the optima of MSBE and MSPBE are different in general. The natural question, which objective yield better solutions in terms of MSE value, is addressed by Scherrer (2010). The minimum of MSPBE often has a lower mean squared error, however, the solution may get unstable and may yield estimates arbitrarily far away from the true solution V^π. Scherrer (2010) illustrated this effect by an example MDP with unstable MSPBE solutions for certain settings of the discount factor γ. On the other hand, the MSBE has been observed to have higher variance in its estimate and is therefore harder to minimize than the MSPBE even if we solve the double-sampling problem (see Section 3.2). While the bound in Equation (15) gives a quality guarantee for the MSBE solution, in practice, it may be too loose for many MDPs as shown in Section 3. In addition, the MSPBE was observed to result in control policies of higher quality, if used as objective

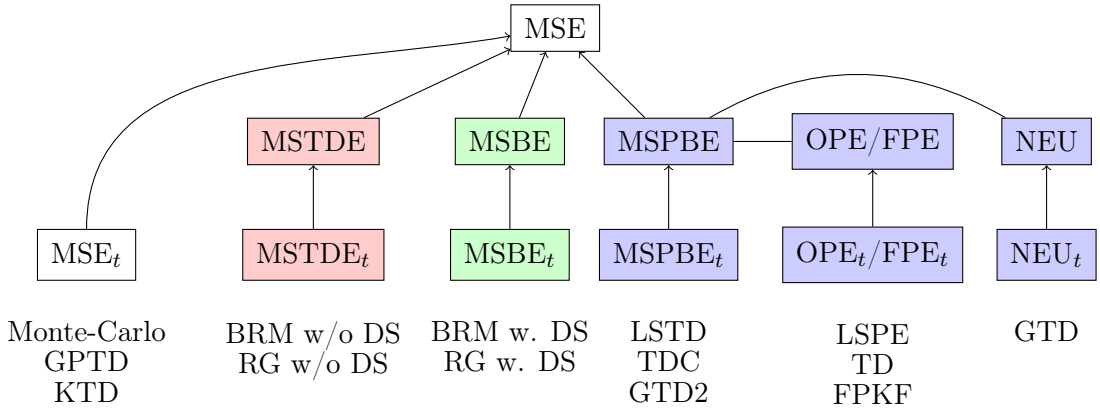


Figure 5: Relations between cost functions and temporal-difference learning algorithms. The methods are listed below the sample-based objective function, which they minimize at timestep t, denoted by the subscript t. The basic idea of temporal difference learning is to optimize for a different (potentially biased) objective function instead of the MSE directly, since its sample-based approximation MSE_t at timestep t converges very slowly to the MSE (due to the large sample variance). The MSPBE, OPE/FPE and NEU objectives (blue shaded) share the same fixed-point and their algorithms converge therefore to the same solution, but possibly at different pace.

functions in a policy iteration loop (Lagoudakis and Parr, 2003). For these reasons the MSPBE is typically preferred. While MSBE and MSPBE have been studied in detail, the quality of the MSTDE cost function and its exact relation to the MSBE are still open questions.

Figure 5 provides a visual overview of the important cost functions and their respective algorithms which are introduced in the following section.

2.2 Algorithm Design

In the following discussion, we will categorize temporal-difference methods as a combination of cost functions and optimization techniques. While we introduced the former in the previous section, we now focus on the optimization techniques. Temporal-difference methods for value function estimation rely either on gradient-based approaches, least-squares minimization techniques or probabilistic models to minimize the respective cost function. Each optimization approach and the consequent family of temporal-difference methods is presented in Sections 2.2.1, 2.2.2 and 2.2.3. We do not give the complete derivation for every method but instead aim for providing their key ingredients and highlighting similarities and difference between the families. Table 1 lists all algorithms presented in this section along with their most important properties.

	Fixpoint	Runtime Complexity	Eligibility Traces	Off-Policy Convergence	Idea
TD	MSPBE	$O(n)$	TD(λ)	no	bootstrapped SGD of MSE
GTD	MSPBE	$O(n)$	-	yes	SGD of NEU
GTD2	MSPBE	$O(n)$	GTD2(λ)	yes	SGD of MSPBE
TDC	MSPBE	$O(n)$	GTD(λ)/TDC(λ)	yes	SGD of MSPBE
RG	MSBE / MSTDE	$O(n)$	gBRM(λ)	yes	SGD of MSBE
BRM	MSBE / MSTDE	$O(n^2)$	BRM(λ)	yes	∇MSBE $= 0$
LSTD	MSPBE	$O(n^2)$	LSTD(λ)	yes	∇MSPBE $= 0$
LSPE	MSPBE	$O(n^2)$	LSPE(λ)	yes	recursive LS Min.
FPKF	MSPBE	$O(n^2)$	FPKF(λ)	?	recursive LS Min.
KTD	MSE	$O(n^2)$	-	no	Parameter Tracking by Kalman Filtering
GPTD	MSE	$O(n^2)$	GPTD(λ)	no	Gaussian Process on V

Table 1: Overview of Temporal-Difference Methods. The methods can divided into gradient-based approaches, least-squares methods and probabilistic models (from top to bottom, separated by horizontal lines). The prior beliefs in probabilistic models acts as a regularization of the cost function. The fixpoint of the residual-gradient algorithm (RG) and Bellman residual minimization (BRM) depends on whether independent second samples for successor states are used or not. The convergence analysis of FPKF for off-policy estimation is still an open problem (Scherrer and Geist, 2011; Geist and Scherrer, 2013).

2.2.1 Gradient-Based Approaches

One family of temporal-difference methods relies on *stochastic gradient descent (SGD)* to optimize their cost function. This optimization technique is directly based on stochastic approximation going back to Robbins and Monro (1951).

Stochastic gradient descent is typically applied to functions of the form $f(\boldsymbol{\theta}) = \mathbb{E}_{p(x)}[g(x;\boldsymbol{\theta})]$, where the expectation is usually approximated by samples and the distribution $p(x)$ is independent of $\boldsymbol{\theta}$. The parameter update in gradient descent follows the negative gradient, that is,

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha_k \nabla f(\boldsymbol{\theta}_k) = \boldsymbol{\theta}_k - \alpha_k \mathbb{E}_{p(x)}[\nabla g(x;\boldsymbol{\theta}_k)],$$

where α_k denotes a step-size. While in ordinary gradient descent, also denoted as batch gradient descent, the gradient is calculated using all samples, stochastic gradient descent only evaluates the gradient for one sample $\tilde{x}$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha_k \nabla g(\tilde{x}; \boldsymbol{\theta}_k) \quad \text{with} \quad \tilde{x} \sim p(x).$$

Stochastic updates are guaranteed to converge to a local minimum of f under the mild stochastic approximation conditions (Robbins and Monro, 1951) such that the step-sizes $\alpha_k \geq 0$ satisfy

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \qquad \sum_{k=0}^{\infty} \alpha_k^2 < \infty.$$

All cost functions in Section 2.1 are expectations with respect to the stationary state distribution d^π. Additionally, observations arrive sequentially in online learning, and therefore, stochastic gradient descent is particularly appealing as it requires only one sample per update. Stochastic gradient temporal-difference methods update the parameter estimate $\boldsymbol{\theta}_t$ after each observed transition from the current state s_t to the next state s_{t+1} with action a_t and reward r_t.

Temporal-Difference Learning. Learning signals similar to temporal differences have been used before, for example by Samuel (1959), but the general concept was first introduced by Sutton (1988) with the *temporal-difference (TD) learning* algorithm. It is considered to be the first use of temporal differences for value-function estimation. Sometimes, also subsequent approaches are referred to as TD learning algorithms. To avoid ambiguity, we use the term TD learning only for the first algorithm presented by Sutton (1988) and denote all other approaches with temporal-difference methods.

The idea behind TD learning is to minimize the MSE where we use $T V_{\boldsymbol{\theta}_t}$ as approximation for the true value function V^π. Minimizing this function $\|\boldsymbol{V}_{\boldsymbol{\theta}} - T\boldsymbol{V}_{\boldsymbol{\theta}_t}\|^2_{\boldsymbol{D}}$ w.r.t. $\boldsymbol{\theta}$ by stochastic gradient descent yields the update rule of TD learning in its basic form

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t [r_t + \gamma V_{\boldsymbol{\theta}_t}(s_{t+1}) - V_{\boldsymbol{\theta}_t}(s_t)] \boldsymbol{\phi}_t = \boldsymbol{\theta}_t + \alpha_t \delta_t \boldsymbol{\phi}_t. \tag{16}$$

As the target values $T\boldsymbol{V}_{\boldsymbol{\theta}_t}$ change over time ($T\boldsymbol{V}_{\boldsymbol{\theta}_0}$,$T\boldsymbol{V}_{\boldsymbol{\theta}_1}$,$T\boldsymbol{V}_{\boldsymbol{\theta}_2}$,...), TD learning does not perform stochastic gradient descent on a well-defined objective function. Thus, general stochastic approximation results are not applicable and in fact several issues emerge from the function change. TD learning as in Equation (16) is only guaranteed to converge if the stationary state distribution d^π is used as sampling distribution, that is, on-policy estimation. If the value function is estimated from off-policy samples, we can easily construct scenarios where TD learning diverges (Baird, 1995). For a more detailed discussion of the off-policy case we refer to Section 2.4.2. In addition, Tsitsiklis and van Roy (1997) have shown that the TD learning algorithm can diverge for non-linear function approximation.

TD learning can be understood more clearly as minimization of the nested optimization problem introduced in Equations (13) and (14). More precisely, TD learning first optimizes the fixpoint problem from Equation (14) by setting $\boldsymbol{\omega} = \boldsymbol{\theta}_t$ and then performs a stochastic gradient step on the operator problem from Equation (13). Hence, we can already conclude that the convergence point of TD learning is given by

$$\boldsymbol{V}_{\boldsymbol{\theta}} = \Pi T^\pi \boldsymbol{V}_{\boldsymbol{\theta}},$$

which is the minimum of the MSPBE objective in Equation (9).

As the results in Section 3 show, the performance of TD learning depends on good step-sizes α_t. Hutter and Legg (2007) aim at overcoming the need for optimizing this hyper-parameter. They re-derived TD learning by formulating the least-squares value-function estimate as an incremental update, which yielded automatically adapting learning rates for tabular feature representations. Dabney and Barto (2012) extended this approach to arbitrary features and additionally proposed another adapting step-size scheme which ensures that the value estimates do not increase the temporal-difference errors $\delta_0, \delta_1, \ldots, \delta_t$ observed in previous timesteps. Autostep (Mahmood et al., 2012), a learning-rate adaptation approach for incremental learning algorithms based on stochastic gradient, yields individual step-sizes for each feature which may boost learning speed. It relies on a meta-step-size but works well for a wide range of step lengths which makes specifying the meta-parameter easier than the step-size for TD learning directly.

Residual-Gradient Algorithm. The *residual-gradient* (RG) algorithm (Baird, 1995) minimizes the *mean squared Bellman error (MSBE)* directly by stochastic gradient descent. Its update rule in the most basic form is given by

$$\begin{aligned}\boldsymbol{\theta}_{t+1} =& \boldsymbol{\theta}_t + \alpha_t[r_t + \gamma V_{\boldsymbol{\theta}_t}(s_{t+1}) - V_{\boldsymbol{\theta}_t}(s_t)](\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1}) \\ =& \boldsymbol{\theta}_t + \alpha_t\delta_t(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1}).\end{aligned}$$

The difference compared to TD learning is that the gradient of $V_{\boldsymbol{\theta}_t}(s_{t+1})$ with respect to $\boldsymbol{\theta}_t$ is also incorporated into the update.

Unfortunately, RG methods suffer from the double-sampling problem mentioned in Section 2.1. Consider the gradient of the MSBE, in Equation (5), given by

$$2\mathbb{E}_d\Big[\Big(V_{\boldsymbol{\theta}}(s) - \mathbb{E}_{\pi,\mathcal{P}}[r(s_t,a_t) + \gamma\phi(s_{t+1})^T\boldsymbol{\theta}|s_t = s]\Big)\Big(\phi(s) - \gamma\mathbb{E}_{\pi,\mathcal{P}}[\phi(s_{t+1})|s_t = s]\Big)\Big]. \quad (17)$$

The outer expectation is computed over the steady state distribution and can be replaced by a single term in stochastic gradient. Both inner expectations are taken over the joint of the policy π and the transition distribution $\mathcal{P}$ of the MDP. Multiplying out the brackets yields $\gamma^2\mathbb{E}[\boldsymbol{\phi}_{t+1}]\mathbb{E}[\boldsymbol{\phi}_{t+1}]^T\boldsymbol{\theta}$ besides other terms. If we replace both expectations with the current observation $\boldsymbol{\phi}_{t+1}$, we obtain a biased estimator since

$$\boldsymbol{\phi}_{t+1}\boldsymbol{\phi}_{t+1}^T \underset{\text{Stoch. Approx.}}{\approx} \mathbb{E}[\boldsymbol{\phi}_{t+1}\boldsymbol{\phi}_{t+1}^T|s_t] = \mathbb{E}[\boldsymbol{\phi}_{t+1}|s_t]\mathbb{E}[\boldsymbol{\phi}_{t+1}|s_t]^T + \mathrm{Cov}[\boldsymbol{\phi}_{t+1},\boldsymbol{\phi}_{t+1}].$$

Hence, updating the parameters only with the current sample is biased by the covariances $\mathrm{Cov}[\boldsymbol{\phi}_{t+1},\boldsymbol{\phi}_{t+1}]$ and $\mathrm{Cov}[r_t,\boldsymbol{\phi}_{t+1}]$ with respect to $\mathcal{P}$ and π. While this effect can be neglected for deterministic MDPs and policies (since $\mathrm{Cov}[\boldsymbol{\phi}_{t+1},\boldsymbol{\phi}_{t+1}] = 0, \mathrm{Cov}[r_t,\boldsymbol{\phi}_{t+1}] = 0$), the residual-gradient algorithm does not converge to a minimizer of the MSBE for stochastic MDPs. It has been shown by Maei (2011) that the residual-gradient algorithm converges to a fixed point of the *mean squared TD error*[2] defined in Equation (8) instead. Alternatively,

2. A different characterization of the RG fixpoint was derived by Schoknecht (2002).

each inner expectation in Equation (17) can be replaced by independently drawn samples a'_t, r'_t, s'_{t+1} and a''_t, r''_t, s''_{t+1} of the transition

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t[r'_t + \gamma V_{\boldsymbol{\theta}_t}(s'_{t+1}) - V_{\boldsymbol{\theta}_t}(s_t)](\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}''_{t+1}).$$

With double samples, the residual-gradient algorithm indeed convergences to a fixpoint of the MSBE. However, a second sample is only available, if the model of the MDPs is known or a previously observed transition from the same state is reused. While there have been efforts to avoid the double-sampling problem for other approaches such as the projected fixpoint methods or Bellman residual minimization (Farahmand et al., 2008), it is still an open question whether the bias of the residual-gradient algorithm with single samples can be removed by similar techniques.

Projected-Fixpoint Methods. The key idea of the projected fixpoint algorithms is to minimize the MSPBE directly by stochastic gradient descent (Sutton et al., 2009) and, therefore, overcome the issue of TD learning which alters the objective function between descent steps.

Sutton et al. (2009) proposed two different stochastic gradient descent techniques. The derivation starts by writing the MSPBE in a different form given by

$$\text{MSPBE}(\boldsymbol{\theta}) = \mathbb{E}[\delta_t\boldsymbol{\phi}_t]^T\mathbb{E}[\boldsymbol{\phi}_t\boldsymbol{\phi}_t^T]^{-1}\mathbb{E}[\delta_t\boldsymbol{\phi}_t]. \tag{18}$$

The proof of this equation is provided in Appendix A, Equation (43). We can write the gradient of Equation (18) as

$$\nabla\,\text{MSPBE}(\boldsymbol{\theta}) = -2\mathbb{E}[(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1})\boldsymbol{\phi}_t^T]\mathbb{E}[\boldsymbol{\phi}_t\boldsymbol{\phi}_t^T]^{-1}\mathbb{E}[\delta_t\boldsymbol{\phi}_t] \tag{19}$$

$$= -2\mathbb{E}[\delta_t\boldsymbol{\phi}_t] + 2\gamma\mathbb{E}[\boldsymbol{\phi}_{t+1}\boldsymbol{\phi}_t^T]\mathbb{E}[\boldsymbol{\phi}_t\boldsymbol{\phi}_t^T]^{-1}\mathbb{E}[\delta_t\boldsymbol{\phi}_t]. \tag{20}$$

The gradient contains a product of expectations of $\boldsymbol{\phi}_{t+1}$ and δ_t in both forms (Equation 19 and 20). As both terms depend on the transition distribution of the MDP, minimizing Equation (18) with stochastic gradient descent again requires two independently drawn samples, as in the residual-gradient algorithm. To circumvent this limitation, a long-term quasi-stationary estimate $\boldsymbol{w}$ of

$$\mathbb{E}[\boldsymbol{\phi}_t\boldsymbol{\phi}_t^T]^{-1}\mathbb{E}[\delta_t\boldsymbol{\phi}_t] = \boldsymbol{(\Phi^T D\Phi)^{-1}\Phi^T D}(T\boldsymbol{V_\theta} - \boldsymbol{V_\theta}) \tag{21}$$

is calculated. To obtain an iterative update for $\boldsymbol{w}$, we realize that the right side of Equation (21) is the solution to the following least-squares problem

$$J(\boldsymbol{w}) = \|\boldsymbol{\Phi}^T\boldsymbol{w} - (T\boldsymbol{V}_\theta - \boldsymbol{V}_\theta)\|_2^2.$$

This least-squares problem can also be solved by stochastic gradient descent with the update rule

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \beta_t(\delta_t - \boldsymbol{\phi}_t^T\boldsymbol{w}_t)\boldsymbol{\phi}_t,$$

and the step-size β_t. Inserting the estimate $\boldsymbol{w}_t$ into Equation (19) and Equation (20) allows us to rewrite the gradient with a single expectation

$$\nabla\,\text{MSPBE}(\boldsymbol{\theta}) = -2\mathbb{E}[(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1})\boldsymbol{\phi}_t]^T\boldsymbol{w}_t \tag{22}$$

$$= -2\mathbb{E}[\delta_t\boldsymbol{\phi}_t] + 2\gamma\mathbb{E}[\boldsymbol{\phi}_{t+1}\boldsymbol{\phi}_t^T]\boldsymbol{w}_t. \tag{23}$$

Minimizing with the gradient of the form of Equation (22) is called the *GTD2 (gradient temporal-difference learning 2)* algorithm with update rule

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1})\boldsymbol{\phi}_t^T\boldsymbol{w}_t,$$

and using the form of Equation (23) yields the *TDC (temporal-difference learning with gradient correction)* algorithm (Sutton et al., 2009)

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t(\delta_t\boldsymbol{\phi}_t - \gamma(\boldsymbol{\phi}_t^T\boldsymbol{w}_t)\boldsymbol{\phi}_{t+1}).$$

As $\boldsymbol{\theta}$ and $\boldsymbol{w}$ are updated at the same time, the choice of step-sizes α_t and β_t are critical for convergence (see also our experiments in Section. 3). Both methods can be understood as a nested version of stochastic gradient descent optimization. TDC is similar to TD learning, but with an additional term to adjust the TD update to approximate the real gradient of MSPBE. The right side of Figure 7 shows this corrections and compares both stochastic approximations to descent following the true gradient. Both algorithms minimize the MSPBE but show different speeds of convergence, as we will also illustrate in the discussion of experimental results in Section 3.

The predecessor of the GTD2 algorithm is the GTD algorithm (Sutton et al., 2008). It minimizes the NEU cost function from Equation (12) by stochastic gradient descent. The gradient of NEU is given by

$$\nabla \mathrm{NEU}(\theta) = -2\mathbb{E}[(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1})\boldsymbol{\phi}_t^T]\mathbb{E}[\delta_t\boldsymbol{\phi}_t].$$

One of the two expectations needs to be estimated by a quasi-stationary estimate in analogy to the other projected fixpoint methods. Hence, the term $E[\delta_t\boldsymbol{\phi}_t]$ is replaced by $\boldsymbol{u}$ which is updated incrementally by

$$\boldsymbol{u}_{t+1} = \boldsymbol{u}_t + \beta_t(\boldsymbol{\phi}_t^T\delta_t - \boldsymbol{u}_t) = (1-\beta_t)\boldsymbol{u}_t + \beta_t\boldsymbol{\phi}_t^T\delta_t.$$

The updates for $\boldsymbol{\theta}$ of GTD are then given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1})\boldsymbol{\phi}_t^T\boldsymbol{u}_t.$$

The update rule for GTD is similar to GTD2 but the quasi stationary estimates $\boldsymbol{w}$ and $\boldsymbol{u}$ are different. While GTD2 searches for the best linear approximation $\boldsymbol{\phi}_t^T\boldsymbol{w}$ of $\mathbb{E}[\delta_t]$, GTD tries to approximate $\mathbb{E}[\delta_t\boldsymbol{\phi}_t]$ with $\boldsymbol{u}$. As shown by Sutton et al. (2009) and our experiments, GTD2 converges faster and should be preferred over GTD.

All gradient-based temporal-difference methods only require sums and products of vectors of length n to update the parameters. Thus, they run in $O(n)$ time per update. The initial value of $\boldsymbol{\theta}$ has a tremendous influence on the convergence speed of gradient-based methods. While other approaches such as LSTD do not require an initial values, the gradient based approaches can benefit from good parameter guesses, which are available in numerous applications. For example, the parameter vector learned in previous steps of policy iteration can be used as initial guesses to speed up learning. Fixed-Point Kalman Filtering (FPKF) proposed by Choi and Roy (2006) is a descent method, that has a close relationship to TD learning. Yet, as it is motivated by least-squares minimization, we present it in the next section.

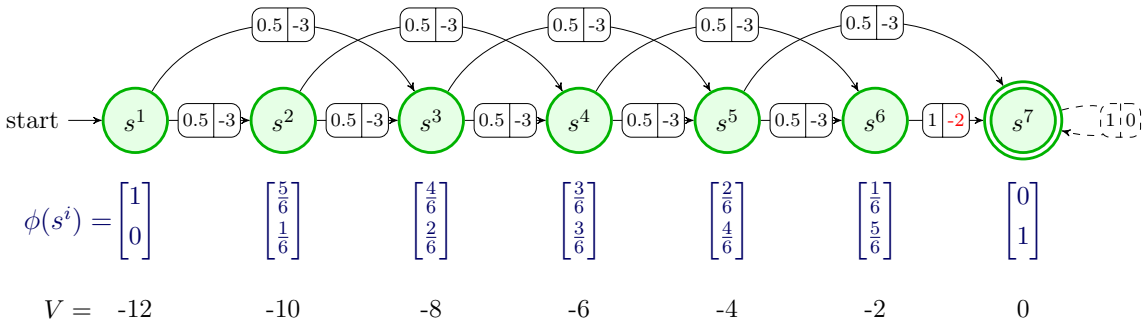


Figure 6: 7-State Boyan Chain MDP from Boyan (2002). Each transition is specified by a probability (left number) and a reward (right number). There are no actions to chose for the agent. The features are linearly increasing / decreasing from left to right and are capable of representing the true value function with parameters $\boldsymbol{\theta} = [-12, 0]^T$.

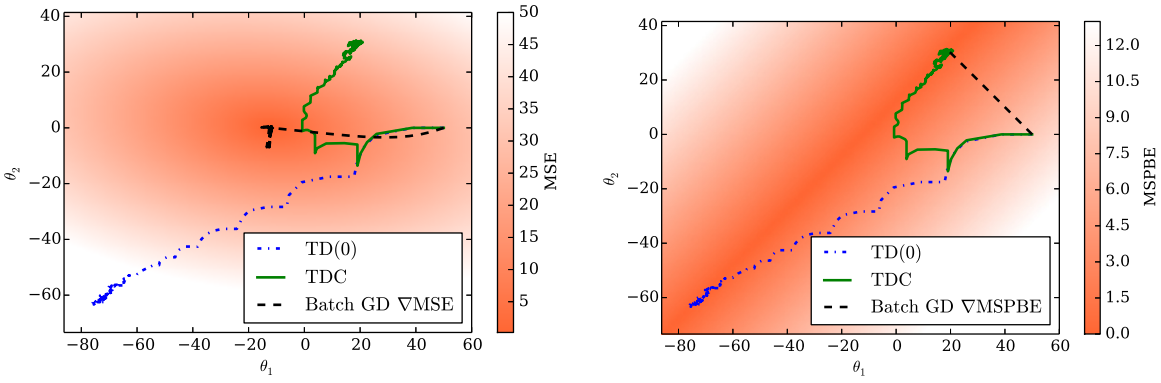


Figure 7: Comparison of Gradient Descent and TD learning for the MDP of Figure 6 with $\gamma = 1$: The left plot shows the mean squared error. Ideally, we want to find parameters that minimize this objective, however, TD(0) and TDC only find a minimum of the MSPBE (Eq. 9, right plot). The dashed lines show the parameter iterates of batch gradient descent of the respective cost functions. Stochastic gradient methods such as TD and TDC are slower since they can only update $\boldsymbol{\theta}_1$ for samples from the beginning of an episode and $\boldsymbol{\theta}_2$ for samples from the end (cf. the features from Figure 6). Comparison of both plots shows, that a fixpoint of MSPBE can give arbitrarily bad results for MSE, as the problem is not discounted and the guarantees of Section 2.1 do not hold. The MSPBE and MSBE measures only compare the difference of the value of a state and its successor's value. Hence, all parameters which yield differences of 2 and arbitrary value of the terminal state are optimal.

2.2.2 Least-Squares Approaches

Least-squares approaches use all previously observed transitions to determine either the value function directly in one step or an update of the value function. All considered objective functions (cf. Section 2.1) have the form of a standard linear regression problem

$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|_{\boldsymbol{U}}^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^T \boldsymbol{U} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}),$$

with respect to the (semi-)norm induced by a positive semi-definite matrix $\boldsymbol{U} \in \mathbb{R}^{k \times k}$. While the targets are denoted by $\boldsymbol{y} \in \mathbb{R}^k$, $\boldsymbol{X} \in \mathbb{R}^{k \times n}$ is the matrix consisting of rows of basis vectors. Setting the gradient of the objective function with respect to $\boldsymbol{\theta}$ to 0 yields the closed-form least-squares solution

$$\boldsymbol{\theta}^* = (\boldsymbol{X}^T \boldsymbol{U} \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{U} \boldsymbol{y}.$$

Least-Squares Temporal-Difference Learning. The most prominent least-squares method for policy evaluation is least-squares temporal-difference (LSTD) learning (Bradtke and Barto, 1996; Boyan, 2002). LSTD uses the MSPBE as objective function. The least-squares solution of the MSPBE from Equation (10) is given by

$$\boldsymbol{\theta} = \underbrace{(\boldsymbol{\Phi}^T \boldsymbol{D} (\boldsymbol{\Phi} - \gamma \boldsymbol{P} \boldsymbol{\Phi}))}_{\boldsymbol{A}}^{-1} \underbrace{\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{R}}_{\boldsymbol{b}}. \tag{24}$$

During the derivation of this solution (see Appendix A) many terms cancel out, including the ones which are connected to the double-sampling problem—in contrast to the analytical solution for minimizing the MSBE shown in Equation (31). Alternatively, LSTD can be derived by considering the analytical solution of the OPE problem from the OPE–FPE formulation (Equations 13 and 14) given by

$$\begin{aligned}\boldsymbol{\theta} =& (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} (\boldsymbol{R} + \gamma \boldsymbol{P} \boldsymbol{\Phi} \boldsymbol{\omega}) \\ \underset{\boldsymbol{\omega} = \boldsymbol{\theta}}{=}& (\boldsymbol{\Phi}^T \boldsymbol{D} (\boldsymbol{\Phi} - \gamma \boldsymbol{P} \boldsymbol{\Phi}))^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{R}.\end{aligned}$$

The LSTD solution in the second line is obtained by inserting the solution of the FPE problem, $\boldsymbol{\omega} = \boldsymbol{\theta}$, into the first line and re-ordering the terms.

LSTD explicitly estimates $\boldsymbol{A} = \boldsymbol{\Phi}^T \boldsymbol{D} (\boldsymbol{\Phi} - \gamma \boldsymbol{P} \boldsymbol{\Phi})$ and $\boldsymbol{b} = \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{R}$ and then determines $\boldsymbol{\theta} = \boldsymbol{A}^{-1} \boldsymbol{b}$ robustly (for example with singular value decomposition). The estimates $\boldsymbol{A}_t, \boldsymbol{b}_t$ at time t can be computed iteratively by

$$\boldsymbol{A}_{t+1} = \boldsymbol{A}_t + \boldsymbol{\phi}_t [\boldsymbol{\phi}_t - \gamma \boldsymbol{\phi}_{t+1}]^T, \tag{25}$$

$$\boldsymbol{b}_{t+1} = \boldsymbol{b}_t + \boldsymbol{\phi}_t r_t, \tag{26}$$

and converge to $\boldsymbol{A}, \boldsymbol{b}$ (Nedic and Bertsekas, 2003) for $t \to \infty$. For calculating $\boldsymbol{\theta}_t$ we need to invert a $n \times n$ matrix. This computational cost can be reduced from $O(n^3)$ to $O(n^2)$ by updating $\boldsymbol{A}_t^{-1}$ directly and maintaining an estimate $\boldsymbol{\theta}_t = \boldsymbol{A}_t^{-1} \boldsymbol{b}_t$ (Nedic and Bertsekas, 2003; Yu, 2010). The direct update of $\boldsymbol{A}_t^{-1}$ can be derived using the Sherman-Morrison formula

$$(\boldsymbol{A}_t + \boldsymbol{u}\boldsymbol{v}^T)^{-1} = \boldsymbol{A}_t^{-1} - \frac{\boldsymbol{A}_t^{-1} \boldsymbol{u} \boldsymbol{v}^T \boldsymbol{A}_t^{-1}}{1 + \boldsymbol{v}^T \boldsymbol{A}_t^{-1} \boldsymbol{u}} \tag{27}$$

with vectors $\boldsymbol{u} := \boldsymbol{\phi}_t$ and $\boldsymbol{v} := \boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1}$. The resulting recursive LSTD algorithm is listed in Appendix C in a more general form with eligibility traces and off-policy weights (for the basic form set $\lambda = 0$ and $\rho_t = 1$).

An initial guess $\boldsymbol{A}_0^{-1}$ has to be chosen by hand. $\boldsymbol{A}_0^{-1}$ corresponds to the prior belief of $\boldsymbol{A}^{-1}$ and is ideally the inverse of the null-matrix. In practice, the choice $\boldsymbol{A}_0^{-1} = \epsilon\boldsymbol{I}$ with $\epsilon \gg 0$ works well. Small values for ϵ act as a regularizer on $\boldsymbol{\theta}$.

Interestingly, LSTD has a model-based reinforcement learning interpretation. For lookup table representations, the matrix $\boldsymbol{A}_t$ contains an empirical model of the transition probabilities. To see that, we write $\boldsymbol{A}_t$ and $\boldsymbol{b}_t$ as

$$\boldsymbol{A}_t = \boldsymbol{N} - \boldsymbol{C} = \boldsymbol{N}(\boldsymbol{I} - \gamma\hat{\boldsymbol{P}}), \quad \boldsymbol{b}_t = \boldsymbol{N}\hat{\boldsymbol{R}},$$

where $\boldsymbol{N}$ is a diagonal matrix containing the state visit counts up to time step t. The elements C_{ij} of matrix $\boldsymbol{C}$ contain the number of times a transition from state i to state j has been observed. The matrix $\hat{\boldsymbol{P}} = \gamma^{-1}\boldsymbol{N}^{-1}\boldsymbol{C}$ denotes the estimated transition probabilities and $\hat{\boldsymbol{R}}_i$ denotes the average observed reward when being in state i. Note that, in this case, the LSTD solution

$$\boldsymbol{\theta}^* = \left(\boldsymbol{N}\left(\boldsymbol{I} - \gamma\hat{\boldsymbol{P}}\right)\right)^{-1}\boldsymbol{N}\hat{\boldsymbol{R}} = \left(\boldsymbol{I} - \gamma\hat{\boldsymbol{P}}\right)^{-1}\hat{\boldsymbol{R}}$$

exactly corresponds to model based policy evaluation with an estimated model. For approximate feature spaces, the equivalence to model-based estimation is lost, but the intuition remains the same. A more detailed analysis of this connection can be found in the work of Boyan (2002) and Parr et al. (2008)

Least-Squares Policy Evaluation. The least-squares policy evaluation (LSPE) algorithm proposed by Nedic and Bertsekas (2003) shares similarities with TD learning and the LSTD method as it combines the idea of least-squares solutions and gradient descent steps. This procedure can again be formalized with the nested OPE-FPE problem from Equations (13) and (14). First, LSPE solves the operator problem

$$\boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{\Phi}\boldsymbol{\theta} - T\boldsymbol{\Phi}\boldsymbol{\omega}_t\|_{\boldsymbol{D}}^2 \tag{28}$$

in closed form with the least-squares solution. Then, it decreases the fixpoint error by performing a step in the direction of the new $\boldsymbol{\theta}_{t+1}$

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \alpha_t(\boldsymbol{\theta}_{t+1} - \boldsymbol{\omega}_t), \tag{29}$$

where $\alpha_t \in (0, 1]$ is a predefined step size. The vector $\boldsymbol{\omega}_t$ is the output of the algorithm at time-step t, that is, the parameter estimate for the value function. In practice, the step-sizes are large in comparison to stochastic gradient approaches and can often be set to 1.

The solution of the LSPE problem from Equation (28) is given by

$$\boldsymbol{\theta}_{t+1} = \underbrace{(\boldsymbol{\Phi}^T\boldsymbol{D}\boldsymbol{\Phi})^{-1}}_{\boldsymbol{M}}\boldsymbol{\Phi}^T\boldsymbol{D}(\boldsymbol{R} + \gamma\boldsymbol{\Phi}'\boldsymbol{\omega}_t), \tag{30}$$

where $\boldsymbol{\Phi}'$ is the matrix containing the features of the successor states, that is, $\boldsymbol{\Phi}' = \boldsymbol{P}^\pi \boldsymbol{\Phi}$. The current estimate $\boldsymbol{M}_t$ of $(\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1}$ can again be updated recursively with the Sherman-Morrison formula from Equation (27) similar to before, which yields the update rule of LSPE summarized in Algorithm 7 in Appendix C. As LSPE solves the nested OPE-FPE problem, it also converges to the MSPBE fixpoint (Nedic and Bertsekas, 2003). Hence, LSPE and LSTD find the same solution, but LSPE calculates the value function recursively using least-squares solutions of the OPE problem while LSTD acquires the value function directly by solving both, OPE and FPE, problems in closed form. LSPE allows adapting the step-sizes α_t of the updates and using prior knowledge for the initial estimate of $\boldsymbol{\omega}_0$, which serves as a form of regularization. Therefore, LSPE does not aim for the minimum of the MSPBE approximated by samples up to the current timestep, it instead refines the previous estimates. Such behavior may avoid numerical issues of LSTD and is less prone to over-fitting.

Fixed-Point Kalman Filtering. Kalman filtering is a well known second-order alternative to stochastic gradient descent. Choi and Roy (2006) applied the Kalman filter to temporal-difference learning, which resulted in the Fixed-Point Kalman Filtering (FPKF) algorithm.

As in TD learning, FPKF solves the nested optimization problem from Equations (13) and (14) and hence finds the minimum of the MSPBE objective function. However, instead of stochastic gradient descent, FPKF performs a second order update by multiplying the standard TD learning update with the inverse of the Hessian H_t of the operator error given in Equation (13). FPKF can therefore be also understood as an approximate Newton-method on the OPE problem. The Hessian H_t is given by the second derivative of the OPE

$$\boldsymbol{H}_t = \frac{1}{t} \sum_{i=1}^{t} \boldsymbol{\phi}_i \boldsymbol{\phi}_i^T.$$

Note that the Hessian is calculated from the whole data set $i = 1, \dots, t$ up to the current time step and does not depend on the parameters $\boldsymbol{\theta}$. The update rule of FPKF is thus given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \boldsymbol{H}_t^{-1} \boldsymbol{\phi}_t \delta_t.$$

This update rule can also be derived directly from the Kalman filter updates if α_t is set to $1/t$. See Figure 8 for a graphical model illustrating the Kalman Filter assumptions (e.g., the definition of the evolution and observation function). For small values of t, the matrix $\boldsymbol{H}_t$ becomes singular. In this case $\boldsymbol{H}_t$ needs to be regularized or the pseudo-inverse of $\boldsymbol{H}_t$ has to be used. As FPKF is a second order method, it typically converges with fewer iterations than TD but comes with additional price of estimating $\boldsymbol{H}_t^{-1}$. Analogously to the derivation of LSPE, $\boldsymbol{H}_t^{-1}$ can updated directly in $O(n^2)$ which yields the recursive parameter updates of FPKF shown in Algorithm 9 (adapted from the work of Scherrer and Geist, 2011 and Geist and Scherrer, 2013).

The step-size $\alpha_t = 1/t$ of Kalman filtering basically assumes a stationary regression problem, where all targets $r_i + \gamma \boldsymbol{\phi}_{i+1}^T \boldsymbol{\theta}_i$ are equally important. However, it is beneficial to give later targets more weight, as the parameter estimates are getting more accurate and therefore the targets $r_i + \gamma \phi_{i+1}^T \boldsymbol{\theta}_i$ are becoming more reliable. Such increased influence of

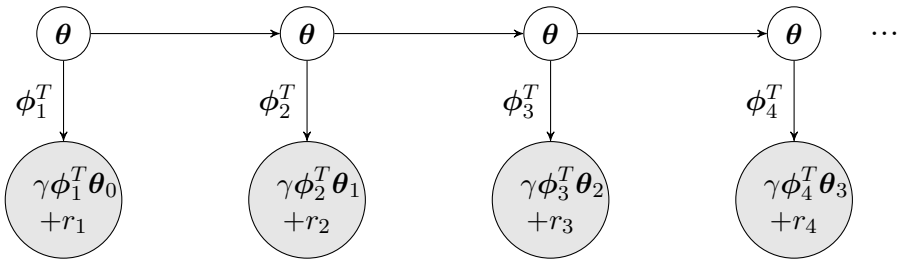


Figure 8: Illustration of the model assumptions of Fixed-Point Kalman Filtering. FPKF aims at estimating the value of the hidden variable $\boldsymbol{\theta}$ assuming a noise-free transition function (stationary environment, variable is constant over time). The main idea of FPKF is to use estimates of previous timesteps to compute the outputs $r_i + \gamma\boldsymbol{\phi}_i^T\boldsymbol{\theta}_i$ and treat them as fixed and observed in later timesteps. This assumption makes FPKF essentially different from KTD (cf. Figure 9), which only considers r_i as observed.

recent time-steps can be achieved by using a step-size α_t which decreases slower than $1/t$, for example, $a/(a+t)$ for some large a.

Bellman Residual Minimization. Bellman residual minimization (BRM) was one of the first approaches for approximate policy evaluation proposed by Schweitzer and Seidmann (1985). It calculates the least-squares solution of the MSBE given by

$$\boldsymbol{\theta} = \underbrace{(\boldsymbol{\Delta\Phi}^T\boldsymbol{D}\boldsymbol{\Delta\Phi})}_{\boldsymbol{F}}^{-1}\underbrace{\boldsymbol{\Delta\Phi}^T\boldsymbol{D}\boldsymbol{R}}_{\boldsymbol{g}}, \tag{31}$$

where $\boldsymbol{\Delta\Phi} = \boldsymbol{\Phi} - \gamma\boldsymbol{P}^\pi\boldsymbol{\Phi}$ denotes the difference of the state features and the expected next state features discounted by γ. Again, the matrices $\boldsymbol{F}$ and $\boldsymbol{g}$ can be estimated by samples, similar to $\boldsymbol{A}$ and $\boldsymbol{b}$ of LSTD, that is,

$$\boldsymbol{F}_t = \sum_{k=0}^{t}(\boldsymbol{\phi}_k - \gamma\boldsymbol{\phi}'_{k+1})(\boldsymbol{\phi}_k - \gamma\boldsymbol{\phi}''_{k+1})^T, \quad \boldsymbol{g}_t = \sum_{k=0}^{t}(\boldsymbol{\phi}_k - \gamma\boldsymbol{\phi}'_{k+1})r''_k.$$

However, as the residual-gradient algorithm, BRM suffers from the double-sampling problem because $\boldsymbol{F}_t$ contains the product $\boldsymbol{\phi}_{k+1}\boldsymbol{\phi}_{k+1}^T$ of the features of the next state and $\boldsymbol{g}_t$ the product $\boldsymbol{\phi}_{k+1}r_k$ (see Section 2.1 and the discussion of the residual-gradient algorithm in Section 2.2.1). It therefore minimizes the MSTDE if we use one successor state sample, that is, set $\boldsymbol{\phi}'_{k+1} = \boldsymbol{\phi}''_{k+1}$. To converge to a minimum of the MSBE, we have to use two independent samples s'_{k+1}, r'_k and s''_{k+1} r''_k. For this reason, the BRM algorithm can only be employed for either a finite state space where we can visit each state multiple times or if we know the model of the MDP. See Algorithm 10 in Appendix C for the recursive update rules with double samples. For updates with a single sample see Algorithm 11, which already includes eligibility traces (from Scherrer and Geist, 2011; Geist and Scherrer, 2013, see also Section 2.4.1). If we compare the least-squares solutions for the MSPBE and the MSBE, we

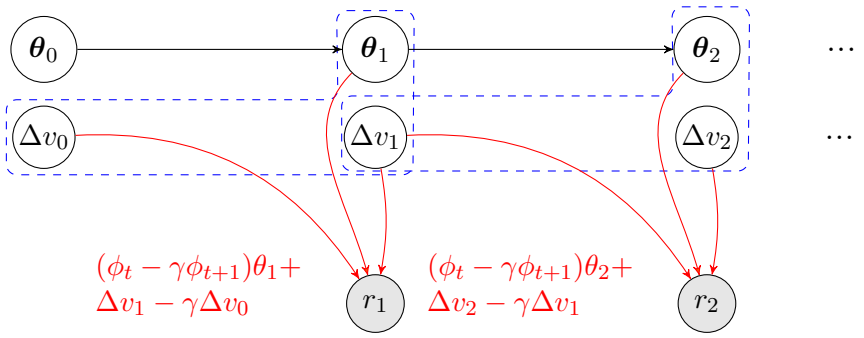


Figure 9: Graphical Model of Kalman TD learning and Gaussian-process TD learning for linearly parameterized value functions. Both approaches assume that a Gaussian process generates the random variables and linear function approximation $v_t = \boldsymbol{\theta}_t^T \phi_t$ is used. KTD aims to track the hidden state (blue dashed set of variables) at each time step given the reward observation generated by the relationship (bend arrows, in red) of the Bellman equation. While GPTD assumed $\boldsymbol{\theta}$ to be constant over time, that is, $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t$, KTD allows changing parameters.

can see that the product of $\mathbf{\Delta\Phi}^T\mathbf{\Delta\Phi}$ cancels out for the MSPBE due to the projection of the MSBE in the feature space and the MSPBE can subsequently avoid the double-sampling problem.

2.2.3 Probabilistic Models

While gradient-based and least-squares approaches are motivated directly from an optimization point of view, probabilistic methods take a different route. They build a probabilistic model and infer value function parameters which are most likely, given the observations. These methods not only yield parameter estimates that optimize a cost function, but also provide a measure of uncertainty of these estimates. Especially in policy iteration, this information can be very helpful to decide whether more observations are necessary or the value function estimate is sufficiently reliable to improve the policy.

Gaussian-process temporal-difference learning (GPTD) by Engel et al. (2003, 2005) assumes that the rewards r_t as well as the unknown true values of the observed states $v_t = V(s_t)$ are random variables generated by a Gaussian process. The Bellman Equation (2) specifies the relation between the variables

$$\tilde{v}_t := v_t + \Delta v_t = r_t + \gamma(v_{t+1} + \Delta v_{t+1}), \tag{32}$$

where $\tilde{v}_t = \sum_{k=t}^{\infty} \gamma^{k-t} r_t$ is the future discounted reward of the current state s_t in the particular trajectory. The difference between $\tilde{v}_t$ and the average future discounted reward v_t is denoted by Δv_t and originates from the uncertainty in the system, that is, the policy π and the state transition dynamics $\mathcal{P}$. Please see Figure 9 for a graphical model illustrating the dependencies of the random variables. While $\Delta v_t \sim \mathcal{N}(0, \sigma^2)$ always has mean zero, we have to set its variances σ_t a-priori based on our belief of π and $\mathcal{P}$. The covariance $\mathbf{\Sigma}_t$ of all

Δv_i for $i = 1, \dots, t$ is a band matrix with bandwidth 2 as the noise terms of two subsequent time steps are correlated.

We consider again a linear approximation of the value function, that is, $v_t = \boldsymbol{\phi}_t^T \boldsymbol{\theta}$. Prior knowledge about the parameter $\boldsymbol{\theta}$ can be incorporated in the prior $p(\boldsymbol{\theta})$, which acts as a regularizer and is usually set to $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. GPTD infers the mean and covariance of the Gaussian distribution of $\boldsymbol{\theta}$ given the observations $r_0, \dots, r_t$ and our beliefs. The mean value corresponds to the maximum a-posteriori prediction and is equivalent to finding the solution of the regularized linear regression problem

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta}} \|\Delta \boldsymbol{\Phi}_t \boldsymbol{\theta} - \boldsymbol{r}_t\|^2_{\boldsymbol{\Sigma}_t^{-1}} + \|\boldsymbol{\theta}\|^2, \tag{33}$$

where $\boldsymbol{r}_t = [r_0, r_1, \dots, r_t]^T$ is a vector containing all observed rewards. The matrix $\Delta \boldsymbol{\Phi}_t = [\Delta \boldsymbol{\phi}_0, \dots, \Delta \boldsymbol{\phi}_t]^T$ is the difference of features of all transitions with $\Delta \boldsymbol{\phi}_t = \boldsymbol{\phi}_t - \gamma \boldsymbol{\phi}_{t+1}$. The solution of this problem can be formulated as

$$\boldsymbol{\theta}_t = \left(\Delta \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_t^{-1} \Delta \boldsymbol{\Phi}_t^T + \boldsymbol{I}\right)^{-1} \Delta \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_t^{-1} \boldsymbol{r}_t. \tag{34}$$

At first glance, the solution is similar to the MSBE least-squares solution. However, the noise terms are often highly correlated and, hence, $\boldsymbol{\Sigma}_t^{-1}$ is not a diagonal matrix. We can transform the regression problem in Equation (33) into a standard regularized least-squares problem with i.i.d. sampled data points by a whitening transformation (for details see Appendix B). We then see that the whitening transforms the reward vector $\boldsymbol{r}_t$ into the vector of the long term returns $\boldsymbol{R}_t$ where the hth elements corresponds to $(\boldsymbol{R}_t)_h = \sum_{k=h}^{t} \gamma^{k-h} r_k$. Consequently, the mean prediction of GPTD is equivalent to regularized Monte-Carlo value-function estimation (cf. Section 1.2) and minimizes the MSE from Equation (4) in the limit of infinite observations. For finite amount of data, the prior on $\boldsymbol{\theta}$ compensates the high variance problem of Monte-Carlo estimation, but may also slow down the learning process.

The quantity in Equation (34) can be computed incrementally without storing $\Delta \boldsymbol{\Phi}_t$ explicitly or inverting a $n \times n$-matrix at every timestep by a recursive algorithm shown in Algorithm 12. Its full derivation can be found in Appendix 2.1 of Engel (2005). The most expensive step involves a matrix product of the covariance matrix $\boldsymbol{P}_t \in \mathbb{R}^{n \times n}$ of $\boldsymbol{\theta}_t$ and $\Delta \boldsymbol{\phi}_{t+1}$. Thus, GPTD has a runtime complexity of $O(n^2)$.

Kalman temporal-difference learning (KTD) by Geist and Pietquin (2010) is another probabilistic model very similar to GPTD. While it is based on the same assumptions of Gaussian distributed random variables, it approaches the value estimation problem from a filtering or signal processing perspective. KTD uses a Kalman Filter to track a hidden state, which changes over time and generates the observed rewards at every timestep. The state consists of the parameter to estimate $\boldsymbol{\theta}_t$ and the value difference variables $\Delta v_t, \Delta v_{t-1}$ with $\Delta v_t = \tilde{v}_t - \boldsymbol{\phi}_t \boldsymbol{\theta}_t$ of the current and last timestep. As in GPTD, the rewards are generated from this state with Equation (32) resulting in the linear observation function g

$$r_t = g(\boldsymbol{\theta}_t, \Delta v_t, \Delta v_{t+1}) = [\boldsymbol{\phi}_t - \gamma \boldsymbol{\phi}_{t+1}] \boldsymbol{\theta}_t + \Delta v_t - \gamma \Delta v_{t+1}.$$

KTD does not necessarily assume that a unique single parameter $\boldsymbol{\theta}$ has created all rewards, but allows the parameter to change over time (as if the environment is non-stationary). More precisely, $\boldsymbol{\theta}_{t+1} \sim \mathcal{N}(\boldsymbol{\theta}_t, \boldsymbol{\Sigma}_\theta)$ is modeled as a random walk. As we focus on stationary environments, we can set $\boldsymbol{\Sigma}_\theta = \mathbf{0}$ to enforce constant parameters and faster convergence.

In this case, KTD and GPTD are identical algorithms for linear value function parametrization (cf. Algorithm 12). The graphical model in Figure 9 illustrates the similar assumptions of both approaches. Besides KTD's ability to deal with non-stationary environments, KTD and GPTD differ mostly in the way they handle value functions that are non-linear in the feature space. KTD relies on the unscented transform (a deterministic sample approximation for nonlinear observation functions), while GPTD avoids explicit function parametrization assumptions with kernels (cf. Section 2.3). Depending on the specific application and available domain knowledge, either a well-working kernel or a specific nonlinear parametrization is easier to chose.

Both probabilistic approaches share the benefit of not only providing a parameter estimate but also an uncertainty measure on it. However, as they optimize the mean squared error similar to Monte Carlo value-function estimation, their estimates may suffer from higher variance. The long-term memory effect originating from the consideration of all future rewards also prevents off-policy learning as discussed by Geist and Pietquin (2010, Section 4.3.2) and Engel (2005).

2.3 Feature Handling

The feature representation $\boldsymbol{\phi}$ of the states has a tremendous influence, not only on the quality of the final value estimate but also on convergence speed. We aim for features that can represent the true value function accurately and are as concise as possible to reduce computational costs and the effects of over-fitting. Many commonly used feature functions are only locally active. Their components are basis functions which have high values in a specific region of the state space and low ones elsewhere. For example, cerebellar model articulation controllers (CMAC) cover the state space with multiple overlapping tilings (Albus, 1975), also known as tile-coding. The feature function consists of binary indicator functions for each tile. Alternatively, smoother value functions can be obtained with radial basis functions. Such bases work well in practice, but often only if they are normalized such that $\|\phi(s)\|_1 = 1$ for all states $s \in \mathcal{S}$ as discussed by Kretchmar and Anderson (1997). The performance of many algorithms, including the regularization methods discussed in Section 2.3.2, can be improved by using normalized features with zero mean and unit variance.

Local function approximators are limited to small-scale settings as they suffer from the curse of dimensionality similar to exact state representations (cf. Section 1.2). When the number of state dimensions increases, the number of features explodes exponentially and so does the amount of data required to learn the value function. Therefore, recent work has focused on facilitating the search for well-working feature functions. These efforts follow two principled approaches: (1) features are either generated automatically from the observed data or (2) the learning algorithms are adapted to cope with huge numbers of features efficiently in terms of data and computation time. We briefly review the advances in both directions in the following two sections.

2.3.1 Automatic Feature Generation

Kernel-based value function estimators represent the value of a state in terms of the similarity of that state to previously observed ones, that is, at each time step the similarity to current state is added as an additional feature. A well chosen kernel, that is, the distance or similarity

measure, is crucial for the performance of kernel-based approaches, as well as an adequate sparsification technique to prevent the number of features to grow unboundedly.

GPTD (Engel et al., 2003) and LSTD (Xu et al., 2005, known as Kernelized LSTD, KLSTD) have been extended to use kernelized value functions. A similar approach was proposed in the work of Rasmussen and Kuss (2003) where a kernel-based Gaussian process is used for approximating value functions based on the Bellman Equation (2). This approach, KLSTD and GPTD were unified in a model-based framework for kernelized value function approximation by Taylor and Parr (2009). Jung and Polani (2006) introduced an alternative online algorithm originating from least-squares support-vector machines to obtain the GPTD value function estimate; however, it is limited to MDPs with deterministic transitions.

An alternative to kernel methods based on spectral learning was presented by Mahadevan and Maggioni (2007). The authors proposed to build a graph-representation of the MDP from the observations and chose features based on the eigenvector of the Graph-Laplacian. Compared to location-based features such as radial basis functions, this graph-based technique can handle discontinuities in the value-function more accurately. In contrast, Menache et al. (2005) assumes a fixed class of features, for example, RBFs, and optimizes only the free parameters (e.g., the basis function widths) by gradient descent or by using the cross-entropy optimization method (De Boer et al., 2010). Keller et al. (2006) uses neighborhood component analysis, a dimensionality reduction techniques for labeled data, to project the high-dimensional state space to a lower dimensional feature representation. They take the observed Bellman errors from Equation (7) as labels to obtain features that are most expressive for the value function. The approaches of Parr et al. (2007), Painter-Wakefield and Parr (2012a) and Geramifard et al. (2013, 2011) are based on the orthogonal matching principle (Pati et al., 1993) and incrementally add features which have high correlation with the temporal-difference error. The intuition is that those additional features enable the algorithms to further reduce the temporal-difference error.

2.3.2 Feature Selection by Regularization

Value function estimators face several challenges when the feature space is high dimensional. First, the computational costs may become unacceptably large. Second, a large number of noise-like features deteriorates the estimation quality due to numerical instabilities and, finally, the amount of samples required for a reliable estimate grows prohibitively. The issues are particularly severe for least-squares approaches which are computationally more involved and tend to over-fit when the number of observed transitions is lower than the dimensionality of the features.

The problem of computational costs for second order methods can be addressed by calculating the second order updates incrementally. For example, the parameter update of incremental LSTD (iLSTD proposed by Geramifard et al., 2006a,b) is linear in the total number of features ($O(n)$ instead of $O(n^2)$ for standard LSTD) if only a very small number of features is non-zero in each state. Most location based features such as CMAC or fixed-horizon radial basis functions fulfill this condition.

Information theoretic approaches which compress extensive feature representations are prominent tools in machine learning for reducing the dimensionality of a problem. Yet, these methods are often computationally very demanding which limits their use in online

	Formulation		Optimization Technique
LSTD with ℓ_2	$f(\boldsymbol{\theta}) \propto \|\boldsymbol{\theta}\|_2^2$	$g(\boldsymbol{\omega}) = 0$	closed form solution (Bradtke and Barto, 1996)
LSTD with ℓ_2,ℓ_2	$f(\boldsymbol{\theta}) \propto \|\boldsymbol{\theta}\|_2^2$	$g(\boldsymbol{\omega}) \propto \|\boldsymbol{\omega}\|_2^2$	closed form solution (Hoffman et al., 2011)
LARS-TD	$f(\boldsymbol{\theta}) \propto \|\boldsymbol{\theta}\|_1$	$g(\boldsymbol{\omega}) = 0$	custom LARS–like solver (Kolter and Ng, 2009)
LC-TD	$f(\boldsymbol{\theta}) \propto \|\boldsymbol{\theta}\|_1$	$g(\boldsymbol{\omega}) = 0$	standard LCP solvers (Johns et al., 2010)
ℓ_1-PBR	$f(\boldsymbol{\theta}) = 0$ (*)	$g(\boldsymbol{\omega}) \propto \|\boldsymbol{\omega}\|_1$	standard Lasso solvers (Geist and Scherrer, 2011)
LSTD with ℓ_2,ℓ_1	$f(\boldsymbol{\theta}) \propto \|\boldsymbol{\theta}\|_2^2$	$g(\boldsymbol{\omega}) \propto \|\boldsymbol{\omega}\|_1$	standard Lasso solvers (Hoffman et al., 2011)
Laplacian-based reg. LSTD	$f(\boldsymbol{\theta}) \propto \|\boldsymbol{L}\boldsymbol{\Phi}_t\boldsymbol{\theta}\|_2^2$	$g(\boldsymbol{\omega}) = 0$	closed form solution (Geist et al., 2012)
LSTD-ℓ_1	$\min t\|\boldsymbol{A}\boldsymbol{\theta} - \boldsymbol{b}\|_2^2 + \mu\|\boldsymbol{\theta}\|_1$		standard Lasso solvers (Pires, 2011)
D-LSTD	$\min \|\boldsymbol{\theta}\|_1$ s.t. $t\|\boldsymbol{A}\boldsymbol{\theta} - \boldsymbol{b}\|_\infty \leq \mu$		standard LP solvers (Geist et al., 2012)

Table 2: Comparison of Regularization Schemes for LSTD. f and g are the regularization terms in the nested problem formulation of LSTD (Equations 2.3.2 and 2.3.2). Parameters μ control the regularization strength. (*) ℓ_1-PBR actually assumes a small ℓ_2 regularization on the operator problem if the estimate of $\boldsymbol{\Phi}^T\boldsymbol{D}\boldsymbol{\Phi}$ is singular, which is usually the case for $t < m$.

reinforcement learning. Some information theoretic approaches are equivalent to a special form of regularization. Regularization is a standard way to avoid over-fitting by adding punishment terms for large parameters $\boldsymbol{\theta}$. The regularization point of view often leads to computationally cheaper algorithms compared to information theory. Hence, there has been increasing interest in adding different regularization terms to LSTD and similar algorithms (cf. Table 2). As in supervised learning, the most common types of regularization terms are ℓ_1 and ℓ_2-regularization, which penalize large ℓ_1 respective ℓ_2 norms of the parameter vector. While ℓ_2-regularization still allows closed form solutions, it becomes problematic when there are only very few informative features and a high number of noise-like features. Regularizing with ℓ_2-terms usually yields solutions with small but non-zero parameters in each dimension, which have low quality when there are many noise-like features. ℓ_1-regularization on the other hand prevents closed form solutions, but is known to induce sparsity for the resulting estimate of $\boldsymbol{\theta}$ where only few entries are different from zero. Hence, ℓ_1-regularization implic-

itly performs a feature selection and can cope well with many irrelevant features. Therefore, it is well suited for cases where the number of features exceeds the number of samples.

Most regularization methods are derived from the nested OPE-FPE optimization formulation of LSTD in Equations (13)–(14) where the regularization term is added either to the FPE problem, to the OPE problem or in both problems[3]

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{V}_{\boldsymbol{\theta}} - T\boldsymbol{V}_{\hat{\boldsymbol{\omega}}}\|_{\boldsymbol{D}}^2 + \frac{1}{t} f(\boldsymbol{\theta}) \quad \text{and}$$
$$\hat{\boldsymbol{\omega}} = \arg\min_{\boldsymbol{\omega}} \|\boldsymbol{\Phi}(\hat{\boldsymbol{\theta}} - \boldsymbol{\omega})\|_{\boldsymbol{D}}^2 + \frac{1}{t} g(\boldsymbol{\omega}).$$

Using the regularization term $f(\boldsymbol{\theta})$ corresponds to regularization before setting the fixpoint solution in the OPE problem while enabling $g(\boldsymbol{\omega})$ regularizes after employing the fixpoint solution. Using an ℓ_2-penalty in $f(\boldsymbol{\theta})$, that is, $f(\boldsymbol{\theta}) = \beta_f \|\boldsymbol{\theta}\|_2^2$ yields the solution $\hat{\boldsymbol{\theta}} = (\boldsymbol{A} + \beta_f t^{-1} \boldsymbol{I})^{-1} \boldsymbol{b}$. This form of regularization is often considered as the standard regularization approach for LSTD, since it is equivalent to initializing $\boldsymbol{M}_0 = \boldsymbol{A}_0^{-1} = \beta_f^{-1} \boldsymbol{I}$ in the recursive LSTD algorithm (Algorithm 5). The posterior mean of GPTD also corresponds to LSTD with an ℓ_2-regularization of the operator problem if both algorithms are extended with eligibility traces (see the next section). In addition to the regularization of the operator problem, Farahmand et al. (2008) and Hoffman et al. (2011) proposed an ℓ_2-penalty for the fixpoint problem (i.e., $g(\boldsymbol{\omega}) = \beta_g \|\boldsymbol{\omega}\|_2^2$). There are still closed-form solutions for both problems with ℓ_2-regularizations. However, the benefits of such regularization in comparison to just using $f(\boldsymbol{\theta})$ still need to be explored.

Regularization with ℓ_1-norm was first used by Kolter and Ng (2009) in the operator problem, that is, $f(\boldsymbol{\theta}) = \beta_f \|\boldsymbol{\theta}\|_1$ and $g(\boldsymbol{\omega}) = 0$. They showed that ℓ_1-regularization gives consistently better results than ℓ_2 in a policy iteration framework and is computationally faster for a large number of irrelevant features. Yet, using ℓ_1-regularization for the OPE problem prevents a closed form solution and the resulting optimization problem called Lasso-TD is non-convex. The least-angle regression algorithm (Efron et al., 2004) could be adapted to solve this optimization problem which yielded the LARS-TD algorithm (Kolter and Ng, 2009). Johns et al. (2010) started from the Lasso-TD problem but reformulated it as a linear complementarity problem (Cottle et al., 1992), for which standard solvers can be employed. Additionally, this linear complementary TD (LC-TD) formulation allows using warm-starts when the policy changes.[4] Ghavamzadeh et al. (2011) showed that the Lasso-TD problem has a unique fixpoint which means that LC-TD and LARS-TD converge to the same solution. In addition, Ghavamzadeh et al. (2011) provided bounds on the MSE for this fixpoint.

The ℓ_1-Projected-Bellman-Residual (ℓ_1-PBR) method (Geist and Scherrer, 2011) puts the regularization onto the fixpoint problem instead of the operator problem, that is, $f(\boldsymbol{\theta}) = 0$ and $g(\boldsymbol{\omega}) = \beta_g \|\boldsymbol{\omega}\|_1^2$. Hoffman et al. (2011) proposed a similar technique but with additional ℓ_2-penalty on the operator problem. Regularizing FPE problem with an ℓ_1 norm allows for a closed form solution of the OPE problem. Using this solution in the regularized FPE problem reduces to a standard Lasso problem and, hence, a standard Lasso solver

3. For notational simplicity, we slightly abuse notation and use the true OPE and FPE objectives instead of the sample-approximations at time t.
4. Warm-starts are valuable in policy iteration: The solution of the last policy can be used to substantially speed-up the computation of the value function for the current policy.

can be employed instead of specialized solution as for the Lasso-TD problem. Furthermore, Lasso-TD has additional requirements on the $\boldsymbol{A}$-matrix of LSTD[5] which generally only hold in on-policy learning. Approaches with ℓ_1-regularized operator problems do not have this limitation and only make mild assumptions in off-policy settings. Despite these theoretical benefits, empirical results indicate comparable performance to the Lasso-TD formulation and, hence, ℓ_1-regularization for the FPE problem is a promising alternative.

Petrik et al. (2010) propose using ℓ_1-regularization in the linear program formulation of dynamic programming for finding the value function. However, their analysis concentrates on the case where the transition kernel $\mathcal{P}^\pi$ is known or approximated by multiple samples. Another family of methods considers the linear system formulation of LSTD $\boldsymbol{A\theta} = \boldsymbol{b}$ directly. Pires (2011) suggests to solve this system approximately with additional ℓ_1-regularization

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{A\theta} - \boldsymbol{b}\|_2^2 + \frac{\beta}{t}\|\boldsymbol{\theta}\|_1.$$

Again, this problem is a standard convex Lasso problem solvable by standard algorithms and applicable to off-policy learning. Dantzig-LSTD (D-LSTD, Geist et al., 2012) takes a similar approach and considers

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_1 \quad \text{subject to} \quad \|\boldsymbol{A\theta} - \boldsymbol{b}\|_\infty \leq \frac{\beta}{t}.$$

This optimization problem, a standard linear program, is motivated by the Dantzig selector of Candes and Tao (2005) and can be solved efficiently. It is also well-defined for off-policy learning. The aim of this problem is to minimize the sum of all parameters while making sure that the linear system of LSTD is not violated by more than βt^{-1} in each dimension.

All regularization approaches so far have treated each parameter dimension equally. However, it might be helpful to give the parameter components different weights. Johns and Mahadevan (2009) suggested to use the Laplacian $\boldsymbol{L}$ of the graph-based representation of the MDP as weights and add $\beta_L\|\boldsymbol{L\Phi\theta}\|_{\boldsymbol{D}}^2$ as an additional term to the MSPBE. Investigating the benefits of other problem-dependent weighted norms are left for future work. The performance of all regularization schemes strongly depends on the regularization strength β, which has to be specified by hand, found by cross-validation or set with the method of Farahmand and Szepesvári (2011).

Instead of regularizing the value-function estimation problem, we could also estimate the value function directly with LSTD in a lower-dimensional feature space. Ghavamzadeh et al. (2010) showed in a theoretical analysis that projecting the original high-dimensional features to a low-dimensional space with a random linear transformation (LSTD with Random Projections, LSTD-RP) has the same effect as regularization. However, no empirical results for this algorithm are given. Alternatively, features can be selected explicitly to form the lower-dimensional space. Hachiya and Sugiyama (2010) proposed to consider the conditional mutual information between the rewards and the features of observed states and provided an efficient approximation scheme to select a good subset as features.

There have also been efforts to regularize Bellman residual minimization. Loth et al. (2007) added an ℓ_1-penalty to the MSBE and proposed a gradient-based technique to find

5. The matrix has to be a P-matrix. P-matrices, a generalization of positive definite matrices, are square matrices with all of their principal minors positive.

the minimum incrementally. In contrast, Farahmand et al. (2008) regularized with an ℓ_2-term to obtain the optimum in closed form. Gradient-based TD-algorithms are less prone to over-fitting than least-squares approaches when few transitions are observed as the norm of the parameter vector is always limited for a small number of updates. However, if the observed transitions are re-used by running several sweeps of stochastic gradient updates, regularization becomes as relevant as for the least-squares approaches. In addition, if a large number of features are irrelevant for the state value, the gradient and especially its stochastic approximation becomes less reliable. Therefore, there has been recent interest in promoting sparseness by adding a soft-threshold shrinkage operator to gradient-based algorithms (Painter-Wakefield and Parr, 2012b; Meyer et al., 2012) and reformulating the regularized objective as a convex-concave saddle-point problem (Liu et al., 2012).

Despite the extensive work on regularization schemes for LSTD, many directions still need to be explored. For example, many feature spaces in practice have an inherent structure. They may for instance consist of multiple coverings of the input space with radial basis functions of different widths. There has been work on exploiting such structures with hierarchical regularization schemes in regression and classification problems (Zhao et al., 2009; Jenatton et al., 2010). These approaches divide the parameters into groups and order the groups in a hierarchical structure (e.g., trees), which determines the regularization in each group. While such schemes have been successfully applied to images and text documents, it is an open question whether they can be adapted to work online and to which extent policy evaluation tasks could benefit from hierarchical regularization.

2.4 Important Extensions

In the previous sections, temporal-difference methods have been introduced in their most basic form to reveal the underlying ideas and avoid cluttered notation. We now introduce two extensions which are applicable to most methods. First, we briefly discuss eligibility traces for improving the learning speed by considering the temporal difference of more than one timestep. Subsequently, importance-reweighting is presented which enables temporal-difference methods to estimate the value function from off-policy samples. While the aim of this paper is giving a survey of existing methods, we will also present an alternative implementation of importance reweighting for LSTD and TDC which considerably decreases the variance of their estimates. We focus on the purpose and the functionality of eligibility traces and importance reweighting and, hence, illustrate their actual implementation only for selected TD methods.

2.4.1 Eligibility Traces

Eligibility traces (e-traces, Sutton, 1988) are an efficient implementation of blending between TD methods and Monte-Carlo sampling. To understand the purpose of this blending, it is beneficial to first identify the different sources of errors in TD methods. While we derived the update rules of the algorithms based on observed samples directly from the underlying cost functions such as MSE, MSBE or MSPBE, we now make the sample-based approximations of the objective functions explicit. These approximations at a given timestep t are denoted by a subscript t, for example, MSE_t, MSBE_t or MSPBE_t (see also Figure 5).

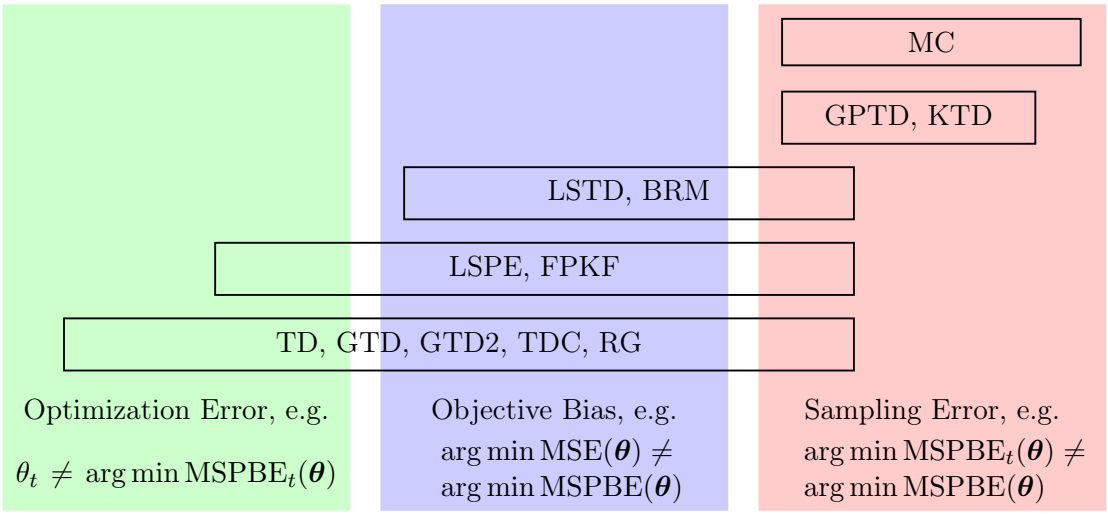


Figure 10: Visualization of Conceptual Error Sources for Policy Evaluation Methods: The *sampling error* is always present and accounts for the approximation of the chosen objective function with observed samples. The higher the variance of these samples the higher the sampling error. If the objective of the method is not directly the MSE, the method will suffer from the *objective bias*. The *optimization error* is present for methods which do not find the minimum of the approximated objective function directly, for example, gradient-based approaches. The positions of the boxes and their overlap with the shaded areas denote the extent, to which the respective methods suffer from each error source. Note that the actual amount of each error source is not visualized and varies drastically between different MDPs, feature representations, policies and number of time steps. MSPBE_t denotes the approximation of the MSPBE with samples observed at timesteps 1 to t.

Error Decomposition. Leaving numerical issues aside, there are three conceptual sources of errors as illustrated in Figure 10. Consider for example Monte-Carlo sampling, which does not rely on temporal differences, but simply takes the observed accumulated reward as a sample for each state. Hence, at time t, it finds the parameter estimate by computing the minimum of

$$\mathrm{MSE}_t(\boldsymbol{\theta}) = \sum_{i=0}^{t} \left(\boldsymbol{\phi}_i^T \boldsymbol{\theta} - \sum_{k=i}^{t} \gamma^{k-i} r_k \right)^2 .$$

After infinitely many time steps, this sample-based approximation of the MSE converges to the true error prescribed by Equation (4), which can also be written as

$$\mathrm{MSE}(\boldsymbol{\theta}) = \left\| \boldsymbol{\Phi}\boldsymbol{\theta} - \sum_{k=0}^{\infty} \gamma^k \boldsymbol{P}^k \boldsymbol{R} \right\|_{\boldsymbol{D}}^2 . \tag{35}$$

The difference between the approximation and the true objective function, referred to as *sampling error*, is present for all methods. TD methods avoid estimating $\gamma^k \boldsymbol{P}^k \boldsymbol{R}$ for $k > 0$ directly by replacing these terms with the value function estimate, that is, use bootstrapping with the Bellman operator T. As the replaced terms cause the high variance, the sampling error decreases at the price of a possible increase in the *objective bias*. This bias denotes the difference between the minimum of the TD objective function (such as MSPBE or MSBE) and the true minimum of the MSE. The regularization with priors in GPTD and KTD is an alternative for reducing the variance at the price of a temporary objective bias. Descent approaches such as the gradient methods, LSPE or FPKF do not compute the minimum of the current objective approximation analytically, but only make a step in its direction. Hence, they suffer from an additional *optimization error*. Although the errors caused by each source do not add up, but may counterbalance each other, it is a reasonable goal to try to minimize the effect of each source.

The magnitude of each type of error depends on the actual MDP, feature representation, policy and number of observed transitions. For example, the objective bias of MSPBE or MSBE vanishes for features that allow representing the true value function exactly. On the other hand, a setup where the MDP and policy are deterministic has zero variance for $\gamma^k \boldsymbol{P}^k \boldsymbol{R}$ and hence, introducing a bias with bootstrapping does not pay off. By interpolating between TD methods and Monte-Carlo estimates, we can often find an algorithm where the effects of sampling error and objective bias is minimized. The natural way to do so is to replace $\gamma^k \boldsymbol{P}^k \boldsymbol{R}$ only for terms $k > h$ in Equation (35), which yields the *h-step Bellman operator*

$$T^h \boldsymbol{V} = \underbrace{TT \dots T}_{h \text{ times}} \boldsymbol{V} = \gamma^h \boldsymbol{P}^h \boldsymbol{V} + \sum_{k=0}^{h-1} \gamma^k \boldsymbol{P}^k \boldsymbol{R}.$$

As it considers the h future rewards, it is also referred to as h-step look-ahead (Sutton and Barto, 1998). If we used the h-step Bellman operator to redefine the objective functions (e.g., MSBE or MSPBE), we would need to observe the rewards $r_t, r_{t+1}, \dots r_{t+h-1}$ and state s_{t+h} before we could use s_t for estimation, that is, approximating $T^h V(s_t)$ with a sample corresponds to

$$T^h V(s_t) \approx \gamma^h V(s_{t+h}) + \sum_{k=0}^{h-1} \gamma^k r_{t+k}.$$

Hence, online estimation is not possible for large h. Eligibility traces circumvent this problem and allow taking each sample into account immediately.

Eligibility Traces. Eligibility traces rely on the λ-*Bellman operator* T_λ defined as a weighted average over all T^k

$$T_\lambda = (1 - \lambda) \sum_{k=0}^{\infty} \lambda^k T^{k+1}. \tag{36}$$

The term $(1-\lambda)$ is a normalization factor which ensures that all weights sum to 1. TD methods extended with eligibility traces minimize objectives redefined on this average Bellman

operator such as the MSPBE$^\lambda$ objective

$$\text{MSPBE}^\lambda(\boldsymbol{\theta}) = \|\boldsymbol{V}_{\boldsymbol{\theta}} - \Pi T_\lambda \boldsymbol{V}_{\boldsymbol{\theta}}\|^2_{\boldsymbol{D}}.$$

For $\lambda = 0$ only $T^1 = T$ is used and, hence, MSPBE^0 corresponds to the standard MSPBE. Only considering T^∞ in the objective corresponds to the MSE from Equation (35). Due to the discount factor γ, there exists a K such that $\|T^k \boldsymbol{V}\|$ deviates less than a small constant from $\|T^\infty \boldsymbol{V}\|$ for all $k > K$. For $\lambda = 1$, the terms $k > K$ in Equation (36) are given infinitely more weight than $k \leq K$ and hence $\lim_{\lambda \to 1} T_\lambda = T^\infty$. We realize that the MSPBE^1 is equivalent to the MSE.

The basic idea of eligibility traces is to approximate the k-step Bellman operators in the weighted sum with samples as soon as possible. Thus, at timestep t, state s_t is used for T^1, state s_{t-1} for T^2, s_{t-2} for T^3 and so on. The special choice of exponentially decreasing weights allows storing the previously observed states efficiently as a summed vector, a so-called eligibility trace.

Implementation for TD Learning. To illustrate the efficient approximation of T_λ and eligibility traces as compact storage of previously observed features, we consider the extension of the standard TD learning algorithm for multi-step temporal differences. Its extended update rule is given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \delta_t \sum_{k=0}^{t} (\lambda\gamma)^k \boldsymbol{\phi}_{t-k}. \tag{37}$$

The parameter λ puts more weight on more recent states. As shown by Sutton and Barto (1998), the multi-step look-ahead (forward view), that is, considering future rewards in the Bellman operator, can also be understood as propagating the temporal-difference error backwards in time (often called the *backward view*), that is, updating the value of states observed before. The update in Equation (37) can be implemented efficiently by computing the sum $\sum_{k=0}^{t} (\lambda\gamma)^k \boldsymbol{\phi}_{t-k}$ incrementally. More precisely, the eligibility trace vector $\boldsymbol{z}_t$ stores the past activations of the features and is updated by

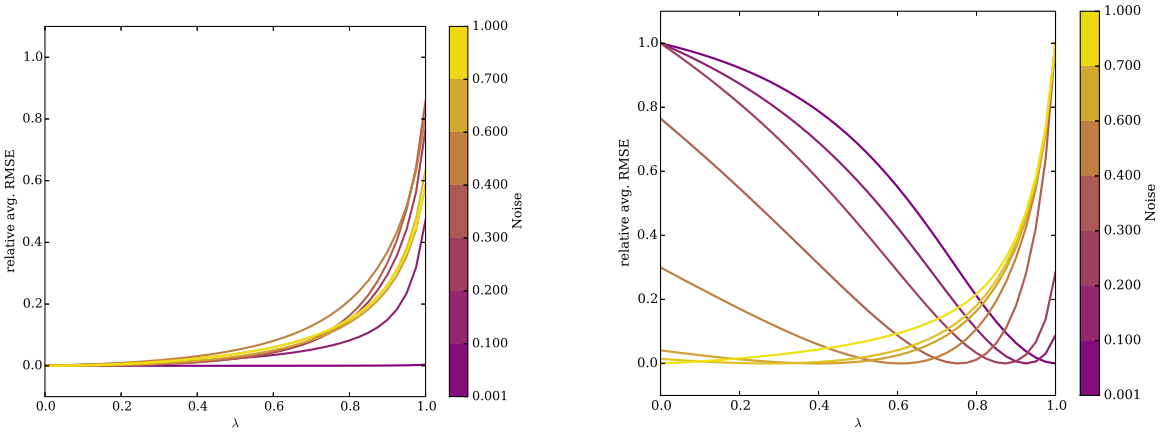
$$\boldsymbol{z}_{t+1} = \boldsymbol{\phi}_t + \lambda\gamma \boldsymbol{z}_t.$$

Updating the eligibility trace in such a way ensures that $\boldsymbol{z}_{t+1} = \sum_{k=0}^{t} (\lambda\gamma)^k \boldsymbol{\phi}_{t-k}$ for all timesteps. The update rule of TD learning can then be written more concisely with eligibility traces as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \delta_t \boldsymbol{z}_{t+1}.$$

The TD learning algorithm with a certain setting of λ is often referred to as TD(λ) where TD(0) corresponds to the standard TD learning algorithm without eligibility traces.

For $\lambda > 0$, the algorithm also updates the value function at states s_h with $h < t$, which is reasonable, as the value at state s_{t-1} is very likely to change when $V(s_t)$ changes. In contrast, TD(0) learning does not reuse its data-points and would need to observe state s_{t-1} again to update the value function at state s_{t-1}. Subsequently, s_{t-2} needs to be visited again to update $V(s_{t-2})$, and so on. Hence, eligibility traces not only allow one to find the best trade-off between objective bias and sampling error, but also reduce the optimization error for gradient based approaches.



(a) Perfect feature representation. If we can approximate the value function perfectly, the bias between MSPBE and MSE is zero, and hence LSTD(0) should always be preferred.

(b) Impoverished features. Here, we have to choose a trade-off between minimizing the variance of the objective function with LSTD(0) and minimizing the bias of the objective function with LSTD(1). The optimal trade-off depends on the amount of noise in the MDP.

Figure 11: Eligibility-Traces implement a trade-off between minimizing the MSE and the MSPBE: Consider Example 2 for the description of the experimental setting. Each graph shows the average of the root of MSE (RMSE $= \sqrt{\text{MSE}}$) of LSTD(λ)-estimates over all timesteps. All curves are normalized by subtracting the minimum and dividing by the maximum. The plots show which λ settings produce lowest errors for for varying amount of stochasticity in the system, where we compare perfect and impoverished features.

Eligibility Traces for Other Algorithms. Most algorithms presented in this paper have been extended to use eligibility traces (see Table 1 for an overview). LSTD (Boyan, 2002), TDC (originally named GTD(λ) in Maei, 2011, but here referred to as TDC(λ) for clarity), FPKF (Scherrer and Geist, 2011; Geist and Scherrer, 2013) and BRM (Scherrer and Geist, 2011; Geist and Scherrer, 2013) have been extended to use multistep-lookahead with eligibility traces. LSPE(λ) (Nedic and Bertsekas, 2003) has been formulated with traces from the beginning. Eligibility traces have also been introduced in GPTD by Engel (2005).[6] Recently, eligibility-traces-versions of GTD2 and the residual-gradient algorithm named GTD2(λ) and gBRM(λ) (Geist and Scherrer, 2013) have been developed. All the algorithms above converge to a minimum of the MSPBE$^\lambda$ objective or respectively MSBE$^\lambda$, which is defined analogously.

Example 2 *We illustrate the benefit of interpolating between MSPBE and MSE in two experiments using e-traces and LSTD. Consider a discrete 40 state / 40 action MDP where actions of the agent deterministically determine its next state. In the first experiment, we use a perfect feature representation, that is, the value function can be estimated perfectly,*

6. In contrast to other methods, the basic version without e-traces corresponds to GPTD(1) and not GPTD(0).

while, in the second experiment, we use an incomplete feature representation by projecting the states linearly on a random 20-dimensional feature space. In both experiments, we evaluated the performance of different λ values for different policies. We varied the stochasticity of the policy, by interpolating between a greedy policy, which visits one state after another, and the uniform policy, which transitions to each state with equal probability.

In Figure 11a, we can see the results for the perfect feature representation. Here, the MSPBE$^\lambda$ does not cause any bias and its minimum coincides with the MSE solution. The plots show the relative MSE values for different λ settings for different levels of stochasticity in the system controlled by the linear blending coefficient between the greedy and uniform policy. The results confirm our intuition that using $\lambda = 0$ is always optimal for perfect features as the MSPBE minimization is unbiased. As the policy is the only source of stochasticity in the system, it behaves deterministically for the greedy policy and the performance is invariant to the choice of λ.

The picture changes for the imperfect feature representation where the minimization of the MSPBE$_\lambda$ causes a bias for the MSE (cf. Figure 11b). Setting $\lambda = 1$ performs best for the greedy policy as there is again no variance on the returns, and, hence, we avoid the bias of the MSPBE$_\lambda$ by directly minimizing the MSE. When gradually increasing the stochasticity of the policy, the optimal value of λ decreases and finally reaches the value of 0. This example illustrates that eligibility traces cause significant speed-ups of learning speed for LSTD(λ) and that the best trade-off between objective bias and sampling error highly depends on the intrinsic stochasticity of the MDP and policy. Hence, λ should be considered as an additional hyper-parameter to optimize for each setting.

2.4.2 Generalization to Off-Policy Learning by Importance Reweighting

In previous sections, we aimed at estimating state values V^π while observing the agent following policy π. However, in many applications we want to know V^π, but only have observed samples with actions chosen by a different policy. Estimating the state values of a different policy than the observed one is referred to as *off-policy value-function estimation* (cf. Section 1.2). For instance, we could be following an exploration policy while we want to know the value function of the optimal greedy policy. In a policy iteration scenario, we can employ off-policy policy evaluation to re-use data points collected with previous policies. Hence, off-policy value-function estimation is an important ingredient for efficiently learning control policies. A different application of off-policy estimation is intra-option learning (Sutton et al., 1998), where we can use samples from different options to update the value functions of the single options.

Importance Reweighting. Leveraging temporal-difference methods for off-policy estimation is based on the idea of *importance sampling* (cf. Glynn and Iglehart, 1989). Importance sampling is a well-known variance-reduction technique in statistics. It is used to approximate the expectation $\mathbb{E}_p[f(X)]$ of a function f with input $X \sim p$, when we cannot directly sample from $p(X)$ but have access to samples from another distribution $q(X)$. In this case, the expectation $\mathbb{E}_p[f(X)]$ can be approximated by

$$\mathbb{E}_p[f(X)] = \mathbb{E}_q\left[\frac{p(X)}{q(X)} f(X)\right] \approx \frac{1}{M}\sum_{i=1}^{M} \frac{p(x_i)}{q(x_i)} f(x_i),$$

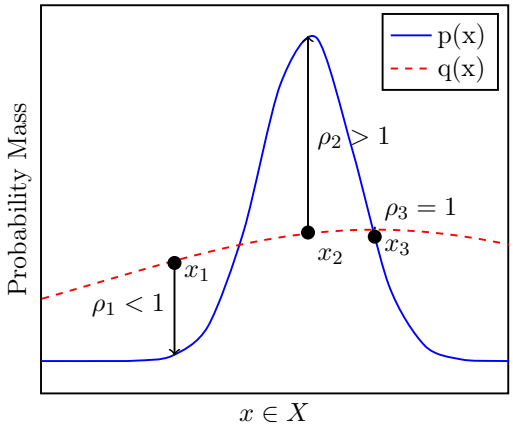


Figure 12: Importance Reweighting: Samples drawn from $q(x)$ are re-weighted by the importance weight ρ to behave like samples from $p(x)$. Data points x_1 in regions, where q is larger than p occur more frequently in the sample from q than from p and are down-weighted. In the orthogonal case x_2, the weights are larger than one and give under-represented samples more weight.

where $x_1, \dots x_M$ are realizations of q. The correctness of this statement in the limit $M \to \infty$ can be easily verified by writing out the expectations. Each sample drawn from q is re-weighted with importance weights $\rho_i = p(x_i)/q(x_i)$ to approximate $\mathbb{E}_p[f(X)]$. See Figure 12 for a visualization. The reweighting is only well-defined, if $q(X) \neq 0$ for all X with non-zero $p(X)$.

Limitations of Off-Policy Estimation. Similar to on-policy estimation, the following observation model for off-policy transitions is assumed: the departing state s_t is distributed according to the state distribution d' while the action a_t is sampled from the behavior policy π_B and the entering state s_{t+1} from the MDP dynamics $\mathcal{P}$. For on-policy value-function estimation, behavior and target policy are the same ($\pi_G = \pi_B$) and d' matches the stationary distribution of the MDP with the policy to evaluate, that is, $d' = d^{\pi_B} = d^{\pi_G}$. However, if the policies differ, the state distribution $d' = d^{\pi_B} \neq d^{\pi_G}$ is not the stationary distribution according to π_G in general. The Example from Section 1.2 in Figure 1 shows such a case.

All approaches to off-policy learning consider the difference of π_G and π_B for the actions a_t but leave the problem of the different stationary distributions unaddressed. Hence, off-policy value-function estimation does not yield the same result as on-policy estimation with samples taken from π_G, even after convergence. For example, the fixpoint of methods minimizing the MSPBE in the off-policy case can be written as

$$\mathrm{MSPBE}(\boldsymbol{\theta}) = \|\boldsymbol{V}_{\boldsymbol{\theta}} - \Pi T^{\pi_G} \boldsymbol{V}_{\boldsymbol{\theta}}\|^2_{\boldsymbol{D}_{\pi_B}}.$$

Hence, the difference of estimating the values with respect to policy π_G from off-policy or on-policy samples is the norm of the objective function. The distance metric $\boldsymbol{D}_{\pi_G}$ and $\boldsymbol{D}_{\pi_B}$ may differ substantially and therefore yield different estimates, which may be a critical

limitation of off-policy estimation. In the following, we use $d = d^{\pi_B}$ to avoid cluttered notation.

In addition, the use of importance reweighting on the policies requires $\pi_B(a|s) > 0$ for all $a \in \mathcal{A}$ and $s \in \mathcal{S}$ with $\pi_G(a|s) > 0$. Thus, each possible sample of the target policy should be observable with the behavior policy. In practice, the behavior policy is often an exploration policy and we aim for value-function estimation of a greedy policy. In this case, the restriction $\pi_B(a|s) > 0$ is not violated.

Let us now discuss specific extensions of TD learning and LSTD for off-policy learning. While we consider the algorithms without eligibility traces for notational simplicity, the derivations hold similarly for algorithms with multi-step predictions. In order to investigate the long-term behavior of algorithms, we have to consider their expected estimates, that is, their update rules in expectation of a transition (defined by the state distribution, the policy and the transition distribution).

Off-Policy TD Learning. First, consider TD learning (Algorithm 1) with the expected parameter update for on-policy learning according to π_G given by $\mathbb{E}_{\pi_G,\mathcal{P},d^{\pi_G}}[\delta_t \phi(s_t)]$. The same updates can be obtained with samples from π_B by rewriting

$$\begin{aligned}
\mathbb{E}_{\pi_G,\mathcal{P},d^{\pi_G}}[\delta_t \boldsymbol{\phi}(s_t)] &= \sum_{s_{t+1}} \sum_{a_t} \sum_{s_t} p(s_t, a_t, s_{t+1}) \delta_t \boldsymbol{\phi}(s_t) \\
&= \sum_{s_{t+1}} \sum_{a_t} \sum_{s_t} \mathcal{P}(s_{t+1}|s_t, a_t) \pi_G(a_t|s_t) d^{\pi_G}(s_t) \;\; \delta_t \boldsymbol{\phi}(s_t) \\
&\approx \sum_{s_{t+1}} \sum_{a_t} \sum_{s_t} \mathcal{P}(s_{t+1}|s_t, a_t) \pi_G(a_t|s_t) d^{\pi_B}(s_t) \;\; \delta_t \boldsymbol{\phi}(s_t) \\
&= \sum_{s_{t+1}} \sum_{a_t} \sum_{s_t} \mathcal{P}(s_{t+1}|s_t, a_t) \pi_B(a_t|s_t) d^{\pi_B}(s_t) \;\; \frac{\pi_G(a_t|s_t)}{\pi_B(a_t|s_t)} \delta_t \boldsymbol{\phi}(s_t) \\
&= \mathbb{E}_{\pi_B,\mathcal{P},d^{\pi_B}}[\rho_t \delta_t \boldsymbol{\phi}(s_t)].
\end{aligned}$$

The expectation w.r.t. the target policy π_G turned into the expectation according to the behavior policy π_B by including the importance weight $\rho_t = \pi_G(a_t|s_t)/\pi_B(a_t|s_t)$. Hence, the off-policy update rule of TD learning is given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \rho_t \delta_t \boldsymbol{\phi}(s_t).$$

Note that on-policy learning can be treated as a special case with $\pi_G = \pi_B$ and $\rho_t = 1$ for all timesteps t. As we will discuss in the following convergence analysis, TD learning might become unstable in off-policy learning. Other gradient methods have also been extended with off-policy weights and do not suffer from this drawback. The Algorithm listings in Appendix C already contain the off-policy weights for all gradient-based and least-squares algorithms.

Convergence Analysis. TD learning may be unstable, when used with a sampling distribution for the states d^{π_B} that differs from the stationary state distribution d^{π_G} induced by the Markov model to evaluate $\mathcal{P}^{\pi_G}$. Consider a batch gradient version of TD learning which

uses the expected gradient instead of the stochastic one. Results from stochastic approximation theory guarantee that the TD learning algorithm converges if batch gradient descent converges and vice versa. In addition, their fixpoints are identical. A batch-gradient step can be written as

$$\begin{aligned}\boldsymbol{\theta}_{k+1} =& \boldsymbol{\theta}_k + \alpha \mathbb{E}[\delta_t \phi_t] \\ =& \boldsymbol{\theta}_k + \alpha \mathbb{E}[(r_t + \gamma \boldsymbol{\phi}_{t+1}^T \boldsymbol{\theta}_k - \boldsymbol{\phi}_t^T \boldsymbol{\theta}_k)\boldsymbol{\phi}_t] \\ =& \boldsymbol{\theta}_k + \alpha \boldsymbol{\Phi}^T \boldsymbol{D} \left(\boldsymbol{R} + \gamma \boldsymbol{P}^{\pi_G} \boldsymbol{\Phi}\boldsymbol{\theta}_k - \boldsymbol{\Phi}\boldsymbol{\theta}_k\right) \\ =& \left(\boldsymbol{I} + \alpha \boldsymbol{A}_{\mathrm{TD}}\right) \boldsymbol{\theta}_k + \alpha \boldsymbol{b}_{\mathrm{TD}}, \end{aligned} \tag{38}$$

with $\boldsymbol{A}_{\mathrm{TD}} = \boldsymbol{\Phi}^T \boldsymbol{D} \left(\gamma \boldsymbol{P}^{\pi_G} - \boldsymbol{I}\right) \boldsymbol{\Phi}$ and $\boldsymbol{b}_{\mathrm{TD}} = \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{R}$.

The iterative update rule of Equation (38) converges if all eigenvalues of the matrix $\boldsymbol{A}_{\mathrm{TD}}$ have only negative real parts (Schoknecht, 2002). It can be shown that if $\boldsymbol{D}$ corresponds to the stationary distribution which has been generated by $\boldsymbol{P}^{\pi_G}$ this condition is satisfied, and hence, TD learning converges. However, this property of $\boldsymbol{A}_{\mathrm{TD}}$ is lost if $\boldsymbol{D}$ does not correspond to the stationary distribution, that is, $\boldsymbol{d}^{\pi_B} \neq \boldsymbol{P}^{\pi_G} \boldsymbol{d}^{\pi_B}$ and, hence, convergence can not be guaranteed for off-policy TD learning. Intuitively, TD learning does not converge because there is more weight on reducing the error of the value function for the starting states s_t of a transition than for the successor states s_{t+1}. Hence, the Bellman error in the successor states might increase, which again affects the estimation of the target values for the next parameter update. If $\boldsymbol{d} = \boldsymbol{P}^{\pi_G} \boldsymbol{d}$, the successor states have the same probability of being updated, and this problem is hence alleviated.

The second order equivalent of batch-gradient TD learning is LSPE. While both methods can be derived from the same nested optimization problem, LSPE is known to converge for off-policy policy evaluation. Hence, it is interesting to briefly look at the reason for this difference. The expected update of LSPE can be obtained from Equations (29) and (30) and is given by

$$\begin{aligned}\boldsymbol{\theta}_{k+1} =& \boldsymbol{\theta}_k + \alpha (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} \left(\boldsymbol{R} + \gamma \boldsymbol{P}^{\pi_G} \boldsymbol{\Phi}\boldsymbol{\theta}_k - \boldsymbol{\Phi}\boldsymbol{\theta}_k\right) \\ =& \left(\boldsymbol{I} + \alpha \boldsymbol{A}_{\mathrm{LSPE}}\right) \boldsymbol{\theta}_k + \alpha \boldsymbol{b}_{\mathrm{LSPE}}, \end{aligned}$$

with $\boldsymbol{A}_{\mathrm{LSPE}} = (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1} \boldsymbol{A}_{\mathrm{TD}}$ and $\boldsymbol{b}_{\mathrm{LSPE}} = (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1} \boldsymbol{b}_{\mathrm{TD}}$. We realize that LSPE scales the TD update with the inverse of a positive definite matrix. This scaling ensures that the matrix $\boldsymbol{A}_{\mathrm{LSPE}}$ stays negative definite and, hence, the update converges (Schoknecht, 2002). More intuitively, the second order term $(\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1}$ normalizes the TD update by the average feature activation, and, hence, overrides the problem that the successor states s_{t+1} have less probability mass than the starting states s_t of a transition.

Off-policy LSTD. We will now discuss how to adapt least-squares approaches for off-policy learning. While we show the off-policy extension for the LSTD algorithm, other methods follow analogously. LSTD relies on the estimates $\boldsymbol{A}_t$ and $\boldsymbol{b}_t$ from Equations (25) and (26) to converge to the true values $\boldsymbol{A}$ and $\boldsymbol{b}$ in Equation (24). In expectation, the estimates at

time t can be written as

$$\mathbb{E}_{d,\pi_G,\mathcal{P}}\left[\boldsymbol{A}_t\right] = \mathbb{E}_d\left[\sum_{i=0}^{t} \boldsymbol{\phi}_i \left(\boldsymbol{\phi}_i - \gamma \mathbb{E}_{\pi_G,\mathcal{P}}\left[\boldsymbol{\phi}_{i+1}\right]\right)^T\right] \quad \text{and}$$

$$\mathbb{E}_{d,\pi_G,\mathcal{P}}\left[\boldsymbol{b}_t\right] = \mathbb{E}_{d,\pi_G,\mathcal{P}}\left[\sum_{i=0}^{t} \boldsymbol{\phi}_i r_i\right].$$

We realize that the only parts which depend on the policy π_G are the terms $\mathbb{E}_{\pi_G,\mathcal{P}}\left[\boldsymbol{\phi}_{i+1}\right]$ and $\mathbb{E}_{d,\pi_G,\mathcal{P}}\left[\sum_{i=0}^{t} \boldsymbol{\phi}_i r_i\right]$. If a behavior policy π_B is used instead of π_G, these parts have to be re-weighted which results in

$$\mathbb{E}_{d,\pi_G,\mathcal{P}}\left[\boldsymbol{A}_t\right] = \mathbb{E}_d\left[\sum_{i=0}^{t} \boldsymbol{\phi}_i \left(\boldsymbol{\phi}_i - \gamma \mathbb{E}_{\pi_B,\mathcal{P}}\left[\frac{\pi_G(a_i|s_i)}{\pi_B(a_i|s_i)}\boldsymbol{\phi}_{i+1}\right]\right)^T\right] \quad \text{and}$$

$$\mathbb{E}_{d,\pi_G,\mathcal{P}}\left[\boldsymbol{b}_t\right] = \mathbb{E}_{d,\pi_B,\mathcal{P}}\left[\sum_{i=0}^{t} \frac{\pi_G(a_i|s_i)}{\pi_B(a_i|s_i)}\boldsymbol{\phi}_i r_i\right].$$

For the sample-based implementation, we arrive at the off-policy parameter estimates of LSTD proposed by Bertsekas and Yu (2009)

$$\boldsymbol{A}_t = \sum_{i=0}^{t} \boldsymbol{\phi}_i[\boldsymbol{\phi}_i - \gamma\rho_i\boldsymbol{\phi}_{i+1}]^T, \quad \boldsymbol{b}_t = \sum_{i=0}^{t} \boldsymbol{\phi}_i\rho_i r_i.$$

We refer to this off-policy reweighting as *Standard Off-Policy Reweighting*. Taking eligibility traces into account and making use of the Sherman-Morrison formula (Equation 27), the recursive off-policy version of LSTD, shown in Algorithm 5 in Appendix C, can be derived (Scherrer and Geist, 2011; Geist and Scherrer, 2013).

The $\boldsymbol{\phi}_i\boldsymbol{\phi}_i^T$ terms in $\boldsymbol{A}_t$ are not re-weighted since it is not necessary to add importance weights to terms which do not depend on the policy for ensuring convergence to the desired solution. However, as our experiments presented in Section 3.4 show, such an approach suffers from a severe drawback. To illustrate the reason, consider the effective number of samples used to calculate the different terms. The effective number of samples for calculating the first term of $\boldsymbol{A}_t$ is always t while, for the second term, the effective number is $\varrho = \sum_i^t \rho_i$. In expectation, ϱ is equal to t and the expected estimate of $\boldsymbol{A}$ is unbiased. However, for a specific sample-based realization, ϱ will in general be different from t. As both terms in $\boldsymbol{A}_t$ are not normalized by the number of samples used for the estimate, a big part of the variance in estimating $\boldsymbol{A}_t$ will just come from the difference of ϱ to t. Despite positive theoretical analysis of the convergence properties of this reweighting strategy (Bertsekas and Yu, 2009; Yu, 2010), our experiments reveal that for more complex problems, for example, in continuous domains, the performance of LSTD with this type of reweighting breaks down due to the drastically increased variance in $\boldsymbol{A}_t$. Instead, the matrix $\boldsymbol{A}_t$ can be estimated more robustly by using the importance weight for the whole transition, that is,

$$\boldsymbol{A}_t = \sum_{i=0}^{t} \rho_i\boldsymbol{\phi}_i[\boldsymbol{\phi}_i - \gamma\boldsymbol{\phi}_{i+1}]^T, \quad \boldsymbol{b}_t = \sum_{i=0}^{t} \boldsymbol{\phi}_i\rho_i r_i.$$

A recursive method based on these updates, *LSTD Transition Off-Policy Reweighting (LSTD-TO)*, is shown in Algorithm 6. Similar reweighting strategies can be formulated for LSPE which yields *LSPE-TO* shown in Algorithm 8. To the best of our knowledge, using a transition-based reweighting for LSTD and LSPE has not been introduced in the literature so far, but is crucial for the performance of off-policy learning with least-squares methods.

3. Comparison of Temporal-Difference Methods

In this section, we compare the performance and properties of the presented policy evaluation methods quantitatively in various experiments. All algorithms were implemented in Python. The source code for each method and experiment is available at `http://github.com/chrodan/tdlearn`. In addition, further supplementary material is available at `http://www.ias.tu-darmstadt.de/Research/PolicyEvaluationSurvey`.

In Section 3.1, we present the experimental setting including the benchmark tasks and the evaluation process. Subsequently, the most important insights gained from the experimental evaluation are discussed. Section 3.2 focuses on results concerning cost functions, Section 3.3 concerning gradient-based methods and Section 3.4 covers results regarding least-squares methods.

3.1 Benchmarks

To evaluate the properties of policy evaluation methods under various conditions, we selected a number of representative benchmark tasks with different specifications. We computed the algorithms' predictions with an increasing number of training data points, and compared their quality with respect to the MSE, MSBE and MSPBE. These experiments are performed on six different Markov decision processes, three with discrete and three with continuous state space. Most experiments were performed both with on-policy and off-policy samples. We also evaluated different feature representations which altogether resulted in the following 12 settings.

1. 14-State Boyan Chain
2. Baird Star Example
3. 400-State Random MDP On-policy
4. 400-State Random MDP Off-policy
5. Linearized Cart-Pole Balancing On-policy Imperfect Features
6. Linearized Cart-Pole Balancing Off-policy Imperfect Features
7. Linearized Cart-Pole Balancing On-policy Perfect Features
8. Linearized Cart-Pole Balancing Off-policy Perfect Features
9. Cart-Pole Swingup On-policy
10. Cart-Pole Swingup Off-policy

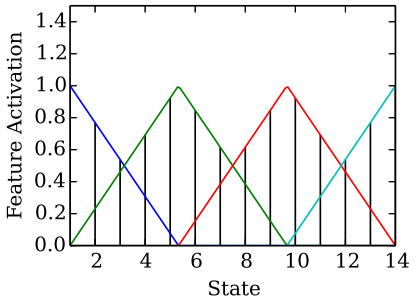


Figure 13: Feature Activation for the Boyan chain benchmark. The state space is densely covered with triangle-shaped basis functions.

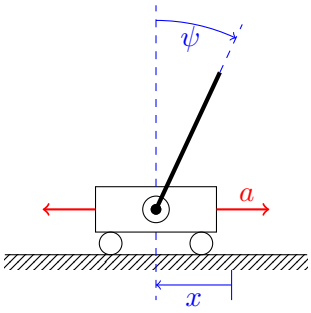


Figure 14: The Cart-Pole System. The pendulum has to be balanced around the peak by moving the cart.

11. 20-link Linearized Pole Balancing On-policy

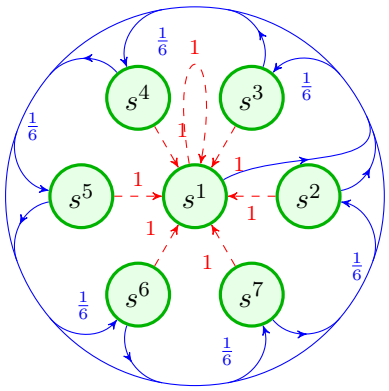
12. 20-link Linearized Pole Balancing Off-policy

3.1.1 Boyan's Chain (Benchmark 1)

The first benchmark MDP is the classic chain example from Boyan (2002). We considered a chain of 14 states $\mathcal{S} = \{s^1, \ldots, s^{14}\}$ and one action. Each transition from state s^i results in state s^{i+1} or s^{i+2} with equal probability and a reward of -3. If the agent is in the second last state s^{13}, it always proceeds to the last state with reward -2 and subsequently stays in this state forever with zero reward. A visualization of a 7-state version of the Boyan chain is given in Figure 6. We chose a discount factor of $\gamma = 0.95$ and four-dimensional feature description with triangular-shaped basis functions covering the state space (Figure 13). The true value function, which is linearly decreasing from s^1 to s^{14}, can be represented perfectly.

3.1.2 Baird's Star Example (Benchmark 2)

Baird's star (Baird, 1995) is a well known example for divergence of TD learning in off-policy scenarios. It is often referred to as "star" MDP as its states can be ordered as a star, one central state and six states at the edges, as shown in Figure 15. There are two



Features:

$$\phi(s^i) = 2e_i + [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]^T \quad \text{for } i = 2 \dots 7$$

$$\phi(s^1) = e_1 + 2e_7 = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 2]^T$$

Policies:

$$\pi_B(\cdot\,|s^i) = \begin{cases} \frac{1}{7} & \text{for - -} \\ \frac{6}{7} & \text{for ——} \end{cases}, \quad \text{for } i = 1 \dots 7$$

$$\pi_G(\cdot\,|s^i) = \begin{cases} 1 & \text{for - -} \\ 0 & \text{for ——} \end{cases}, \quad \text{for } i = 1 \dots 7$$

Figure 15: Baird's Star: 7-State Star MDP, a classic off-policy example problem from Baird (1995) in which TD learning diverges for all step-sizes. While the label of a transition denotes its probability, the reward is always zero. The vector e_i denotes the i-th unit vector.

actions. The solid action chooses one of the solid edges with equal probability and the dashed action always chooses the edge to the central state. We set the discount factor $\gamma = 0.99$ and assume zero reward for each transition. Hence, the true value function is zero in every state for all policies. The evaluation policy always takes the dashed action, and hence, goes to the central state. However, the behavior policy, that is, the policy used to generate the samples, chooses the solid action with probability 6/7. The feature vector has eight components. For the outside-states s^i, $i = 2, \dots, 7$, the i-th entry has value 2 and the last entry is 1 (cf. Figure 15). All other entries are zero. The central state sets the first component of $\boldsymbol{\phi}$ to 1 and the last component to 2. All other entries are again zero. We used $\boldsymbol{\theta}_0 = [1\ 1\ 1\ 1\ 1\ 1\ 10\ 1]^T$ as initial parameter vector for the methods that allow specifying a start estimate. Although the true value function is zero everywhere, TD-learning is known to diverge for this initialization of the parameter-vector.

3.1.3 Randomly Sampled MDP (Benchmarks 3 and 4)

To evaluate the prediction in MDPs with more states and of a less constructed nature, we used a randomly generated discrete MDP with 400 states and 10 actions. The transition probabilities were distributed uniformly with a small additive constant to ensure ergodicity of the MDP, that is,

$$\mathcal{P}(s'|a, s) \propto p^a_{ss'} + 10^{-5}, \quad p^a_{ss'} \sim U[0, 1].$$

The data-generating policy, the target policy as well as the start distribution are sampled in a similar manner. The rewards are uniformly distributed, that is, $r(s^i, a^j) \sim U[0, 1]$. Each state is represented by a 201-dimensional feature vector, 200 dimensions which have been generated by sampling from a uniform distribution and one additional constant feature. The MDP, the policies and the features are sampled once and then kept fix throughout all experiments (all independent trials were executed in the same setting). As behavior- and target-policy were generated independently and differ substantially for Benchmark 4,

the algorithms were tested in a difficult off-policy setting. The discount factor is set to $\gamma = 0.95$.

3.1.4 Linearized Cart-Pole-Balancing (Benchmarks 5- 8)

The Cart-Pole Balancing problem is a well known benchmark task which has been used for various reinforcement learning algorithms. As we want to know the perfect feature representation also for a continuous system, we linearized cart-pole dynamics and formulated the task as a linear system with a quadratic reward function and Gaussian noise. Linear-Quadratic-Gaussian (LQG) systems are one of the few continuous settings for which we can compute the true value function exactly. The perfect features for the value function of a LQG system are all first and second order terms of the state vector $\boldsymbol{s}$.

Figure 14 visualizes the physical setting of the benchmark. A pole with mass m and length l is connected to a cart of mass M. It can rotate 360° and the cart can move right and left. The task is to balance the pole upright. The state $\boldsymbol{s} = [\psi, \dot{\psi}, x, \dot{x}]^T$ consists of the angle of the pendulum ψ, its angular velocity $\dot{\psi}$, the cart position x and its velocity $\dot{x}$. The action a acts as a horizontal force on the cart. The system dynamics are given by (cf. Deisenroth, 2010, Appendix C.2 with $\psi = \theta + \pi$)

$$\ddot{\psi} = \frac{-3ml\dot{\psi}^2 \sin(\psi)\cos(\psi) + 6(M+m)g\sin(\psi) - 6(a - b\dot{\psi})\cos(\psi)}{4l(M+m) - 3ml\cos(\psi)} \quad \text{and} \tag{39}$$

$$\ddot{x} = \frac{-2ml\dot{\psi}^2 \sin(\psi) + 3mg\sin(\psi)\cos(\psi) + 4a - 4b\dot{\psi}}{4(M+m) - 3m\cos(\psi)}, \tag{40}$$

where $g = 9.81\frac{m}{s^2}$ and b is a friction coefficient of the cart on the ground (no friction is assumed between pole and cart). If the pole is initialized at the upright position and the policy is keeping the pole around this upright position, the system dynamics can be approximated accurately by linearizing the system at $\psi = 0$. In this case, the linearization yields $\sin\psi \approx \psi$, $\dot{\psi}^2 \approx 0$ and $\cos\psi \approx 1$ and we obtain the linear system

$$\boldsymbol{s}_{t+1} = \begin{bmatrix} \psi_{t+1} \\ \dot{\psi}_{t+1} \\ x_{t+1} \\ \dot{x}_{t+1} \end{bmatrix} = \begin{bmatrix} \psi_t \\ \dot{\psi}_t \\ x_t \\ \dot{x}_t \end{bmatrix} + \Delta t \begin{bmatrix} \dot{\psi}_t \\ \frac{3(M+m)\psi - 3a + 3b\dot{\psi}}{4Ml - ml} \\ \dot{x}_t \\ \frac{3mg\psi + 4a - 4b\dot{\psi}}{4M - m} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ z \end{bmatrix},$$

where the time difference between two transitions is denoted by $\Delta t = 0.1$s and z is Gaussian noise on the velocity of the cart with standard deviation 0.01. We set the length of the pole l to 0.6m, the mass of the cart M to 0.5kg, the mass of the pole m to 0.5kg and the friction coefficient of b to $0.1\,\mathrm{N(ms)}^{-1}$. The reward function is given by

$$R(\boldsymbol{s}, a) = R(\psi, \dot{\psi}, x, \dot{x}, a) = -100\psi^2 - x^2 - \frac{1}{10}a^2,$$

that is, deviations from the desired pole position are strongly penalized, while large offsets of the cart and the magnitude of the current action cause only minor costs. As the transition model is a Gaussian with a linear mean-function and the reward function is quadratic, the exact value function and optimal policy can be computed by dynamic programming (Bertsekas

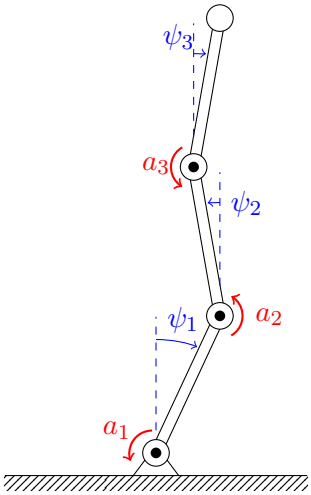


Figure 16: Balancing setup of a 3-link actuated pendulum. Each joint i is actuated by the signal a_i. The state of each joint is denoted by its angle ψ_i against the vertical direction. The pole is supposed to be balanced upright, that is, all ψ_i should be as close to 0 as possible.

and Tsitsiklis, 1996). It is well known that the features of the true value function are given by a constant plus all squared terms of the state[7] $\phi_p(\boldsymbol{s}) = [1, s_1^2, s_1 s_2, s_1, s_3, s_1 s_4, s_2^2, \ldots, s_4^2]^T \in \mathbb{R}^{11}$. The optimal policy $\pi(a|\boldsymbol{s}) = \mathcal{N}(a|\boldsymbol{\beta}^T \boldsymbol{s}, \sigma^2)$ is linear. The target policy π_G is set to the optimal policy, that is, the gains $\boldsymbol{\beta}$ are obtained by dynamic programming and the exploration rate σ^2 is set to a low noise level. The data-generating policy π_B uses the same $\boldsymbol{\beta}$ but a higher noise level in the off-policy case.

To additionally compare the algorithms on a approximate feature representation, we used $\phi_a(\boldsymbol{s}) = [1, s_1^2, s_2^2, s_3^2, s_4^2]^T \in \mathbb{R}^5$ as feature vector in Benchmarks 5 and 6. All evaluations were generated with a discount factor of $\gamma = 0.95$.

3.1.5 Linearized 20-link Balancing (Benchmark 11 and 12)

To evaluate the algorithms on systems with higher-dimensional state- and action-spaces, we considered a 20-link actuated inverted pendulum. Each of the 20 rotational joints are controlled by motor torques $\boldsymbol{a} = [a_1, \ldots a_{20}]^T$ to keep the pendulum balanced upright. See Figure 16 for a visualization of a pendulum with 3 links. The difference in the angle to the upright position of link i is denoted by ψ_i. The state space is 40 dimensional and consists of the angles of each joint $\boldsymbol{q} = [\psi_1, \ldots, \psi_{20}]^T$ and the angular velocities $\dot{\boldsymbol{q}} = [\dot{\psi}_1, \ldots, \dot{\psi}_{20}]^T$.

The derivation of the linearized system dynamics can be found in the supplementary material and yields

$$\begin{bmatrix} \mathbf{q}_{t+1} \\ \dot{\mathbf{q}}_{t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \Delta t\, \boldsymbol{I} \\ -\Delta t\, \mathbf{M}^{-1}\boldsymbol{U} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \mathbf{q}_t \\ \dot{\mathbf{q}}_t \end{bmatrix} + \Delta t \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1} \end{bmatrix} \mathbf{a} + \boldsymbol{z},$$

7. The linear terms disappear as we have linearized at $\boldsymbol{s} = \mathbf{0}$.

where $\Delta t = 0.1$s is the time difference of two time steps and $\boldsymbol{M}$ is the mass matrix in the upright position. Its entries are computed by $\boldsymbol{M}_{ih} = l^2(21 - \max(i,h))m$ with length $l = 5$m and mass $m = 1$kg of each link. The matrix $\boldsymbol{U}$ is a diagonal matrix with entries $\boldsymbol{U}_{ii} = -gl(21 - i)m$. Each component of $\boldsymbol{z}$ contains Gaussian noise. Again, we used a quadratic reward function

$$R(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{a}) = -\boldsymbol{q}^T \boldsymbol{q},$$

which penalizes deviations from the upright position. The target policy is given by the optimal policy (obtained by dynamic programming) with Gaussian noise and analogously the behavior policy but with increased noise level for the off-policy estimation case. A discount factor of $\gamma = 0.95$ and a 41-dimensional approximate feature representation $\phi(\boldsymbol{q}, \dot{\boldsymbol{q}}) = [\psi_1^2, \psi_2^2, \ldots, \psi_{20}^2, \dot{\psi}_1^2, \dot{\psi}_2^2, \ldots, \dot{\psi}_{20}^2, 1]^T$ were used in the experiments.

3.1.6 Cart-Pole Swing-up (Benchmarks 9 and 10)

Besides discrete and linear systems, we also include a non-linear problem, the cart-pole swing-up task. The non-linear system dynamics from Equations (39) and (40) were used and the constants were set to the same values as in the linearized task. The reward function directly rewards the current height of the pole and mildly penalizes offsets of the cart

$$R(\boldsymbol{s}, a) = R(\psi, \dot{\psi}, x, \dot{x}, a) = \cos(\psi) - 10^{-5}|x|.$$

We used an approximately optimal policy learned with the PILCO-Framework (Deisenroth and Rasmussen, 2011) and added Gaussian noise to each action a. The resulting policy manages to swing-up and balance the pendulum in about 70% of the trials, depending on the initial pole position which is sampled uniformly. Each episode consists of 200 timesteps of 0.15s duration. A normalized radial basis function network and an additional constant feature has been chosen as feature representation. To obtain a compact representation, we first covered the four-dimensional state space with a grid of basis functions and then removed all features for which the summed activations were below a certain threshold. Thus, we omitted unused basis functions which are located in areas of the state space, which are not visited. The resulting feature vector had 295 dimensions.

3.1.7 Hyper-Parameter Optimization

The behavior of policy evaluation methods can be influenced by adjusting their hyper-parameters. We set those parameters by performing an exhaustive grid-search in the hyper-parameter space minimizing the MSBE (for the residual-gradient algorithm and BRM) or MSPBE. Optimizing for MSBE or MSPBE introduces a slight bias in the choice of the optimal parameters. For example, smaller value of λ for the eligibility traces are preferred as small values of λ as the objective bias is not taken into account. However, as opposed to the MSE, these objectives can be computed without knowledge of the true values, and, hence, can be evaluated also in practice on a small set of samples. We evaluated the algorithms for an increasing number of observed time steps and computed a weighted average over the errors of all considered time steps to obtain a single score per trial. We increased the weights from 1 for the first to 2 for the last estimate and therefore put emphasis on a good final value of the estimates but also promoted fast convergence. The scores of three independent trials

Parameter	Evaluated Values
α	$2 \cdot 10^{-4}$, $5 \cdot 10^{-4}, 10^{-3}$, $0.002, \ldots, 0.009, 0.01,$ $0.02, \ldots, 0.09, 0.1, 0.2, 0.3, 0.4, 0.5$
α_{LSPE}	$0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 0.8, 0.9, 1$
α_{FPKF}	$0.01, 0.1, 0.3, 0.5, 0.8, 1$
β_{FPKF}	$1, 10, 100, 1000,$
τ_{FPKF}	$0, 500, 1000$
μ	$10^{-4}, 10^{-3}, 10^{-2}, 0.05, 0.1, 0.5, 1, 2, 4, 8, 16$
λ	$0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1$
ϵ	$10^5, 10^3, 10^2, 10, 1, 0.1, 0.01$
ζ	$0.01, 0.02, \ldots 0.09, 0.1, 0.2, \ldots, 0.9, 1, 5, 10, 30$
η	$0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.5$

Table 3: Considered values in the grid-search parameter optimization for the algorithms listed in Table 4.

were averaged to obtain a more stable cost function during hyper-parameter grid-search. Table 4 provides a listing of all considered algorithms with their hyper-parameters. Each parameter in Table 4 is evaluated in the grid-search at the values listed in Table 3.

All shown results are averages over 50 independent trials for continuous MDPs or 200 trials for discrete MDPs, if not stated otherwise. For discrete and linear continuous systems, the MSBE / MSPBE / MSE values were calculated exactly, whereas the stationary state distribution d^π is approximated by samples. For the non-linear continuous system, also the expectations inside the MSBE and MSPBE were approximated by samples while the true value function was estimated by exhaustive Monte-Carlo roll-outs.[8] We often used the square-root of costs to present results in the following, which are denoted by RMSE, RMSBE or RMSPBE.

3.1.8 Normalization of Features

Two types of features were used in the continuous environments, the squared terms of the state $\boldsymbol{s}$ in the linear case, and a radial basis function network in the non-linear case. Both representations were normalized. For the squared terms, we subtracted the mean feature vector and divided by the standard deviation of the individual features. Hence, each feature was normalized to have zero-mean and unit variance. For the radial basis function representation, we always divided the activations of the radial basis functions by their sum,

8. Ten roll-outs were used for each sample of the stationary distribution. We compared the Monte-Carlo estimates from ten roll-outs against estimates from 20 roll-outs on a small subset of samples but did not observe significant differences. Due to the high computational effort, we therefore settled for ten roll-outs per state.

(A) residual-gradient (RG) algorithm
- constant step-sizes $\alpha_t = \alpha$

(B) RG algorithm with double samples (RG DS)
- constant step-sizes $\alpha_t = \alpha$

(C) TD(λ) learning
- constant step-sizes $\alpha_t = \alpha$
- bootstrapping trade-off λ

(D) TD learning decreasing steps
- diminishing step-sizes $\alpha_t = \zeta t^{-\eta}$

(E) GTD
- constant step-sizes $\alpha_t = \alpha$
- second step-size $\beta_t = \alpha\mu$

(F) GTD2
- constant step-sizes $\alpha_t = \alpha$
- second step-size $\beta_t = \alpha\mu$

(G) TDC(λ)
- constant step-sizes $\alpha_t = \alpha$
- second estimate step-size $\beta_t = \alpha\mu$
- bootstrapping trade-off λ

(H) LSTD(λ)
- ℓ_2 regularization ϵ
- bootstrapping trade-off λ

(I) LSTD(λ)-TO
- ℓ_2 regularization ϵ
- bootstrapping trade-off λ

(J) LSPE(λ)
- constant step-sizes $\alpha_t = \alpha_{\text{LSPE}}$
- bootstrapping trade-off λ

(K) LSPE(λ)-TO
- constant step-sizes $\alpha_t = \alpha_{\text{LSPE}}$
- bootstrapping trade-off λ

(L) FPKF(λ)
- constant step-sizes $\alpha_t = \begin{cases} \alpha_{\text{FPKF}} \frac{\beta_{\text{FPKF}}}{\beta_{\text{FPKF}}+t} & \text{for } \tau_{\text{FPKF}} \leq t \\ 0 \quad \text{otherwise} \end{cases}$
- bootstrapping trade-off λ

(M) BRM
- ℓ_2 regularization ϵ
- bootstrapping trade-off λ

(N) BRM with double samples (BRM DS)
- ℓ_2 regularization ϵ

Table 4: Overview of all considered algorithms with their hyper-parameters. As GPTD(λ) is equivalent to LSTD(λ) with ℓ_2 regularization, it is not included explicitly.

that is, the sum of their activations is always 1. Since the feature function includes an additional constant feature 1, the total activation for each state $\boldsymbol{s}$ is $\|\phi(\boldsymbol{s})\|_1 = 2$. Features in discrete settings were not normalized.

3.2 Insights on the Algorithms-Defining Cost Functions

The objective function of the policy evaluation algorithm determines its fixpoint, and, hence, largely influences the final quality of the predictions. As discussed in Section 2.1, only few theoretical results such as loose bounds could be derived. Additionally, constructed examples show that the quality of fixpoints with respect to the mean-squared error highly depends on the problem setting. Therefore, empirical comparisons of the MSTDE, MSBE and MSPBE fixpoints for common problems with different challenges are of particular interest.

Message 1 *Empirically, the magnitudes of the biases of different objective functions with respect to the MSE fixpoint are: bias(MSTDE) $\geq$ bias(MSBE) $\geq$ bias(MSPBE).*

	bias of		
	MSTDE	MSBE	MSPBE
1. 14-State Boyan Chain	1.93	**0.06**	0.10
2. Baird Star Example	**0.00**	0.00	0.03
3. 400-State Random MDP On-policy	0.06	**0.04**	**0.04**
4. 400-State Random MDP Off-policy	0.06	0.08	**0.05**
5. Lin. Cart-Pole Balancing On-pol. Imp. Feat.	4.52	3.80	**2.60**
6. Lin. Cart-Pole Balancing Off-pol. Imp. Feat.	4.37	3.82	**2.47**
7. Lin. Cart-Pole Balancing On-pol. Perf. Feat.	1.92	0.05	**0.03**
8. Lin. Cart-Pole Balancing Off-pol. Perf. Feat.	1.94	0.13	**0.04**
9. Cart-Pole Swingup On-policy	3.83	3.82	**1.99**
10. Cart-Pole Swingup Off-policy	4.28	4.30	**2.17**
11. 20-link Lin. Pole Balancing On-pol.	7.71	7.45	**4.27**
12. 20-link Lin. Pole Balancing Off-pol.	0.08	0.08	**0.04**

Table 5: Mean squared error values of fixpoints of other cost-functions: The fixpoints are estimated by the prediction of LSTD, BRM or BRM with double samples after convergence. The MSPBE has the lowest bias in almost all experiments.

Table 5 shows the observed MSE value of each fixpoint for every benchmark problem. We estimated the MSBE, MSTDE and MSPBE fixpoints by running either the Bellman residual minimization algorithm with or without double sampling or LSTD until convergence (up to a certain accuracy, without eligibility traces). While this procedure introduces approximation errors, which are not entirely neglectable, it still allows us to compare the fixpoints of the cost functions. We ensured that regularization did not impair with the results by comparing the fixpoint estimations for different regularization parameters.

The results confirm the findings of Scherrer (2010) on discrete MDPs. The MSPBE fixpoint yields a lower MSE than the MSBE in all continuous experiments. MSTDE and MSBE are observed to generate substantially inferior predictions, often with errors almost twice as big. While the MSTDE usually yields the worst predictions, the difference to the MSBE depends on the amount of stochasticity in each transition. For example, both are almost identical on the swing-up task due to low noise in the policy and the MDP. While the MSPBE and MSBE fixpoints are identical to the MSE fixpoint for experiments with perfect feature representations (up to numerical issues, Benchmarks 1,2,7,8), the MSTDE fixpoint is often substantially different (Benchmarks 1,7,8). The problem of a potential dramatic failure of the MSPBE solution, as sketched by Scherrer (2010), was not encountered throughout all evaluations.

Message 2 *Optimizing for the MSBE instead of MSTDE by using double samples introduces high variance in the estimate. Particularly, Bellman residual minimization requires stronger regularization which results in slower convergence than relying on one sample per transition.*

The objective function does not only determine the objective bias but also affects the sampling error (see Figure 10). Using double samples, that is, optimizing for MSBE instead of MSTDE, decreases the objective bias, however, our experiments show that the second

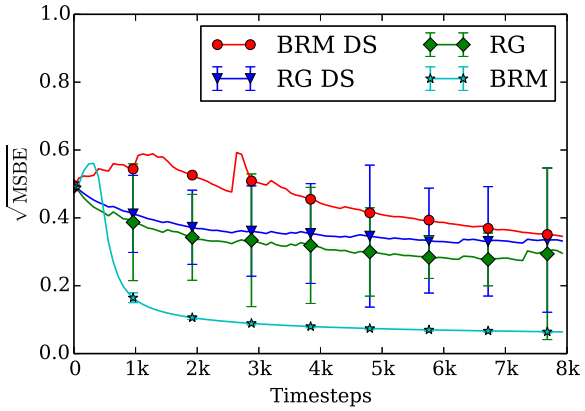


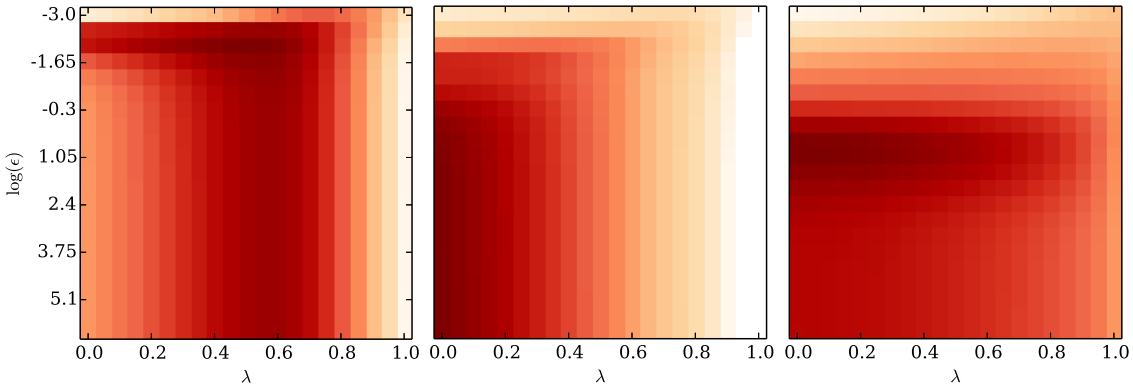
Figure 17: Comparison of double-sampling for BRM and the residual-gradient algorithm in systems with high variance (4. 400-State Random MDP Off-policy). The error bars indicating standard deviation of BRM with double-sampling (BRM DS) are omitted for clarity.

sample per transition is not the only price to pay. To determine whether the sampling error for the two objectives is different, we compared the online performance of residual-gradient algorithm (RG) and Bellman residual minimization (BRM) with or without double sampling (DS). We have observed that double-sampling variants converge significantly slower, that is, their predictions have higher variance. The effect is particularly present in MDPs with high variance such as the random discrete MDPs, see Figure 17. Double-sampling algorithms require stronger regularization and therefore converge slower. Bellman residual minimization suffers more from this effect than the residual-gradient algorithm.

Message 3 *Interpolating between the MSPBE/MSTDE and the MSE with eligibility traces can improve the performance of policy evaluation.*

In Example 2 in Section 2.4.1, we illustrated the benefits of eligibility traces with a specially tailored MDP for which we could control its stochasticity easily. The question remains whether the interpolation between the MSE and MSPBE is also useful for noise levels in MDPs encountered in practice. Is the noise so large that the variance is always the dominant source of error or does reducing the bias with eligibility traces pays off? We therefore compared the MSE of LSTD(λ) and TD(λ) predictions for different λ values on several benchmark tasks. Representative results of LSTD, shown in Figure 18b, confirm that eligibility traces are of no use if no bias is present in the MSPBE due to perfect features. The same holds for systems with large stochasticity such as in the randomly sampled discrete MDP shown in Figure 18c. Yet, interpolating between the MSPBE and MSE boosts the performance significantly if the MSPBE introduces a bias due to an imperfect feature representation and the variance of the MDP is not too high. Such behavior is shown in Figure 18a, where we used the approximate features instead of the perfect feature representation.

Similar to LSTD, TD learning can be improved by eligibility traces as shown for the linearized cart pole balancing benchmark with imperfect features in Figure 19a. Best pre-



(a) Cart-Pole Balancing Imp. Feat. Off-Policy (6.)

(b) Pole Balancing Perf. Feat. Off-Policy (8.)

(c) Discrete Random MDP On-Policy (3.)

Figure 18: Hyper-parameter space of LSTD(λ). Each point is the logarithm of the averaged MSE. Darker colors denote low errors, while white indicates divergence. The colormaps are log log-normalized, that is, the absolute difference in the dark regions are smaller than those in the bright areas. The regularization parameter ϵ is plotted logarithmically on the vertical axis and the eligibility traces parameter λ on the horizontal one.

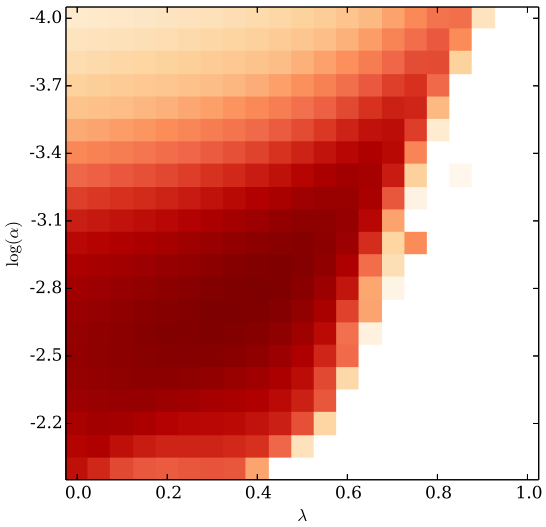
dictions are obtained with $0.1 < \lambda < 0.5$ depending on the step-size α. Yet, different to LSTD, TD learning also benefits from eligibility traces for perfect features (see Figure 19b). They speed up learning by reducing the optimization error of gradient-based approaches (cf. Figure 10) and make the algorithms more robust to the choice of the step-size. These benefits are also present in systems with high stochasticity as Figure 19c indicates. While eligibility traces diminishes the prediction quality of LSTD in such highly stochastic systems, TD learning works best for all λ settings as long as the step-size is set appropriately.

3.2.1 Double Sampling vs. Eligibility Traces

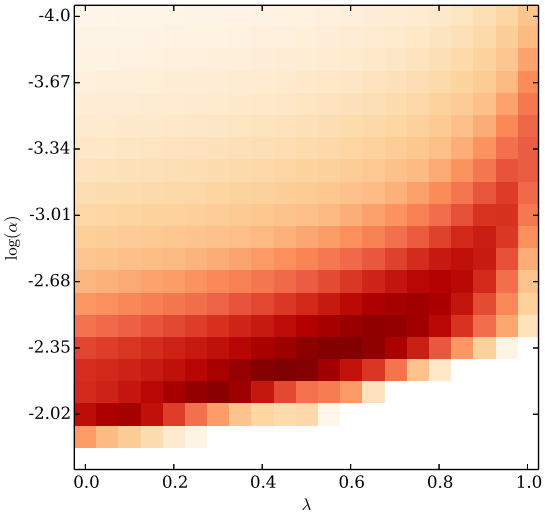
Both, double sampling and eligibility-traces, can be used to reduce the bias of the MSTDE objective at the price of higher variance. However, which approach works better in practice? To shed some light on this matter, we compared BRM with and without double sampling and BRM with eligibility traces. The results for the cart-pole balancing task with imperfect features (Benchmark 5) are shown in Figure 20. We chose the λ-parameter such that both approaches have the same convergence speed, that is, comparable variance. The plot shows that eligibility traces can reduce the bias of the MSTDE more than double sampling at the same increase of variance.

3.2.2 MSPBE vs. MSE Performance

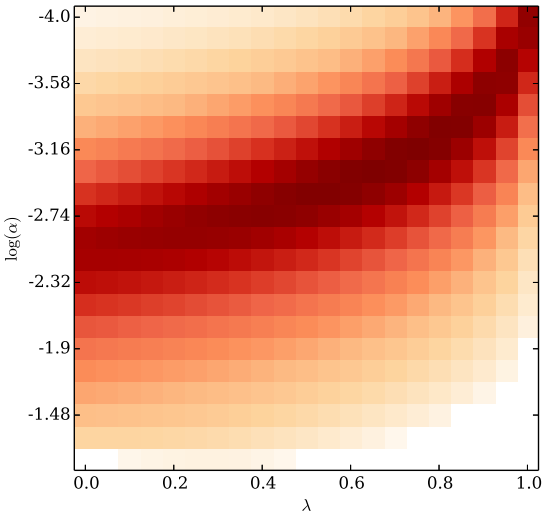
During our evaluation, we often made the interesting observation that a prediction with significantly lower MSPBE than another prediction does not necessarily have a significantly lower MSE. See Figure 21 for such an example, which shows the MSE and MSPBE of predictions for the randomly sampled discrete MDP (Benchmark 3). The performance of



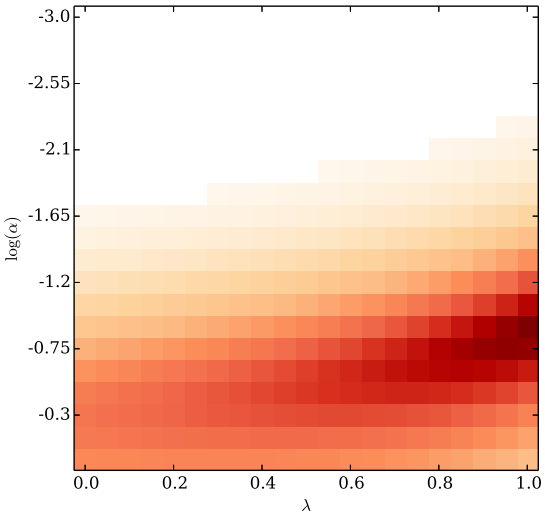
(a) Cart-Pole Balancing Imperf. Feat. Off-Policy (6)



(b) Cart-Pole Balancing Perf. Feat. On-Policy (7)



(c) Discrete Random MDP On-Policy (3)



(d) Boyan Chain (1)

Figure 19: Hyper-parameter space of TD(λ). The color of each point represents the averaged MSE. Darker colors are denoted to low errors, while white indicates divergence. The colormaps are log log-normalized, that is, the absolute difference in the dark regions are smaller than those in the bright areas.

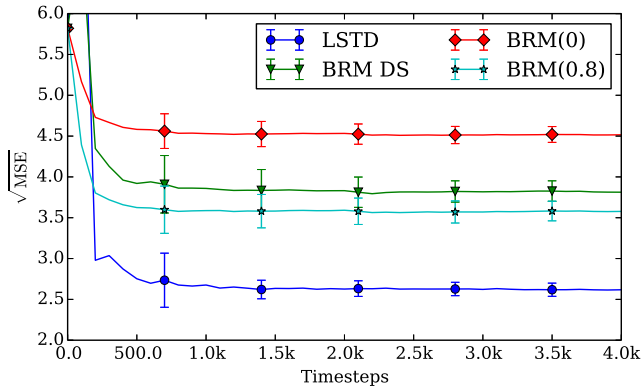


Figure 20: Comparison of bias reduction with double-sampling or eligibility traces for BRM. Eligibility traces introduce less or equal variance than double-sampling and decrease the bias more than double-sampling for linearized cart-pole balancing with imperfect features. Still, LSTD produces less biased predictions at the same level of variance.

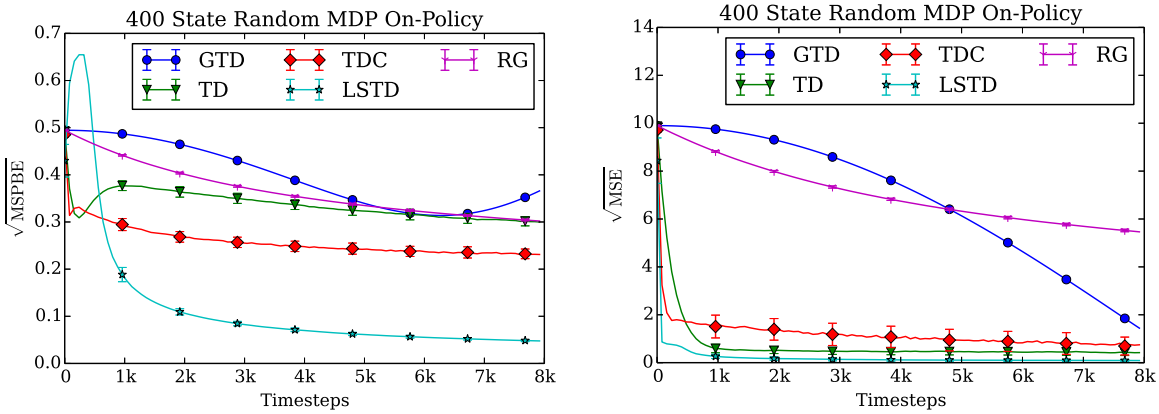


Figure 21: Difference between MSE- and MSPBE-values of the same predictions on the random discrete MDP. Differences w.r.t. MSPBE are not always present in the MSE (see the RG performance).

LSTD and TDC or TD is very different with respect to the MSPBE but they perform almost identically w.r.t. the MSE. While this observation does not always hold (compare for example RG and LSTD), we experienced similar effects in many experiments including continuous MDPs.

3.3 Results on Gradient-based Methods

In this section, we present the most important observations for gradient-based methods.

Message 4 *Normalization of the features is crucial for the prediction quality of gradient-based temporal-difference methods.*

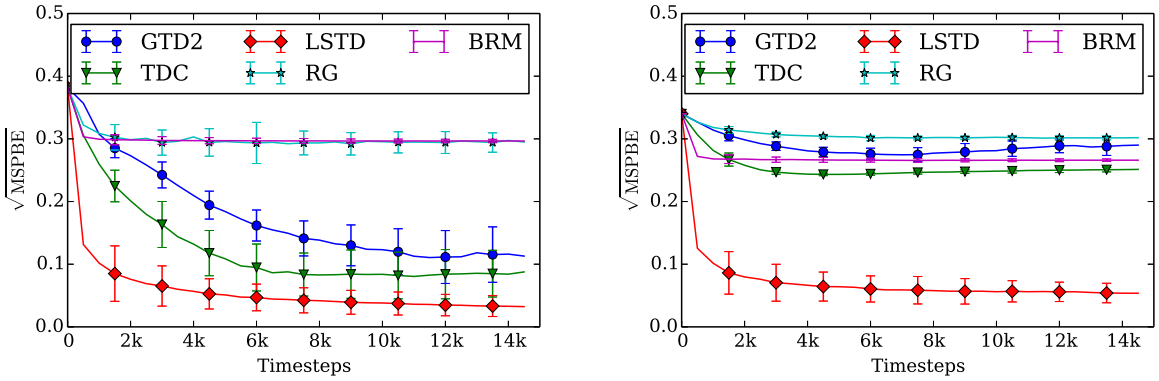
Throughout all experiments, we observed that normalizing the feature representation improves, or least does not harm, the performance of all temporal-difference methods. However, for gradient-based approaches, feature normalization is crucial for a good performance. Features can be normalized *per time step*, for example, all components of the feature vector $\boldsymbol{\phi}_t$ sum up to one, or *per dimension*, for example, each feature is shifted and scaled such that it has mean zero and variance one. Per-time-step normalization is, for example, typically used in radial basis function networks (see Benchmark 11 and 12) to ensure that each time step has the same magnitude of activation and consequently all transition samples have the same weight. As such, its effect resembles that of using natural gradients (Amari, 1998; a discussion of the relation between using the Hessian and natural gradients is provided by Roux and Fitzgibbon, 2010). Since the gradient varies less for different states with per-time-step normalization, the actual distribution of states is less important and the estimate becomes more robust for finitely many samples.

Per-dimension normalization gives each feature comparable importance. Under the assumption that the value function changes similarly fast in each feature dimension, per-dimension normalization causes the Hessian matrix to become more isotropic. It has therefore an effect similar to using the inverse of the Hessian to adjust the gradient as least-squares methods do. However, least-squares methods still can benefit from such a normalization since their regularization has more equate effect on all dimensions.

We compared per-dimension normalized (Figure 22a) and unnormalized features (Figure 22b) for the cart-pole balancing task. The results show that the performance of gradient-based approaches degrade drastically without normalization. As this benchmark requires only little regularization, the performance of least-squared methods is not significantly affected by feature normalization. To understand why normalization plays such an important role for gradient-based methods, consider the MSPBE function in an unnormalized feature space. It may correspond to a quadratic loss function which is flat along some dimension and steep in others, that is, its Hessian contains large and small eigenvalues. Hence, the optimal step-size for gradient descent algorithms can vary significantly per dimension, resulting in either slow convergence of gradient-based algorithms with small step-sizes or a bias if larger step-sizes are used, see, for example, TDC in Figure 22b.

Message 5 *GTD performs worse than its successors GTD2 and TDC. TDC minimizes the MSPBE faster than the other gradient-based algorithms GTD, GTD2 and TD learning.*

We assessed the ability of the gradient based methods TD learning, GTD, GTD2 and TDC to minimize the MSPBE. The results are given in Table 6. Each entry corresponds to the accumulated $\sqrt{\text{MSPBE}}$-values of the predictions for all time steps. Low numbers indicate accurate estimates with small number of samples. We observe that the performance of GTD is always worse than the performance of the other methods except for the Boyan chain (Benchmark 1) and the 20-link Pole Balancing Off-policy task (Benchmark 12). GTD2 converged very slowly in these two experiments. Throughout all tasks, GTD performs significantly worse than other approaches and yields unreliable results in general, that is, sometimes the



(a) Normalized Features.

(b) Unnormalized Features.

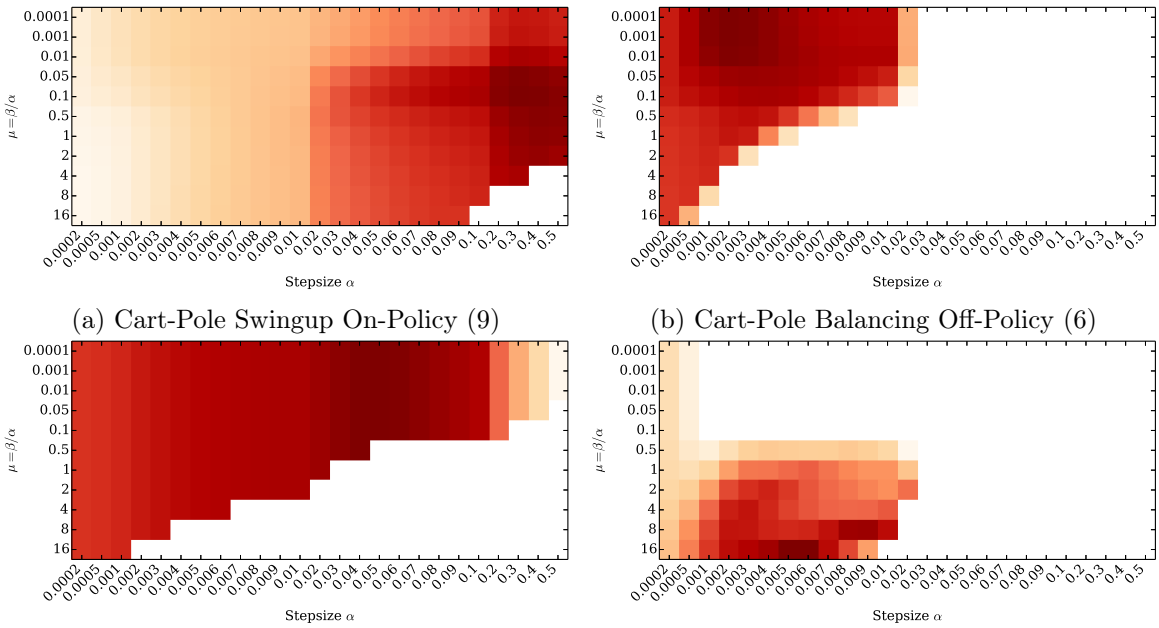
Figure 22: Comparison for the cart pole balancing task (Benchmark 5) with normalized and unnormalized features. Differences in the magnitude of features are particularly harmful for gradient-based approaches.

	GTD	GTD2	TDC	TD
1. 14-State Boyan Chain	58.11	48.65	**16.51**	16.51
2. Baird Star Example	1504.38	1520.09	**1237.70**	$> 10^{10}$
3. 400-State Random MDP On-policy	39.58	31.06	**25.90**	33.62
4. 400-State Random MDP Off-policy	40.90	38.08	**30.50**	37.08
5. Lin. Cart-Pole Balancing On-pol. Imp. Feat.	8.56	5.37	**3.84**	3.84
6. Lin. Cart-Pole Balancing Off-pol. Imp. Feat.	35.86	21.05	**13.31**	13.31
7. Lin. Cart-Pole Balancing On-pol. Perf. Feat.	10.18	8.07	**7.07**	7.98
8. Lin. Cart-Pole Balancing Off-pol. Perf. Feat.	20.65	18.40	**15.47**	19.68
9. Cart-Pole Swingup On-policy	40.21	24.66	**23.08**	25.60
10. Cart-Pole Swingup Off-policy	41.09	30.44	**25.28**	30.14
11. 20-link Lin. Pole Balancing On-pol.	21.97	20.24	**17.22**	18.58
12. 20-link Lin. Pole Balancing Off-pol.	0.39	0.43	**0.26**	0.30

Table 6: Sum of square root MSPBE for all timesteps of GTD, GTD2, TDC and TD learning (TD). GTD is observed to always yield the largest error except for Benchmark 2 and 12. TDC outperformed the other methods in all experiments. The values are obtained after optimizing the hyper-parameters for the individual algorithms.

estimates have a drastically higher error (Benchmark 1, 5, 6). TDC outperformed all other gradient-based methods in minimizing the MSPBE throughout all tasks.

Message 6 *If we optimize the hyper-parameters of TDC, TDC is always at least as good as TD-learning, but comes at the price of optimizing an additional hyper-parameter. Often, hyper-parameter optimization yields very small values for the second learning rate β, in which case TDC reduces to TD-learning.*



(a) Cart-Pole Swingup On-Policy (9)

(b) Cart-Pole Balancing Off-Policy (6)

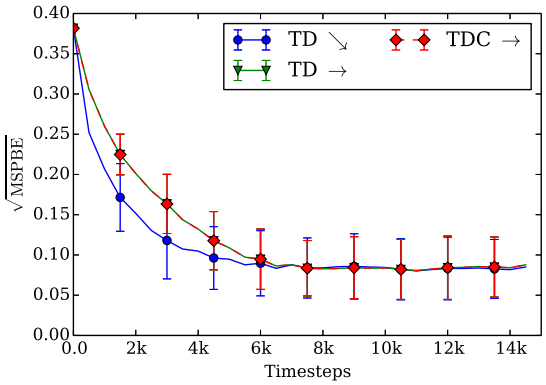
(c) 20-link Pole Balancing Off-Policy (12)

(d) Baird's Star Example (2)

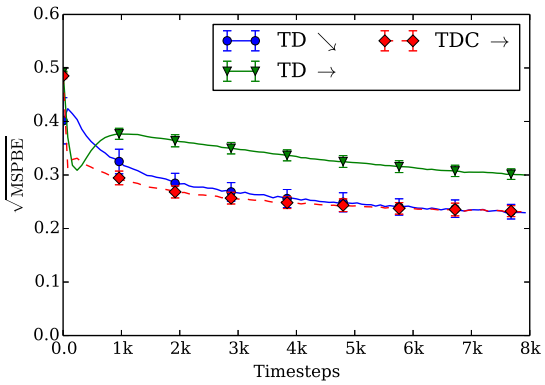
Figure 23: Hyper-parameter space of TDC for $\lambda = 0$. The primary step-size of TDC is denoted by α and $\mu = \beta/\alpha$ is the ratio of secondary to primary step-size. Each point is the logarithm of the averaged MSPBE. Darker colors are denoted to low errors, while white indicates divergence.

As TDC is identical to TD if we set the second learning rate β for the vector $\boldsymbol{w}$ (or the ratio $\mu = \beta/\alpha$) to zero, the performance of TDC is at least as good as that of TD if the hyper-parameters are optimized. As the results in Table 6 show, the difference of TDC and TD is negligible in some tasks (Tasks 1, 5, 6). In these cases, the optimal values for the ratio μ are very small, as the results of the grid-search in Figure 23b indicate, and TDC reduces to TD.

Large μ values were only observed to yield good performance for the Baird's Star task (Benchmark 2) where TD learning always diverged (see Figure 23d). Apart from this example, which is specifically tailored to show divergence of TD learning, TD converged in all off-policy benchmarks. However, even if TD learning converges, the use of the second step-size can be beneficial in some scenarios (e.g., in Benchmark 3, 4, 8, 9 and 10), as the MSPBE of the predictions can be reduced significantly. The grid search results of the Swing-up task shown in Figure 23a as well as the prediction error over time shown in Figure 24b clearly indicate an advantage of TDC. However, TDC comes at the price that we need to optimize the second learning rate as an additional hyper-parameter despite that it may have almost no effect in some problems (see Figure 23c).



(a) Cart-pole Balancing With Imperfect Features, On-policy. (Benchmark 6). The graphs of TDC and TD with constant step-sizes are identical.



(b) Discrete Random MDP, On-policy (Benchmark 3)

Figure 24: Convergence Speed for TD learning with decreasing (TD ↘) and constant step-sizes (TD →) and TDC with constant step-sizes (TDC →).

3.3.1 Constant vs. Decreasing Learning Rates

We also evaluated whether using a decreasing learning rate—an assumption on which the convergence proofs of stochastic gradient-based methods rely—improves the prediction performance for a finite number of time steps. We compared constant learning rates against exponentially decreasing ones (see C and D in Table 4) for TD learning. In most tasks, no significant improvements with decreasing rates could be observed. Only for the cart pole balancing with imperfect features (Benchmark 6) and the discrete random MDP (Benchmark 3), we could speed up the convergence to low-error predictions. Figure 24 illustrates the difference. However, using decreasing learning rates is harder as at least two parameters per learning rate need to be optimized, which we experienced to have high influence on the prediction quality, and, hence, we do not recommend to use decreasing step-sizes for a limited number of observations.

3.3.2 Influence of Hyper-Parameters

We can consider the prediction error of each method as a function of the method's hyper-parameters. As Figure 19 and Figure 23 indicate, these functions are smooth, uni-modal and often even convex for gradient-based algorithms.[9] Only parts of the hyper-parameter spaces are shown, yet, we observed the functions to be well-behaved in general, and, hence, local optimization strategies for selecting hyper-parameters can be employed successfully.

3.4 Results on Least-Squares Methods

In this section, we present the most important insights from the experimental evaluation of least-squares methods.

9. The plots in Figure 19 seem non-convex due to the log-scale of the step-size parameter α.

Task	GTD	GTD2	TD	TDC	RG	RG DS	BRM	BRM DS	LSPE	LSTD	FPKF
1	7.22	6.89	5.56	0.40	5.56	6.83	2.32	0.26	**0.10**	**0.10**	0.79
2	1.74	1.63	0.03	0.03	1.56	2.20	**0.00**	0.00	0.03	0.03	0.22
3	1.44	1.36	1.05	0.75	5.46	2.32	0.12	1.44	**0.09**	**0.09**	4.34
4	1.39	1.97	0.93	1.29	3.10	2.95	**0.10**	3.20	0.13	0.13	9.72
5	2.37	**2.37**	3.58	2.51	4.42	3.75	4.52	3.80	2.60	2.60	2.58
6	2.59	**2.33**	4.37	2.44	4.42	3.88	4.37	3.82	2.47	2.47	2.91
7	5.45	3.12	**0.15**	1.75	3.14	1.19	**0.15**	**0.15**	**0.15**	**0.15**	0.24
8	5.43	4.04	1.95	2.10	3.18	1.53	1.95	1.95	**0.17**	**0.17**	3.82
9	5.13	3.98	3.83	3.86	4.61	4.60	3.83	3.82	**1.97**	1.99	2.88
10	5.45	4.85	4.28	3.91	4.71	4.71	4.28	4.30	4.68	**2.17**	4.28
11	4.29	4.41	7.71	4.75	7.60	7.44	7.71	7.45	**4.26**	4.27	7.30
12	0.058	0.08	0.077	0.052	0.077	0.075	0.077	0.076	0.043	**0.042**	0.081

Table 7: Mean squared errors of final predictions. Task names and descriptions associated with the numbers can be found in Section 3.1. LSPE and LSTD are shown with transition-based off-policy reweighting (LSPE-TO and LSTD-TO).

Message 7 *In general, LSTD and LSPE produce the predictions with lowest errors for sufficiently many observations.*

Table 7 shows the square roots of mean-squared errors ($\sqrt{\text{MSE}}$) of the estimate from each method at the last timestep. The values are the final errors obtained with specific methods for each benchmark. The best prediction is generated either by LSTD or LSPE for almost all tasks. In cases where LSPE has the lowest error, LSTD is only marginally worse and does not depend on a learning rate α to be optimized.

The Star Example of Baird has a true value function of constant $\mathbf{0}$ and is therefore not suited to compare least-squares methods. Each least-squares method can yield perfect results with overly strong regularization. The small difference between the errors of BRM and LSTD in the second row of Table 7 are caused by numerical issues avoidable by stronger regularization.

Interestingly, BRM outperforms all other methods in Benchmark 4, the randomly generated discrete MDP with off-policy samples. The MSTDE has a high bias but at the same time a low variance which seems to be particularly advantageous here as this benchmark is highly stochastic. We experienced unexpected results for the cart-pole balancing tasks with imperfect features (Tasks 5 and 6). Here, the gradient-based approaches perform exceedingly well and GTD even obtains the lowest final MSE value. However, Figure 25 reveals that the reason for this effect is only an artifact of the optimization error introduced by gradient methods. The two sources of error, objective bias and optimization error, counterbalance each other in this example. The MSPBE fixpoint has an error of about 2.5 and LSTD converges to it quickly. The gradient methods, however, converge slower due to their optimization error. Yet, when they approach the MSPBE fixpoint, the estimates pass through regions with lower MSE. As GTD has not converged for the maximum number of evaluated observations, it is still in the region with lower MSE. It therefore yields the best

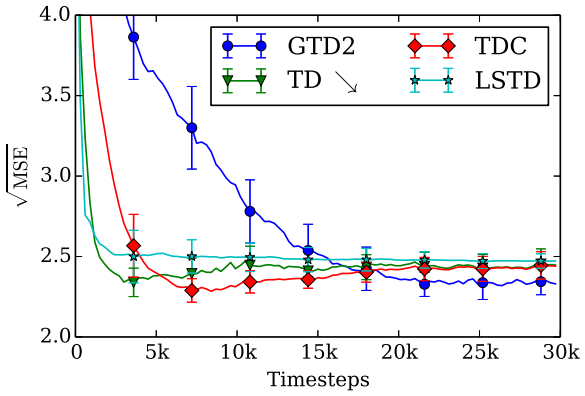


Figure 25: Pole Balancing task with impoverished features. Slightly sub-optimal prediction with respect to the MSPBE yield lowest MSE.

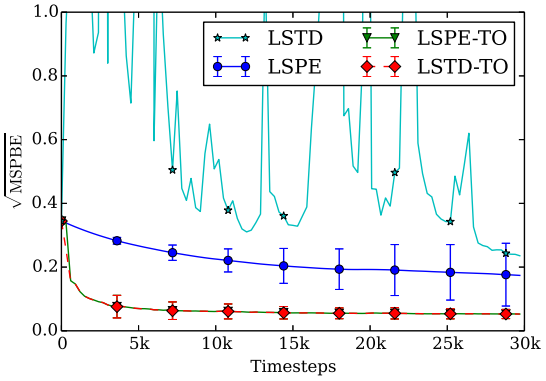
final prediction. However, it would eventually converge to the worse MSPBE fixpoint. Unfortunately, we typically do not have knowledge when such a coincidence happens without actually evaluating the MSE, and, thus, can not exploit this effect. Apart from Tasks 4, 5 and 6, LSTD always yields the lowest or almost the lowest errors, while other methods perform significantly worse on at least some benchmarks (e.g., Task 10 for LSPE). According to the results of our experiments, LSTD is a very accurate and reliable method. It is the method of our choice, although it may behave unstable in some cases, for example, when the number of observations is less than the number of features or some features are redundant, that is, linearly dependent.

Message 8 *In practice, LSTD and LSPE perform well with off-policy samples only if the newly introduced transition reweighting (proposed in Section 2.4.2) is used. The variance of LSTD with standard reweighting makes the algorithm unusable in practice.*

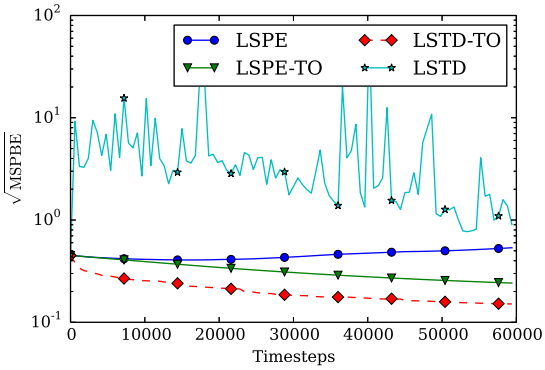
In Section 2.4.2, we proposed an off-policy reweighting based on the entire transition as an alternative to the standard off-policy sample reweighting for LSTD and LSPE. Both reweighting strategies converge to the same solution for infinitely many observations. However, transition reweighting yields much faster convergence as Figure 26 illustrates. Figure 26a compares the reweighting approaches on the linearized cart-pole problem (Benchmark 6). Despite strong ℓ_2-regularization, LSTD yields very noisy estimates with standard reweighting, rendering the algorithm inapplicable. The variance induced by the standard reweighting prevents fast convergence, as the variance of 1.0 between different experiment runs indicates. While the estimates of LSPE with standard reweighting show decreasing error over time (due to a very small step size chosen by the hyper-parameter optimization), the increasing standard deviation indicates that the variance of the estimates is problematic. In contrast, LSPE and LSTD with transition reweighting converge substantially faster and yield good estimates after already 5,000 time steps. The benefit of transition reweighting is even more salient in the results of the off-policy cart-pole swing-up task (Benchmark 10) shown in Figure 26b on a logarithmic scale. The observations are consistent with all off-policy tasks (see Table 8). The price for using off-policy samples instead of on-policy samples, in terms of convergence

	LSPE	LSPE-TO	LSTD	LSTD-TO
4. 400-State Random MDP Off-policy	110.16	21.30	1727.10	**15.50**
6. Lin. Cart-Pole Balancing Off-pol. Imp. Feat.	22.08	6.88	223.64	**6.79**
8. Lin. Cart-Pole Balancing Off-pol. Perf. Feat.	15.52	4.38	49.27	**3.52**
10. Cart-Pole Swingup Off-policy	45.26	32.11	493.18	**20.58**
12. 20-link Lin. Pole Balancing Off-pol.	0.43	**0.24**	0.43	0.32

Table 8: Sum of square-roots of MSPBE for all timesteps of LSPE and LSTD with standard importance reweighting and transition off-policy reweighting (LSPE-TO, LSTD-TO). The Baird-Star Example (Task 2) is omitted as it is not suited well for evaluating least-squares approaches since perfect estimates can be achieved with overly strong regularization.



(a) Cart pole balancing with 5 features, off-policy (Benchmark 6). The error-bars of LSTD with standard reweighting are omitted for visibility.



(b) Cart pole swingup, off-policy (Benchmark 10) on a logarithmic scale.

Figure 26: Comparison of the standard off-policy reweighting scheme and the transition reweighting (TO) proposed in this paper.

speed, is similar to other methods if LSTD and LSPE are used with transition reweighting. LSTD and LSPE with transition reweighting obtain the most accurate estimates of all methods as Table 7 indicates.

Message 9 *For a modest number of features, least-squares methods are superior to gradient-based approaches both in terms of data-efficiency and even CPU-time if we want to reach the same error level. For a very large number of features (e.g., $\geq$ 20,000), gradient-based methods should be preferred as least-squares approaches become prohibitively time- and memory-consuming.*

Except for the artifact in the linearized cart-pole balancing task with imperfect features (Benchmarks 5 and 6), least-squares methods yield the most accurate final predictions (see Table 7). However, least-squares approaches may behave very unstable for a small number of

observations. A strong regularization is usually required if the number of samples is smaller than the number of features. An example is given in Figure 17 that shows the increase of the error of BRM predictions for the first 500 samples. However, Figures 22a, 21, 17 and 25 show that least-squares approaches converge much faster than gradient-based methods after this stage of potential instability. Least-squares methods are therefore more data efficient than gradient-based methods.

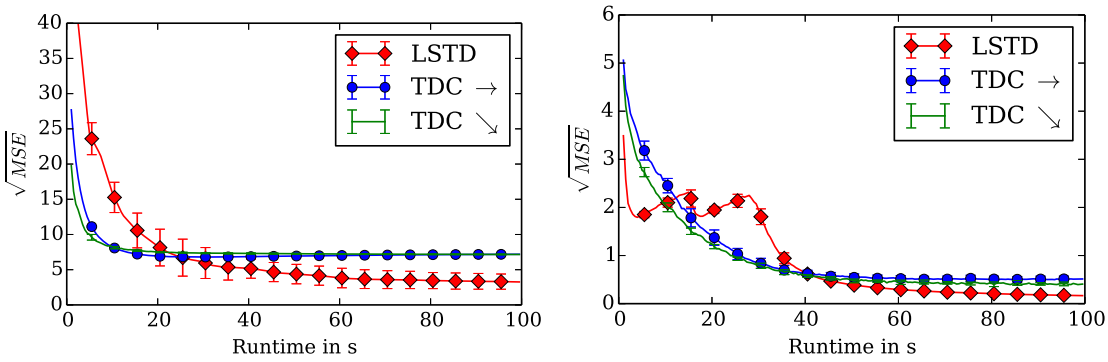
However, the required CPU-time per transition is quadratic in the number of features instead of linear as for the gradient approaches. Hence, it is also interesting to compare both approaches from a computational viewpoint with a fixed budget of CPU-time. Figure 27 compares the prediction quality of LSTD and TDC, the best-performing representatives of both classes, for a given budget of CPU-time. In order to compare the performance on a task with a vast number of features, we changed the number of dimensions in the pendulum balancing task from 20 to 100 and used the perfect feature representation of 20101 dimensions.[10] LSTD requires more CPU time to converge since TDC can do several sweeps through the provided transition samples (7000 in total) while LSTD can update the parameters only a few times due to the high-dimensional features. Yet, TDC converges faster only up to an error level of approximately 8, both for constant and decreasing step-sizes. The prediction error of TDC with decreasing step-sizes still decreases, and will eventually reach the same minimum as the one of LSTD, but very slowly. This observation is consistent with results on stochastic gradient methods in general (see Sra et al., 2012, Chapter 4.1.2). Additionally, we evaluated the methods on a 30-link pendulum with a moderate number of 1830 features. The results are shown in Figure 27b. Due to the smaller number of features LSTD converges faster from the beginning on. However, as the stagnant prediction error up to second 30 shows, LSTD may still yield unstable results as it has not processed enough observations to ensure that its $\boldsymbol{A}_t$-matrix (cf. Equation 25) is invertible and needs strong regularization.

Least-squares methods clearly outperform gradient-based approaches for problems with few features. The quadratic run-time and memory consumption as well as the unstable behavior with few observations become more problematic for increasing numbers of features but, as our results show, least-squares methods may still be a good option for up to 20.000 features

3.4.1 Alternative Regularization Approaches

We implemented and evaluated alternative regularization approaches for LSTD in preliminary experiments, including LARS-TD, LSTD with ℓ_2, ℓ_1, LSTD-ℓ_1 and D-LSTD. However, we observed no performance gain for our benchmarks in comparison to ℓ_2-regularization. We attribute this result to the fact that most of the features in our benchmarks had sufficient quality. The sparse solutions produced by alternative regularization schemes had thus no significant advantage as the noise introduced by active low-quality features in non-sparse solutions was not large enough. We also did not observe the theoretically derived benefits of LSTD with random projections (LSTD-RP), which only become important for extremely many features.

10. The features include the products of all state variables, cf. the features of Benchmark 7.



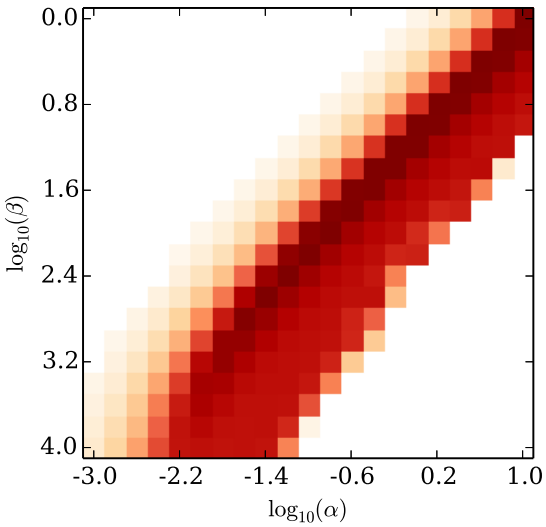
(a) 100-link Pole Balancing with 20101 features (400 dim. state, 20100 squared state features + 1 constant dim.)

(b) 30-link Pole Balancing with 1831 features (60 dim. state, 1830 squared state features + 1 constant dim.)

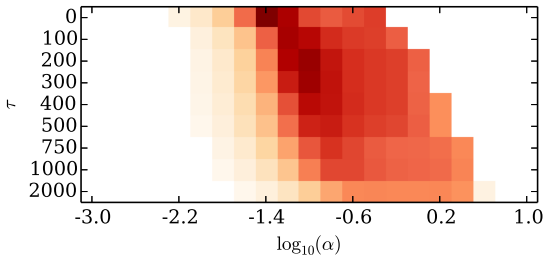
Figure 27: Comparison of the prediction quality for given CPU times of LSTD and TDC with constant (TDC →) and decreasing step-sizes (TDC ↘). The methods are evaluated on multi-link Pole Balancing tasks (in analogy to Benchmark 11) with perfect feature representations. The methods are provided with a total of 7000 transitions. The results are averages of 10 independent runs executed on a single core of an i7 Intel CPU.

3.4.2 Dependency on Hyper-Parameters

Most least-squares methods are robust against a poor choice of the hyper-parameters. LSTD and BRM, in particular, which are controlled only by the regularization parameter ϵ and the parameter λ of the eligibility-traces, converge for almost all values (cf. Figure 18). In contrast, FPKF has four hyper-parameters to optimize, the eligibility-trace parameter λ; a general scaling factor α_{FPKF} for the step-sizes; β_{FPKF} delaying the decrease of the step-length and τ_{FPKF} which controls the minimum number of observations necessary to update the estimate. In particular, α_{FPKF} and β_{FPKF} need to be set correctly to prevent FPKF from diverging as the large white areas in Figure 28a indicate. Also, τ_{FPKF} has large influence on the performance and may cause divergence (Figure 28b). Hence, descent-based optimization strategies such as block gradient descent (with finite differences) are difficult to use for hyper-parameter search as choosing an initial hyper-parameter setting that works is not trivial. On the contrary, LSPE does not rely as much on well-chosen hyper-parameters. As LSTD and BRM, it has an eligibility-traces parameter λ and a regularization-intensity ϵ, but also incorporates a step-size α, which affects the prediction in a similar way as ϵ, that is, it controls the amount of learning. Representative results, given in Figure 29, illustrate that LSPE performs well and stable up to a certain step-size but then diverges. Still, LSTD does not require any step-size at all and performs similarly or better than LSPE (see also Message 7).



(a) Step-size scale α_{FPKF} and length-decreasing parameter β_{FPKF}



(b) Step-size scale α_{FPKF} and the initial update delay τ_{FPKF}

Figure 28: Hyper-parameter space of FPKF(λ) for Benchmark 5. Each point is the logarithm of the averaged MSE. Darker colors show lower errors, while white indicates divergence. The color-maps are log-normalized. Each plot shows a slice of the 4-dimensional hyper-parameter space around the setting used in our experiments (optimal w.r.t. the MSPBE).

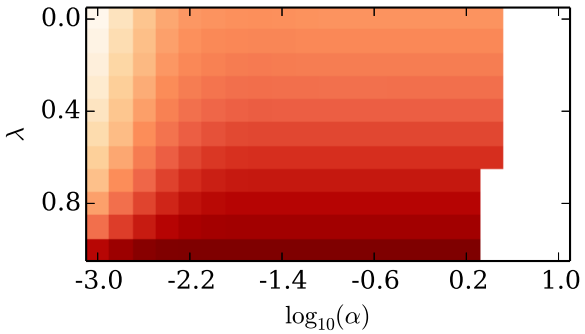


Figure 29: Hyper-parameter space of LSPE(λ) for Benchmark 5. Each point is the logarithm of the averaged MSE. Darker colors show lower errors, while white indicates divergence. The color-maps are log-normalized. The step-size α has little influence on the performance, while e-traces may speed-up learning significantly.

4. Conclusion and Outlook

We conclude the paper with a short summary of its main contributions and a brief outlook on possible directions for future research on temporal-difference methods.

4.1 Conclusion

With this paper, we aimed at giving an exhaustive survey of past and current research activities on value-function estimation with temporal differences—both from a theoretical and an empirical point of view. Almost all important methods originated in this area of research have been presented from a unifying viewpoint of function optimization. The algorithms have been systematically categorized based on their underlying cost functions and the employed optimization technique: stochastic gradient descent, analytic least-squares solutions or a probabilistic problem formulation. We aimed for a concise, yet comprehensive and coherent presentation to make these algorithms available to novices and practitioners.

In addition, we provide an overview over recent work on feature representations for value function approximation. These developments aim either at an automatic generation of state features or at more robust methods that can deal with very large numbers of irrelevant features. We also have provided a qualitative analysis of conceptual error sources that aids novices in understanding the effects of eligibility traces which implicitly perform multi-step look-ahead. Discerning the sources of errors is important for choosing the most suitable estimation method given a new task at hand and for identifying reasons why a particular approach might not work. Furthermore, we have discussed the use of importance reweighting for implementing off-policy value estimation. We have shown that the commonly employed importance reweighting strategy of least-squares methods such as LSTD and LSPE is unsuitable for non-trivial tasks due to its high variance. To alleviate this problem, we have introduced a novel importance reweighting strategy with drastically reduced variance. Our importance reweighting strategy works well in practice—even where the standard reweighting strategy exhibits strong instabilities and yields unsuitable results.

One of the most important contribution of this paper is that it is one of the first comprehensive experimental comparisons of the different value-function estimation methods, including the recent developments. Their performance was evaluated on 12 different benchmark tasks that exhibited different characteristics, including MDPs with continuous and discrete state spaces as well as on- and off-policy transition samples. Our work also provides further evidence on relevant future research questions, such as, which objective function has the lowest bias in practice or which algorithms are preferable in terms of data efficiency or computational demands. Moreover, the experimental evaluation reveals the behavior and the limitations of each temporal-difference method in various scenarios. These findings will hopefully give insights for improving the state of the art in policy evaluation.

4.2 Outlook

Efficient estimation of value functions is a corner stone of reinforcement learning as the value function is needed in the policy improvement step of the many reinforcement learning algorithms. Temporal-difference methods have been used since the late 1980s to estimate the value function of a policy and have since then been an active field of research. In recent

years, the research concentrated on overcoming the instability of the TD learning method with off-policy samples, improving the sample efficiency, or value-function estimation in high-dimensional feature spaces. This research resulted in the development of the current state of the art, such as LSTD or TD learning with Gradient Correction. In addition, the theoretical analysis from different view-points has led to a good theoretical understanding of the foundations of temporal-difference methods.

However, the use of temporal-difference methods has been restricted by several limitations of the methods. In many domains, the assumption of having a compact and informative feature representation is not realistic. Such features are often difficult to define by hand, for example, in real world robotic systems, in health care diagnosis or for controlling of prostheses. We therefore expect the recent efforts of learning the feature representation (cf. Section 2.3) to continue and increase substantially.

Additionally, the number of samples necessary for learning value functions is still prohibitively large for many real-world scenarios, especially those that involve hardware such as robots or other complex, expensive equipment. One way for addressing this shortcoming is to make the learning problem easier by incorporating prior knowledge. Domain knowledge can usually be incorporated most easily in model-based methods that learn the underlying forward dynamics of the task. Such models can be used to create samples in simulation without hardware or by directly using dynamic programming with the learned model. Deisenroth and Rasmussen (2011) successfully learned complex robot control policies by simultaneously learning a system model and using this model to optimize the policy (direct model-based policy search). As their work show, it is crucial to include the uncertainty about the learned model into the actual estimation problem. Value-function estimation methods that can incorporate a learned model and its uncertainty efficiently are therefore a promising direction for future research.

In this paper, we have treated minimizing the mean squared error of a value function estimate as the ultimate goal. However, in most cases the value function is only an intermediate result used for improving the policy. What counts in the end is not the quality of the value function approximation but the quality of the policy after the policy improvement step. Other objective functions might exist that are easier to minimize and do not harm the convergence properties of policy iteration schemes. Characterizing such objectives can prospectively lead to algorithms with faster convergence to an optimal or at least viable policy.

This survey and comparison solely concentrated on policy evaluation. A comprehensive survey and experimental evaluation of temporal differences in policy iteration which builds on the results of this paper is left as future work and strongly needed by the reinforcement learning community.

Acknowledgments

The research leading to these results has received funding from the European Community's Framework Programme CompLACS (FP7-ICT-2009-6 Grant.No.270327).

Appendix A. Derivation of Least-Squares Temporal-Difference Learning

The derivation of the LSTD algorithm begins with rewriting the MSPBE from Equation (9) in terms of a different norm. While this formulation is, strictly speaking, not necessary, it helps to understand the connection to the TDC, GTD and GTD2 algorithms and let us derive subsequent steps more concisely

$$\begin{aligned}
\text{MSPBE}(\boldsymbol{\theta}) &= \|\boldsymbol{V}_{\boldsymbol{\theta}} - \Pi T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}}\|_{\boldsymbol{D}}^2 \\
&= \|\Pi(\boldsymbol{V}_{\boldsymbol{\theta}} - T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}})\|_{\boldsymbol{D}}^2 \quad \text{(since } \boldsymbol{V}_{\theta} \text{ is parametrizable)} \\
&= [\Pi(\boldsymbol{V}_{\boldsymbol{\theta}} - T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}})]^T \boldsymbol{D} [\Pi(\boldsymbol{V}_{\boldsymbol{\theta}} - T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}})] \\
&= [\boldsymbol{V}_{\boldsymbol{\theta}} - T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}}]^T \Pi^T \boldsymbol{D} \Pi [\boldsymbol{V}_{\boldsymbol{\theta}} - T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}}] \\
&= [\boldsymbol{V}_{\boldsymbol{\theta}} - T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}}]^T \boldsymbol{D} \boldsymbol{\Phi} (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi} (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} [\boldsymbol{V}_{\boldsymbol{\theta}} - T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}}] \\
&= [\boldsymbol{V}_{\boldsymbol{\theta}} - T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}}]^T \boldsymbol{D} \boldsymbol{\Phi} (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} [\boldsymbol{V}_{\boldsymbol{\theta}} - T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}}] \\
&= [\boldsymbol{\Phi}^T \boldsymbol{D} (\boldsymbol{V}_{\boldsymbol{\theta}} - T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}})]^T (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1} [\boldsymbol{\Phi}^T \boldsymbol{D} (\boldsymbol{V}_{\boldsymbol{\theta}} - T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}})] \\
&= \|\boldsymbol{\Phi}^T \boldsymbol{D} (\boldsymbol{V}_{\boldsymbol{\theta}} - T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}})\|_{(\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1}}^2 \qquad (41) \\
&= \|\mathbb{E}_{d,\pi,\mathcal{P}}[\boldsymbol{\phi}_t \delta_t]\|_{(\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1}}^2 . \qquad (42)
\end{aligned}$$

The matrix $\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi}$ and its inverse $\boldsymbol{M} = (\boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{\Phi})^{-1}$ are symmetric positive definite matrices (for independent features and $d(s) > 0, \forall s \in \mathcal{S}$). Hence, $\| \cdot \|_{\boldsymbol{M}}$ is a norm and $\boldsymbol{\theta}$ minimizes the MSPBE if and only if $\mathbb{E}_{d,\pi,\mathcal{P}}[\boldsymbol{\phi}_t \delta_t] = 0$. Equation (42) also allows us to rewrite the MSPBE as a product of expectations

$$\text{MSPBE}(\boldsymbol{\theta}) = \mathbb{E}_{d,\pi,\mathcal{P}}[\boldsymbol{\phi}_t \delta_t]^T \mathbb{E}_d[\boldsymbol{\phi}_t \boldsymbol{\phi}_t^T]^{-1} \mathbb{E}_{d,\pi,\mathcal{P}}[\boldsymbol{\phi}_t \delta_t], \qquad (43)$$

which is the basis for the GTD2 and TDC algorithms. Since $V_{\boldsymbol{\theta}}$ is parameterized linearly, we can replace $T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}}$ with

$$T^{\pi} \boldsymbol{V}_{\boldsymbol{\theta}} = \boldsymbol{R} + \gamma \boldsymbol{\Phi}' \boldsymbol{\theta},$$

where $\boldsymbol{R} \in \mathbb{R}^n$ is the expected intermediate reward $\boldsymbol{R}_i = \mathbb{E}_{\pi}[r(s^i, a)]$ in state s^i and $\boldsymbol{\Phi}' = \boldsymbol{P}^{\pi} \boldsymbol{\Phi}$ is the matrix containing the expected feature for the successor states, that is, $\boldsymbol{\Phi}'_i = \mathbb{E}_{\mathcal{P},\pi}[\boldsymbol{\phi}_{t+1}^T | s_t = s^i]$. Equation (41) can then be written as

$$\begin{aligned}
\text{MSPBE}(\boldsymbol{\theta}) &= \|\boldsymbol{\Phi}^T \boldsymbol{D} (\boldsymbol{\Phi} \boldsymbol{\theta} - \gamma \boldsymbol{\Phi}' \boldsymbol{\theta} - \boldsymbol{R})\|_{\boldsymbol{M}}^2 \\
&= \|\boldsymbol{\Phi}^T \boldsymbol{D} [\boldsymbol{\Phi} - \gamma \boldsymbol{\Phi}'] \boldsymbol{\theta} - \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{R})\|_{\boldsymbol{M}}^2 \\
&= \|\boldsymbol{\Phi}^T \boldsymbol{D} \Delta \boldsymbol{\Phi} \boldsymbol{\theta} - \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{R})\|_{\boldsymbol{M}}^2 \\
&= \|\boldsymbol{A} \boldsymbol{\theta} - \boldsymbol{b}\|_{\boldsymbol{M}}^2,
\end{aligned}$$

where $\Delta \boldsymbol{\Phi} = \boldsymbol{\Phi} - \gamma \boldsymbol{\Phi}'$ and $\boldsymbol{b} = \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{R}$. The matrix $\boldsymbol{A} = \boldsymbol{\Phi}^T \boldsymbol{D} \Delta \boldsymbol{\Phi}$ has been shown to be positive definite and thus invertible (Bertsekas and Tsitsiklis, 1996, Proposition 6.3.3). Minimizing this MSPBE formulation directly by setting the gradient to 0 yields

$$\boldsymbol{\theta} = (\boldsymbol{A}^T \boldsymbol{M} \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{M} \boldsymbol{b} = \boldsymbol{A}^{-1} \boldsymbol{M}^{-1} \boldsymbol{A}^{-T} \boldsymbol{A}^T \boldsymbol{M} \boldsymbol{b} = \boldsymbol{A}^{-1} \boldsymbol{b} = (\boldsymbol{\Phi}^T \boldsymbol{D} \Delta \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{D} \boldsymbol{R}.$$

Appendix B. Parametric GPTD Whitening Transformation

Equation (32) can also be written in matrix form in terms of the reward, that is,

$$\boldsymbol{r}_{t-1} = \boldsymbol{\Gamma}_t \boldsymbol{V}_t + \boldsymbol{n}_t, \qquad \boldsymbol{\Gamma}_t = \begin{bmatrix} 1 & -\gamma & 0 & \dots & 0 \\ 0 & 1 & -\gamma & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & \dots & 0 & 1 & -\gamma \end{bmatrix},$$

where $\boldsymbol{\Gamma}_t$ connects the values of subsequent timesteps, $\boldsymbol{r}_{t-1} = [r_1, \dots, r_{t-1}]$ and $\boldsymbol{V}_t = [v_1 \dots v_t]$. The noise term $\boldsymbol{n}_t$ is now given as $\boldsymbol{n}_t = \boldsymbol{\Gamma}_t \Delta \boldsymbol{v}_t$, and hence is distributed as $\boldsymbol{n}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_t)$, with

$$\boldsymbol{\Sigma}_t = \sigma^2 \boldsymbol{\Gamma}_t \boldsymbol{\Gamma}_t^T = \sigma^2 \begin{bmatrix} 1+\gamma^2 & -\gamma & 0 & \dots & 0 \\ -\gamma & 1+\gamma^2 & -\gamma & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & -\gamma & 1+\gamma^2 \end{bmatrix}.$$

We realize that the required whitening transformation is given by

$$\boldsymbol{Z}_t = \boldsymbol{\Gamma}_t^{-1} = \begin{bmatrix} 1 & \gamma & \gamma^2 & \dots & \gamma^t \\ 0 & 1 & \gamma & \dots & \gamma^{t-1} \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix},$$

and hence, we get the following regression problem

$$\boldsymbol{Z}_t \boldsymbol{r}_{t-1} = \boldsymbol{Z}_t \boldsymbol{\Gamma}_t \boldsymbol{V}_t + \boldsymbol{Z}_t \boldsymbol{n}_t.$$

Appendix C. Algorithms

The following pseudo-code listings show the update rules of all discussed temporal-difference algorithms. These updates are executed for each transition from s_t to s_{t+1} performing action a_t and getting the reward r_t.

Algorithm 1 TD(λ) Learning

$$\begin{aligned} \boldsymbol{z}_{t+1} &= \rho_t(\boldsymbol{\phi}_t + \lambda\gamma \boldsymbol{z}_t) \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha_t \delta_t \boldsymbol{z}_{t+1} \end{aligned}$$

Algorithm 2 GTD

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha_t \rho_t (\boldsymbol{\phi}_t - \gamma \boldsymbol{\phi}_{t+1}) \boldsymbol{\phi}_t^T \boldsymbol{w}_t \\ \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t + \beta_t \rho_t (\delta_t \boldsymbol{\phi}_t - \boldsymbol{w}_t) \end{aligned}$$

Algorithm 3 GTD2

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha_t \rho_t (\boldsymbol{\phi}_t - \gamma \boldsymbol{\phi}_{t+1}) \boldsymbol{\phi}_t^T \boldsymbol{w}_t \\ \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t + \beta_t (\rho_t \delta_t - \boldsymbol{\phi}_t^T \boldsymbol{w}_t) \boldsymbol{\phi}_t \end{aligned}$$

Algorithm 4 TDC(λ)

$$\begin{aligned} \boldsymbol{z}_{t+1} &= \rho_t(\boldsymbol{\phi}_t + \lambda\gamma \boldsymbol{z}_t) \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha_t (\delta_t \boldsymbol{z}_t - \gamma(1-\lambda)(\boldsymbol{z}_t^T \boldsymbol{w}_t) \boldsymbol{\phi}_{t+1}) \\ \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t + \beta_t (\rho_t \delta_t \boldsymbol{z}_t - \boldsymbol{\phi}_t^T \boldsymbol{w}_t \boldsymbol{\phi}_t) \end{aligned}$$

Algorithm 5 recursive LSTD(λ) (Init: $\boldsymbol{M}_0 = \epsilon \boldsymbol{I}$)

$$
\begin{aligned}
\Delta\boldsymbol{\phi}_{t+1} =& \boldsymbol{\phi}_t - \rho_t \gamma \boldsymbol{\phi}_{t+1} \\
\boldsymbol{z}_t =& \gamma\lambda\rho_{t-1}\boldsymbol{z}_{t-1} + \boldsymbol{\phi}_t \\
\boldsymbol{K}_{t+1} =& \frac{\boldsymbol{M}_t \boldsymbol{z}_t}{1 + \Delta\boldsymbol{\phi}_{t+1}^T \boldsymbol{M}_t \boldsymbol{z}_t} \\
\boldsymbol{\theta}_{t+1} =& \boldsymbol{\theta}_t + \boldsymbol{K}_{t+1}(\rho_t r_t - \Delta\boldsymbol{\phi}_{t+1})^T \boldsymbol{\theta}_t) \\
\boldsymbol{M}_{t+1} =& \boldsymbol{M}_t - \boldsymbol{K}_{t+1}(\boldsymbol{M}_t^T \Delta\boldsymbol{\phi}_{t+1})^T
\end{aligned}
$$

Algorithm 6 recursive LSTD-TO(λ) (Init: $\boldsymbol{M}_0 = \epsilon \boldsymbol{I}$)

$$
\begin{aligned}
\Delta\boldsymbol{\phi}_{t+1} =& \boldsymbol{\phi}_t - \gamma \boldsymbol{\phi}_{t+1} \\
\boldsymbol{z}_t =& \gamma\lambda\rho_{t-1}\boldsymbol{z}_{t-1} + \boldsymbol{\phi}_t \\
\boldsymbol{K}_{t+1} =& \rho_t \frac{\boldsymbol{M}_t \boldsymbol{z}_t}{1 + \rho_t \Delta\boldsymbol{\phi}_{t+1}^T \boldsymbol{M}_t \boldsymbol{z}_t} \\
\boldsymbol{\theta}_{t+1} =& \boldsymbol{\theta}_t + \boldsymbol{K}_{t+1}(r_t - \Delta\boldsymbol{\phi}_{t+1}^T \boldsymbol{\theta}_t) \\
\boldsymbol{M}_{t+1} =& \boldsymbol{M}_t - \boldsymbol{K}_{t+1}(\boldsymbol{M}_t^T \Delta\boldsymbol{\phi}_{t+1})^T
\end{aligned}
$$

Algorithm 7 recursive LSPE(λ) (Init: $\boldsymbol{N}_0 = \epsilon \boldsymbol{I}, \boldsymbol{A}_0 = \boldsymbol{0}, \boldsymbol{b}_0 = \boldsymbol{0}$)

$$
\begin{aligned}
\boldsymbol{z}_t =& \gamma\lambda\rho_{t-1}\boldsymbol{z}_{t-1} + \boldsymbol{\phi}_t \\
\boldsymbol{N}_{t+1} =& \boldsymbol{N}_t - \frac{\boldsymbol{N}_t \boldsymbol{\phi}_t \boldsymbol{\phi}_t^T \boldsymbol{N}_t}{1 + (\boldsymbol{\phi}_t^T \boldsymbol{N}_t \boldsymbol{\phi}_t)} \\
\boldsymbol{A}_{t+1} =& \boldsymbol{A}_t + \boldsymbol{z}_t(\boldsymbol{\phi}_t - \gamma\rho_t \boldsymbol{\phi}_{t+1}))^T \\
\boldsymbol{b}_{t+1} =& \boldsymbol{b}_t + \rho_t r_t \boldsymbol{z}_t \\
\boldsymbol{\theta}_{t+1} =& \boldsymbol{\theta}_t + \alpha_t \boldsymbol{N}_t(\boldsymbol{b}_t - \boldsymbol{A}_t \boldsymbol{\theta}_t)
\end{aligned}
$$

Algorithm 8 recursive LSPE-TO(λ) (Init: $\boldsymbol{N}_0 = \epsilon \boldsymbol{I}, \boldsymbol{A}_0 = \boldsymbol{0}, \boldsymbol{b}_0 = \boldsymbol{0}$)

$$
\begin{aligned}
\boldsymbol{z}_t =& \gamma\lambda\rho_{t-1}\boldsymbol{z}_{t-1} + \boldsymbol{\phi}_t \\
\boldsymbol{N}_{t+1} =& \boldsymbol{N}_t - \frac{\boldsymbol{N}_t \boldsymbol{\phi}_t \boldsymbol{\phi}_t^T \boldsymbol{N}_t}{1 + (\boldsymbol{\phi}_t^T \boldsymbol{N}_t \boldsymbol{\phi}_t)} \\
\boldsymbol{A}_{t+1} =& \boldsymbol{A}_t + \rho_t \boldsymbol{z}_t(\boldsymbol{\phi}_t - \gamma \boldsymbol{\phi}_{t+1}))^T \\
\boldsymbol{b}_{t+1} =& \boldsymbol{b}_t + \rho_t r_t \boldsymbol{z}_t \\
\boldsymbol{\theta}_{t+1} =& \boldsymbol{\theta}_t + \alpha_t \boldsymbol{N}_t(\boldsymbol{b}_t - \boldsymbol{A}_t \boldsymbol{\theta}_t)
\end{aligned}
$$

Algorithm 9 FPKF(λ) (Init: $\boldsymbol{N}_0 = \boldsymbol{0}$, $\boldsymbol{Z}_0 = \boldsymbol{0}, \boldsymbol{z}_0 = \boldsymbol{0}$)

$$
\begin{aligned}
\boldsymbol{z}_t =& \gamma\lambda\rho_{t-1}\boldsymbol{z}_{t-1} + \boldsymbol{\phi}_t \\
\boldsymbol{Z}_t =& \gamma\lambda\rho_{t-1}\boldsymbol{Z}_{t-1} + \boldsymbol{\phi}_t \boldsymbol{\theta}_t^T \\
\boldsymbol{N}_{t+1} =& \boldsymbol{N}_t - \frac{\boldsymbol{N}_t \boldsymbol{\phi}_t \boldsymbol{\phi}_t^T \boldsymbol{N}_t}{1 + (\boldsymbol{\phi}_t^T \boldsymbol{N}_t \boldsymbol{\phi}_t)} \\
\boldsymbol{\theta}_{t+1} =& \boldsymbol{\theta}_t + \alpha_t \boldsymbol{N}_t(\boldsymbol{z}_t \rho_t r_t - \boldsymbol{Z}_t(\boldsymbol{\phi}_t - \gamma\rho_t \boldsymbol{\phi}_{t+1}))
\end{aligned}
$$

Algorithm 10 recursive BRM with double samples (Init: $\boldsymbol{M}_0 = \epsilon \boldsymbol{I}$, $\boldsymbol{b}_0 = \boldsymbol{0}$)

$$
\begin{aligned}
\Delta\boldsymbol{\phi}'_{t+1} =& \boldsymbol{\phi}_t - \gamma \boldsymbol{\phi}'_{t+1} \\
\Delta\boldsymbol{\phi}''_{t+1} =& \boldsymbol{\phi}_t - \gamma \boldsymbol{\phi}''_{t+1} \\
\boldsymbol{b}_{t+1} =& \boldsymbol{b}_t + \frac{1}{2}\rho'_t \rho''_t (\Delta\boldsymbol{\phi}''_{t+1} r'_t + \Delta\boldsymbol{\phi}'_{t+1} r''_t) \\
\boldsymbol{M}_{t+1} =& \boldsymbol{M}_t - \frac{\rho'_t \rho''_t \boldsymbol{M}_t \Delta\boldsymbol{\phi}'_{t+1} \Delta\boldsymbol{\phi}''^T_{t+1} \boldsymbol{M}_t}{1 + \rho'_t \rho''_t \Delta\boldsymbol{\phi}'^T_{t+1} \boldsymbol{M}_t \Delta\boldsymbol{\phi}''_{t+1}} \\
\boldsymbol{\theta}_{t+1} =& \boldsymbol{M}_{t+1} \boldsymbol{b}_{t+1}
\end{aligned}
$$

Algorithm 11 recursive BRM(λ)
(Init: $\boldsymbol{M}_0 = \epsilon \boldsymbol{I}, \boldsymbol{x}_0 = 0, y_0 = 1, z_0 = 0$)

Compute auxiliary values:

$$\begin{aligned}
\Delta\boldsymbol{\phi}_{t+1} =& \boldsymbol{\phi}_t - \gamma\rho_t\boldsymbol{\phi}_{t+1} \\
p_{t+1} =& \frac{\gamma\lambda\rho_{t-1}}{\sqrt{y_t}} \\
\boldsymbol{U}_{t+1} =& \left[\sqrt{y_t}\Delta\boldsymbol{\phi}_{t+1} + p_{t+1}\boldsymbol{x}_t, \quad p_{t+1}\boldsymbol{x}_t\right] \\
\boldsymbol{V}_{t+1} =& \left[\sqrt{y_t}\Delta\boldsymbol{\phi}_{t+1} + p_{t+1}\boldsymbol{x}_t, \ -p_{t+1}\boldsymbol{x}_t\right]^T \\
\boldsymbol{W}_{t+1} =& \left[\sqrt{y_t}\rho_t r_t + p_{t+1}z_t, \qquad -p_{t+1}z_t\right]^T \\
\boldsymbol{B}_{t+1} =& \boldsymbol{M}_t\boldsymbol{U}_{t+1}[\boldsymbol{I} + \boldsymbol{V}_{t+1}\boldsymbol{M}_t\boldsymbol{U}_{t+1}]^{-1}
\end{aligned}$$

Update traces:

$$\begin{aligned}
\boldsymbol{M}_{t+1} =& \boldsymbol{M}_t - \boldsymbol{B}_{t+1}\boldsymbol{V}_{t+1}\boldsymbol{M}_t \\
y_{t+1} =& (\gamma\lambda\rho_t)^2 y_t + 1 \\
\boldsymbol{x}_{t+1} =& (\gamma\lambda\rho_{t-1})\boldsymbol{x}_t + y_t\Delta\phi_{t+1} \\
z_{t+1} =& (\gamma\lambda\rho_{t-1})z_t + r_t\rho_t y_t
\end{aligned}$$

Update estimate:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{B}_{t+1}(\boldsymbol{W}_{t+1} - \boldsymbol{V}_{t+1}\boldsymbol{\theta}_t)$$

Algorithm 12 parametric GPTD (Init: $\boldsymbol{P}_0 = \boldsymbol{I}$, $\boldsymbol{p}_o = \boldsymbol{0}, d_0 = 0$, $s_0^{-1} = 0$)

$$\begin{aligned}
\Delta\boldsymbol{\phi}_{t+1} =& \boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1} \\
\boldsymbol{p}_{t+1} =& \boldsymbol{p}_t\frac{\gamma\sigma_t^2}{s_t} + \boldsymbol{P}_t\Delta\boldsymbol{\phi}_{t+1} \\
d_{t+1} =& d_t\frac{\gamma\sigma_t^2}{s_t} + r_t - \Delta\boldsymbol{\phi}_{t+1}^T\boldsymbol{\theta}_t \\
s_{t+1} =& \sigma_t^2 + \gamma^2\sigma_{t+1}^2 - \frac{\gamma^2\sigma_t^4}{s_t} \\
& + \left[\boldsymbol{p}_{t+1} + \frac{\gamma\sigma_t^2}{s_t}\boldsymbol{p}_t\right]^T\Delta\boldsymbol{\phi}_{t+1} \\
\boldsymbol{\theta}_{t+1} =& \boldsymbol{\theta}_t + \frac{1}{s_{t+1}}\boldsymbol{p}_{t+1}d_{t+1} \\
\boldsymbol{P}_{t+1} =& \boldsymbol{P}_t - \frac{1}{s_{t+1}}\boldsymbol{p}_{t+1}\boldsymbol{p}_{t+1}^T
\end{aligned}$$

Algorithm 13 Residual-gradient algorithm without double-samples

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t\rho_t\delta_t(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1})$$

References

J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems Measurement and Control*, 97(September): 220–227, 1975.

S.-i. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2): 251–276, Feb. 1998.

A. Antos, C. Szepesvári, and R. Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

F. Bach and E. Moulines. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems 24*, 2011.

L. Baird. Residual algorithms : Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

P. Balakrishna, R. Ganesan, and L. Sherry. Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of Tampa Bay departures. *Transportation Research Part C: Emerging Technologies*, 18(6):950–962, 2010.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, Massachusetts, 1996. ISBN 1-886529-10-8.

D. P. Bertsekas and H. Yu. Projected equation methods for approximate solution of large linear systems. *Journal of Computational and Applied Mathematics*, 227(1):27–50, 2009.

J. A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2):233–246, 2002.

S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

E. Candes and T. Tao. The Dantzig selector: statistical estimation when p is much larger than n. *The Annals of Statistics*, 35(6):2313–2351, 2005.

D. Choi and B. Roy. A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning. *Discrete Event Dynamic Systems*, 16(2):207–239, 2006.

R. W. Cottle, J.-S. Pang, and R. E. Stone. *The Linear Complementarity Problem*. Computer Science and Scientific Computing. Academic Press, 1992. ISBN 0121923509.

R. H. Crites and A. G. Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2-3):235–262, 1998.

W. Dabney and A. G. Barto. Adaptive step-size for online temporal difference learning. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 2012.

P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, (1):19–67, 2010.

M. P. Deisenroth. *Efficient Reinforcement Learning using Gaussian Processes*. PhD thesis, Karlsruhe Institute of Technology, 2010.

M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

Y. Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, Hebrew University, 2005.

Y. Engel, S. Mannor, and R. Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

A.-m. Farahmand and C. Szepesvári. Model selection in reinforcement learning. *Machine Learning*, 85(3):299–332, 2011.

A.-m. Farahmand, M. Ghavamzadeh, C. Szepesvári, and S. Mannor. Regularized policy iteration. In *Advances in Neural Information Processing Systems 21*, 2008.

J. Frank, S. Mannor, and D. Precup. Reinforcement learning in the presence of rare events. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

M. Geist and O. Pietquin. Kalman temporal differences. *Journal of Artificial Intelligence Research*, 39(1):483–532, 2010.

M. Geist and B. Scherrer. l1-penalized projected Bellman residual. In *Proceedings of the Nineth European Workshop on Reinforcement Learning*, 2011.

M. Geist and B. Scherrer. Off-policy learning with eligibility traces : A survey. Technical report, INRIA Lorraine - LORIA, 2013.

M. Geist, B. Scherrer, A. Lazaric, and M. Ghavamzadeh. A Dantzig selector approach to temporal difference learning. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

S. Gelly and D. Silver. Achieving master level play in 9 x 9 computer go. In *Proceedings of the 23th AAAI Conference on Artificial Intelligence*, 2008.

A. Geramifard, M. Bowling, and R. S. Sutton. Incremental least-squares temporal difference learning. *Proceedings of the 21th AAAI Conference on Artificial Intelligence*, 2006a.

A. Geramifard, M. Bowling, M. Zinkevich, and R. S. Sutton. iLSTD: Eligibility traces and convergence analysis. In *Advances in Neural Information Processing Systems 19*, 2006b.

A. Geramifard, F. Doshi, J. Redding, N. Roy, and J. P. How. Online discovery of feature dependencies. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

A. Geramifard, T. J. Walsh, and J. P. How. Batch-iFDD for representation expansion in large MDPs. In *Conference on Uncertainty in Artificial Intelligence*, 2013.

M. Ghavamzadeh, A. Lazaric, O.-A. Maillard, and R. Munos. LSTD with random projections. In *Advances in Neural Information Processing Systems 23*, 2010.

M. Ghavamzadeh, A. Lazaric, R. Munos, and M. Hoffman. Finite-sample analysis of Lasso-TD. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

P. W. Glynn and D. L. Iglehart. Importance sampling for stochastic simulations. *Management Science*, 35(11):1367–1392, 1989.

H. Hachiya and M. Sugiyama. Feature selection for reinforcement learning: Evaluating implicit state-reward dependency via conditional mutual information. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2010.

M. Hoffman, A. Lazaric, M. Ghavamzadeh, and R. Munos. Regularized least squares temporal difference learning with nested l2 and l1 penalization. In *Proceedings of the Nineth European Workshop on Reinforcement Learning*, 2011.

M. Hutter and S. Legg. Temporal difference updating without a learning rate. In *Advances in Neural Information Processing Systems 20*, 2007.

R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

J. Johns and S. Mahadevan. Sparse approximate policy evaluation using graph-based basis functions. Technical report, University of Massachusetts Amherst, 2009.

J. Johns, C. Painter-Wakefield, and R. Parr. Linear complementarity for regularized policy evaluation and improvement. In *Advances in Neural Information Processing Systems 23*, 2010.

T. Jung and D. Polani. Least squares SVM for least squares TD learning. In *European Conference on Artificial Intelligence*, 2006.

P. W. Keller, S. Mannor, and D. Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

J. Z. Kolter and A. Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

R. M. Kretchmar and C. W. Anderson. Comparison of CMACs and radial basis functions for kocal function approximators in reinforcement learning. In *International Conference on Neural Networks*, 1997.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4(Dec):1107–1149, 2003.

A. Lazaric, M. Ghavamzadeh, and R. Munos. Finite-sample analysis of LSTD. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

L. Li. A worst-case comparison between temporal difference and residual gradient with linear function approximation. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

B. Liu, S. Mahadevan, and J. Liu. Regularized off-policy TD-learning. In *Advances in Neural Information Processing Systems 25*, 2012.

M. Loth, M. Davy, and P. Preux. Sparse temporal difference learning using LASSO. In *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning*, 2007.

H. R. Maei. *Gradient Temporal-Difference Learning Algorithms*. PhD thesis, University of Alberta, 2011.

S. Mahadevan and M. Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, 8(Oct):2169–2231, 2007.

A. R. Mahmood, R. S. Sutton, T. Degris, and P. M. Pilarski. Tuning-free step-size adaptation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012.

I. Menache, S. Mannor, and N. Shimkin. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134(1):215–238, 2005.

D. Meyer, H. Shen, and K. Diepold. l1-regularized gradient temporal-difference learning. In *Proceedings of the Tenth European Workshop on Reinforcement Learning*, 2012.

A. Nedic and D. P. Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, 13(1-2):79–110, 2003.

C. Painter-Wakefield and R. Parr. Greedy algorithms for sparse reinforcement learning. In *Proceedings of the 29th International Conference on Machine Learning*, 2012a.

C. Painter-Wakefield and R. Parr. L1 regularized linear temporal difference learning. Technical report, Duke University, Durham, NC, 2012b.

R. Parr, C. Painter-Wakefield, L. Li, and M. Littman. Analyzing feature generation for value-function approximation. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27th Asilomar Conference on Signals, Systems and Computers*, 1993.

M. Petrik, G. Taylor, R. Parr, and S. Zilberstein. Feature selection using regularization in approximate linear programs for Markov decision processes. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

B. A. Pires. *Statistical Analysis of L1-penalized Linear Estimation with Applications*. Master thesis, University of Alberta, 2011.

C. E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems 16*, 2003.

M. Riedmiller and T. Gabel. On experiences in a complex and competitive gaming domain: Reinforcement learning meets robocup. In *IEEE Symposium on Computational Intelligence and Games*, 2007.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

M. Rosenblatt. *Markov Processes. Structure and Asymptotic Behavior*. Springer, 1971. ISBN 978-3642652400.

N. L. Roux and A. Fitzgibbon. A fast natural Newton method. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

B. Scherrer. Should one compute the temporal difference fix point or minimize the Bellman residual? the unified oblique projection view. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

B. Scherrer and M. Geist. Recursive least-squares learning with eligibility traces. In *Proceedings of the Nineth European Workshop on Reinforcement Learning*, 2011.

R. Schoknecht. Optimality of reinforcement learning algorithms with linear function approximation. In *Advances in Neural Information Processing Systems 15*, 2002.

P. J. Schweitzer and A. Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110(2):568–582, 1985.

D. Silver, R. Sutton, and M. Müller. Reinforcement learning of local shape in the game of go. In *International Joint Conference on Artificial Intelligence*, 2007.

S. Sra, S. Nowozin, and S. J. Wright. *Optimization for Machine Learning*. MIT Press, 2012. ISBN 9780262016469.

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Adaptive computation and machine learning. MIT Press, 1998. ISBN 9780262193986.

R. S. Sutton, D. Precup, and S. Singh. Intra-option learning about temporally abstract actions. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.

R. S. Sutton, C. Szepesvári, and H. R. Maei. A convergent O(n) algorithm for off-policy temporal-difference learning with linear function approximation. In *Advances in Neural Information Processing Systems 21*, 2008.

R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

G. Taylor and R. Parr. Kernelized value function approximation for reinforcement learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

G. Tesauro. TD-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

J. N. Tsitsiklis and B. van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions On Automatic Control*, 42(5):674–690, 1997.

R. J. Williams and L. C. Baird. Tight performance bounds on greedy policies based on imperfect value functions. In *Yale Workshop on Adaptive and Learning Systems*, 1993.

X. Xu, T. Xie, D. Hu, and X. Lu. Kernel least-squares temporal difference learning. *International Journal of Information Technology*, 11(9):54–63, 2005.

H. Yu. Convergence of least squares temporal difference methods under general conditions. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, 37(6A):3468–3497, 2009.

Journal of Machine Learning Research 15 (2014) 885-920 Submitted 9/12; Revised 6/13; Published 3/14

Active Learning Using Smooth Relative Regret Approximations with Applications

Nir Ailon NAILON@CS.TECHNION.AC.IL
Ron Begleiter RONBEG@CS.TECHNION.AC.IL
Department of Computer Science
Taub Building
Technion Israel Institute of Technology
Haifa 32000, Israel

Esther Ezra ESTHER@CIMS.NYU.EDU
Coutrant Institute of Mathematical Science
New York University
251 Mercer Street
New York, NY, 10012 USA

Editor: Yoav Freund

Abstract

The disagreement coefficient of Hanneke has become a central data independent invariant in proving active learning rates. It has been shown in various ways that a concept class with low complexity together with a bound on the disagreement coefficient at an optimal solution allows active learning rates that are superior to passive learning ones.

We present a different tool for pool based active learning which follows from the existence of a certain uniform version of low disagreement coefficient, but is not equivalent to it. In fact, we present two fundamental active learning problems of significant interest for which our approach allows nontrivial active learning bounds. However, any general purpose method relying on the disagreement coefficient bounds only, fails to guarantee any useful bounds for these problems. The applications of interest are: Learning to rank from pairwise preferences, and clustering with side information (a.k.a. semi-supervised clustering).

The tool we use is based on the learner's ability to compute an estimator of the difference between the loss of any hypothesis and some fixed "pivotal" hypothesis to within an absolute error of at most ε times the disagreement measure (ℓ_1 distance) between the two hypotheses. We prove that such an estimator implies the existence of a learning algorithm which, at each iteration, reduces its in-class excess risk to within a constant factor. Each iteration replaces the current pivotal hypothesis with the minimizer of the estimated loss difference function with respect to the previous pivotal hypothesis. The label complexity essentially becomes that of computing this estimator.

Keywords: active learning, learning to rank from pairwise preferences, semi-supervised clustering, clustering with side information, disagreement coefficient, smooth relative regret approximation

1. Introduction

An *active learner* selects the instances from which it learns, contrary to standard *PAC learning*. In the streaming setting, active learners may reject labels for instances arriving

in a stream, and in the pool setting they may collect a pool of instances and then choose a subset from which labels are requested. Although a relatively young field compared to traditional (passive) learning, there is by now a significant body of literature on the subject (see, e.g., Freund et al., 1997; Dasgupta, 2005; Castro et al., 2005; Kääriäinen, 2006; Balcan et al., 2006; Sugiyama, 2006; Hanneke, 2007; Balcan et al., 2007; Dasgupta et al., 2007; Bach, 2007; Castro and Nowak, 2008; Balcan et al., 2008; Dasgupta and Hsu, 2008; Cavallanti et al., 2008; Hanneke, 2009; Beygelzimer et al., 2009, 2010; Koltchinskii, 2010; Cesa-Bianchi et al., 2010; Yang et al., 2010; Hanneke and Yang, 2010; El-Yaniv and Wiener, 2010; Hanneke, 2011; Orabona and Cesa-Bianchi, 2011; Cavallanti et al., 2011; Yang et al., 2011; Wang, 2011; Minsker, 2012). Refer to the survey by Settles (2009) for a further discussion about active learning.

The disagreement coefficient of Hanneke (2007) has become a central data independent invariant in proving active learning rates. It has been shown in various ways that a concept class with low complexity together with a bound on the disagreement coefficient at an optimal solution allow active learning rates that are superior to passive rates under certain low noise conditions (see, e.g., Hanneke, 2007; Balcan et al., 2007; Dasgupta et al., 2007; Castro and Nowak, 2008; Beygelzimer et al., 2010). The best results assuming only bounded VC dimension d and disagreement coefficient θ can roughly be stated as follows: If the sought (in-class) excess risk μ has the same order of magnitude as the optimal error ν or larger, then the number of required queries is roughly $\widetilde{O}(\theta d \log(1/\mu))$.[1] Otherwise, the number is roughly $\widetilde{O}(\theta d \nu^2/\mu^2)$. Note that this results makes no assumption on the noise (except maybe for its magnitude). Better results can be achieved by assuming certain statistical properties of the noise (especially the model of Mammen and Tsybakov, 1999; Tsybakov, 2004).

The idea behind the disagreement coefficient is intuitive and simple: If a hypothesis h is r-close to the optimum, then the *difference between their losses* (the regret of h) can be computed from instances in the *disagreement region* only, defined as the set of instances on which the r-ball around the optimum is not unanimous on. This means that for minimizing regret, one may restrict attention to hypotheses laying in iteratively shrinking *version spaces* and to instances in the corresponding disagreement regions, which are shrinking in tandem with the version spaces if the disagreement coefficient is small. As pointed out in Beygelzimer et al. (2010), ignoring hypotheses outside the version space is brittle business, because an error in computation of the version space results in a failure of the algorithm. They propose a scheme in which no version space is computed. Instead, a certain importance weighted scheme is used. We also use importance weighting, but in the pool based setting and not in the streaming setting as they do.[2]

Analyzing the difference between losses of hypotheses ("relative regrets") is used vastly in numerous theoretical work on active learning, but not attached directly. In this work we argue that a careful construction of empirical processes uniformly estimating the relative regret of all hypotheses with respect to a fixed "pivotal" hypothesis yields fast active learning rates. We call such constructions "SRRA" (Smooth Relative Regret Approximations).

1. The $\widetilde{O}$ notation suppresses polylogarithmic terms.
2. Note that a practitioner can pretend that any pool based input is a stream, though that approach would probably not take full advantage of the data.

We also show that low disagreement coefficient and VC dimension assumptions imply such efficient constructions, and give rise to yet another proof for the usefulness of the disagreement coefficient in active learning. Nevertheless, our SRRA-based iterative method does *not* need to compute or restrict itself to shrinking version spaces. This is supported by presenting two fundamental pool based learning problems for which *direct* SRRA constructions yield superior active learning rates, whereas approaches that exploit the disagreement coefficient only (in the sense presented in Section 3), requires the practitioner to obtain labels for the entire pool (!) even for moderately chosen parameters. We conclude that the SRRA method is, up to minor factors, at least as good as the disagreement coefficient method, but can be significantly better in certain cases.

We note that another important line of design and analysis of active learning algorithms makes certain structural or Bayesian assumptions on the noise (e.g., Balcan et al., 2007; Castro and Nowak, 2008; Hanneke, 2009; Koltchinskii, 2010; Yang et al., 2010; Wang, 2011; Yang et al., 2011; Minsker, 2012). We expect that one can get yet improved analysis in our framework under these assumptions. We leave this to future work.

The rest of the paper is laid out as follows: In Section 2 we present notations and basic definitions, including an introduction to our method. In Section 3 we show that low disagreement coefficients imply efficient SRRAs. Then, we present our two main applications of SRRA that lie beyond the scope of disagreement coefficients: (i) learning to rank from pairwise preferences (LRPP) (Section 4), and (ii) clustering with side information in (Section 5). In Section 6 we present additional results and practical considerations, and, in particular, the application of our method with convex relaxation in case the corresponding ERM problems are too difficult (computationally) to optimally solve. We conclude in Section 7 and suggest future directions.

2. Definitions, Notations, and Core Results

We follow the notation of Hanneke (2011): Let $\mathcal{X}$ be an instance space, and let $\mathcal{Y} = \{0, 1\}$ be a label space. Denote by $\mathcal{D}$ the distribution over $\mathcal{X} \times \mathcal{Y}$, with corresponding marginals $\mathcal{D}_{\mathcal{X}}$ and $\mathcal{D}_{\mathcal{Y}}$. In this work we assume for convenience (only) that each label Y is a deterministic function of X, so that if $X \sim \mathcal{D}_{\mathcal{X}}$ then $(X, Y(X))$ is distributed according to $\mathcal{D}$.

By $\mathcal{C}$ we denote a concept class of functions mapping $\mathcal{X}$ to $\mathcal{Y}$. The error rate of a hypothesis $h \in \mathcal{C}$ equals

$$\mathrm{er}_{\mathcal{D}}(h) = E_{(X,Y)\sim\mathcal{D}}[h(X) \neq Y(X)] \ .$$

The noise rate ν of $\mathcal{C}$ is defined as $\nu = \inf_{h\in\mathcal{C}} \mathrm{er}_{\mathcal{D}}(h)$. We will focus on the scenario in which ν is attained at an optimal hypothesis h^*, so that $\mathrm{er}_{\mathcal{D}}(h^*) = \nu$. Define the *distance* $\mathrm{dist}(h_1, h_2)$ between two hypotheses $h_1, h_2 \in \mathcal{C}$ as $\Pr_{X\sim\mathcal{D}_{\mathcal{X}}}[h_1(X) \neq h_2(X)]$; observe that $\mathrm{dist}(\cdot, \cdot)$ is a pseudo-metric over pairs of hypotheses. For a hypothesis $h \in \mathcal{C}$ and a number $r \geq 0$, the ball $\mathcal{B}(h, r)$ around h of radius r is defined as $\{h' \in \mathcal{C} : \ \mathrm{dist}(h, h') \leq r\}$. For a set $V \subseteq \mathcal{C}$ of hypotheses, let $\mathrm{DIS}(V)$ denote

$$\mathrm{DIS}(V) = \{x \in \mathcal{X} : \exists h_1, h_2 \in V \text{ such that } h_1(x) \neq h_2(x)\} \ .$$

2.1 The Disagreement Coefficient

The disagreement coefficient of h with respect to $\mathcal{C}$ under $\mathcal{D}_{\mathcal{X}}$ is defined as

$$\theta_h = \sup_{r>0} \frac{\Pr_{\mathcal{D}_{\mathcal{X}}}\left[\mathrm{DIS}\left(\mathcal{B}(h,r)\right)\right]}{r} \ , \tag{1}$$

where $\Pr_{\mathcal{D}_{\mathcal{X}}}[\mathcal{W}]$ for $\mathcal{W} \subseteq \mathcal{X}$ denotes the probability measure with respect to the distribution $\mathcal{D}_{\mathcal{X}}$. Define the uniform disagreement coefficient θ as $\sup_{h\in\mathcal{C}} \theta_h$, namely

$$\theta = \sup_{h\in\mathcal{C}} \sup_{r>0} \frac{\Pr_{\mathcal{D}_{\mathcal{X}}}\left[\mathrm{DIS}\left(\mathcal{B}(h,r)\right)\right]}{r} \ . \tag{2}$$

Remark 1 *A useful slight variation of the definitions of θ_h and θ can be obtained by replacing $\sup_{r>0}$ with $\sup_{r\geq\nu}$ in (1) and (2). We will explicitly say when we refer to this variation in what follows.*

2.2 Smooth Relative Regret Approximations (SRRA)

Fix $h \in \mathcal{C}$ (which we call the *pivotal hypothesis*). Denote by $\mathrm{reg}_h : \mathcal{C} \mapsto \mathbb{R}$ the function defined as

$$\mathrm{reg}_h(h') = \mathrm{er}_{\mathcal{D}}(h') - \mathrm{er}_{\mathcal{D}}(h) \ .$$

We call reg_h the *relative regret function with respect to h*. Note that for $h = h^*$ this is simply the usual regret, or (in-class) excess risk function.

Definition 2 *Let $f : \mathcal{C} \mapsto \mathbb{R}$ be any function, and $0 < \varepsilon < 1/5$ and $0 < \mu \leq 1$. We say that f is an* (ε,μ)-smooth relative regret approximation ((ε,μ)-SRRA) *with respect to h if for all $h' \in \mathcal{C}$,*

$$|f(h') - \mathrm{reg}_h(h')| \leq \varepsilon \cdot \left(\mathrm{dist}(h,h') + \mu\right) \ .$$

If $\mu = 0$ we simply call f an ε-smooth relative regret approximation *with respect to h.*

Although the definition is general, we focus here on the pool based active learning setting. Intuitively, think of f as an empirical version of reg_h. The definition guides us to query labels such that we cover the whole spectrum of disagreement-instances w.r.t. the pivot h, while assuring corresponding estimation accuracies with granularity proportional to inverse distances from h. Intuitively, the condition supports holistic explore-exploit query strategies: We cover the whole range of error "types" (exploration), and at the same time put more querying efforts "near" the intermediate solution (exploitation). The following theorem and corollary constitute the main ingredient in our work. They show that a sequence of (ε,μ)-SRRA estimators define a competitive hypothesis.

Theorem 3 *Let $h \in \mathcal{C}$ and f be an (ε,μ)-SRRA with respect to h. Let $h_1 = \mathrm{argmin}_{h'\in\mathcal{C}} f(h')$. Then*

$$\mathrm{er}_{\mathcal{D}}(h_1) = (1 + O(\varepsilon))\,\nu + O\left(\varepsilon \cdot \mathrm{er}_{\mathcal{D}}(h)\right) + O(\varepsilon\mu) \ .$$

Proof Applying the definition of (ε, μ)-SRRA we have:

$$\begin{aligned}
\mathrm{er}_{\mathcal{D}}(h_1) \leq & \mathrm{er}_{\mathcal{D}}(h) + f(h_1) + \varepsilon \cdot \mathrm{dist}(h, h_1) + \varepsilon\mu \\
\leq & \mathrm{er}_{\mathcal{D}}(h) + f(h^*) + \varepsilon \cdot \mathrm{dist}(h, h_1) + \varepsilon\mu \\
\leq & \mathrm{er}_{\mathcal{D}}(h) + \nu - \mathrm{er}_{\mathcal{D}}(h) + \varepsilon \cdot \mathrm{dist}(h, h^*) + \varepsilon \cdot \mathrm{dist}(h, h_1) + 2\varepsilon\mu \\
\leq & \nu + \varepsilon\Big(2\mathrm{dist}(h, h^*) + \mathrm{dist}(h_1, h^*)\Big) + 2\varepsilon\mu \,. \qquad (3)
\end{aligned}$$

The first inequality is from the definition of (ε, μ)-SRRA, the second is from the fact that h_1 minimizes $f(\cdot)$ by construction, the third is again from the definition of (ε, μ)-SRRA, and the definitions of h^* and reg_h, the fourth is by the triangle inequality. The proof is completed by plugging $\mathrm{dist}(h, h^*) \leq \mathrm{er}_{\mathcal{D}}(h) + \nu$, and $\mathrm{dist}(h_1, h^*) \leq \mathrm{er}_{\mathcal{D}}(h_1) + \nu$ into Equation 3, subtracting $\varepsilon \cdot \mathrm{er}_{\mathcal{D}}(h_1)$ from both sides, and dividing by $(1 - \varepsilon)$. ■

A simple inductive use of Theorem 3 proves the following corollary, bounding the excess risk of an ERM based active learning algorithm (see Algorithm 1 for corresponding pseudocode). The algorithm's query-complexity depends on the specific constructions of (ε, μ)-SRRA estimators. Note that this algorithm never restricts itself to a shrinking version space.

Corollary 4 *Let $h_0, h_1, h_2, \ldots$ be a sequence of hypotheses in $\mathcal{C}$ such that for all $i \geq 1$, $h_i = \mathrm{argmin}_{h' \in \mathcal{C}} f_{i-1}(h')$, where f_{i-1} is an (ε, μ)-SRRA with respect to h_{i-1}. Then for all $i \geq 0$,*

$$\mathrm{er}_{\mathcal{D}}(h_i) = (1 + O(\varepsilon))\,\nu + O(\varepsilon^i)\mathrm{er}_{\mathcal{D}}(h_0) + O(\varepsilon\mu) \,.$$

Proof Applying Theorem 3 with h_i and h_{i-1}, we have

$$\mathrm{er}_{\mathcal{D}}(h_i) = (1 + O(\varepsilon))\,\nu + O\left(\varepsilon \cdot \mathrm{er}_{\mathcal{D}}(h_{i-1})\right) + O(\varepsilon\mu) \,.$$

Solving this recursion, one gets

$$\mathrm{er}_{\mathcal{D}}(h_i) = \sum_{j=1}^{i} \varepsilon^{j-1}\,(1 + O(\varepsilon))\,\nu + O(\varepsilon^i) \cdot \mathrm{er}_{\mathcal{D}}(h_0) + O\left(\sum_{j=1}^{i} \varepsilon^j\right)\mu \,.$$

The result follows easily by bounding geometric sums. ■

Algorithm 1 An Active Learning Algorithm from SRRA's

Input: an initial solution $h_0 \in \mathcal{C}$, estimation parameters $\epsilon \in (0, 1/5), \mu > 0$, and number of iterations T

1: **for** $i = 0, 1, \ldots, T-1$ **do**
2: $\quad h_{i+1} \leftarrow \mathrm{argmin}_{h' \in \mathcal{C}} f(h')$, where f is an (ε, μ)-smooth relative regret approximation with respect to h_i
3: **end for**
4: **return** h_T

We will show below problems of interest in which (ε, μ)-SRRA's with respect to a given hypothesis h can be obtained using labels at few randomly (and adaptively) selected points

X from the pool $\mathcal{X}$, if the uniform disagreement coefficient θ is small. This will constitute another proof for the usefulness of the disagreement coefficient in design and analysis of active learning algorithms. We then present two problems for which a direct construction of an SRRA yields a significantly better query complexity than that guaranteed using the disagreement coefficient alone.

3. Constant Uniform Disagreement Coefficient Implies Efficient SRRAs

We show that a bounded uniform disagreement coefficient implies existence of query efficient (ε, μ)-SRRAs. This constitutes yet another proof of the usefulness of the disagreement coefficient in design of active learning algorithms, via Algorithm 1. Plugging the resulting query efficient (ε, μ)-SRRAs into our iterative SRRA method (Algorithm 1) provides an active learning algorithm with query complexity that matches the state-of-the-art, yet does not improve it. In the following sections, however, we design two other (ε, μ)-SRRAs constructions that yield active learning algorithms which beats the corresponding state-of-the-art query complexity guarantees.

3.1 The Construction

Returning to our problem, assume the uniform disagreement coefficient θ corresponding to $\mathcal{C}$ is finite and $\nu > 0$. Fix $h \in \mathcal{C}$ and let $L = \lceil \log \mu^{-1} \rceil$. Define $\mathcal{X}_0 = \mathrm{DIS}(\mathcal{B}(h, \mu))$ and for $i = 1, \ldots, L$ define $\mathcal{X}_i$ to be

$$\mathcal{X}_i = \mathrm{DIS}(\mathcal{B}(h, \mu 2^i)) \setminus \mathrm{DIS}(\mathcal{B}(h, \mu 2^{i-1})) \ .$$

Let $\eta_i = \Pr_{\mathcal{D}_\mathcal{X}}[\mathcal{X}_i]$ be the measure of $\mathcal{X}_i$, and δ a failure probability hyper-parameter. For each $i \geq 0$ draw a sample $X_{i,1}, \ldots, X_{i,m}$ of

$$m \ = \ O\left(\varepsilon^{-2}\theta\left(d \log \theta + \log\left(\delta^{-1} \log(1/\mu)\right)\right)\right)$$

examples in $\mathcal{X}_i$, each of which drawn independently from the distribution $\mathcal{D}_\mathcal{X}|\mathcal{X}_i$ (with repetitions). (By $\mathcal{D}_\mathcal{X}|\mathcal{X}_i$ we mean, the distribution $\mathcal{D}_\mathcal{X}$ conditioned on $\mathcal{X}_i$.) We will now define an estimator function $f : \mathcal{C} \mapsto \mathbb{R}$ of reg_h, as follows. For any $h' \in \mathcal{C}$ and $i = 0, 1, \ldots, L$ let

$$f_i(h') = \eta_i m^{-1} \sum_{j=1}^{m} \left(\mathbf{1}_{Y(X_{i,j}) \neq h'(X_{i,j})} - \mathbf{1}_{Y(X_{i,j}) \neq h(X_{i,j})}\right) \ .$$

Our estimator is now defined as $f(h') = \sum_{i=0}^{L} f_i(h')$. The estimator $f(h')$ is an unbiased empirical counterpart of the relative regret in which each empirical error $\left(\mathbf{1}_{Y(X_{i,j}) \neq h'(X_{i,j})} - \mathbf{1}_{Y(X_{i,j}) \neq h(X_{i,j})}\right)$ is weighted with respect to η_i. The weights are depicted as different element sizes. Observe that the weight is inverse proportional to the probability of drawing $X_{i,j}$.

We next show that f is an (ε, μ)-SRRA with respect to h. This allows us to incorporate f into Algorithm 1 and gain a $(1+\varepsilon)\nu$ competitive hypothesis. Thus, the query complexity will boil down to the size of the sample defining the empirical relative-regret minimizer f.

Theorem 5 *Let f, h, h', m be as above. With probability at least $1-\delta$, f is an (ε, μ)-SRRA with respect to h.*

Proof A main tool to be exploited in the proof is called *relative ε-approximations* due to Haussler (1992) and Li et al. (2000). It is defined as follows. Let $h \in \mathcal{X} \mapsto \mathbb{R}^+$ be some function, and let $\mu_h = E_{X\sim\mathcal{D}_\mathcal{X}}[h(X)]$. Let $X_1, \ldots, X_m$ denote i.i.d. draws from $\mathcal{D}_\mathcal{X}$, and let $\hat{\mu}_h = \frac{1}{m} \cdot \sum_{i=1}^m h(X_i)$ denote the empirical average. Let $\kappa > 0$ be an adjustable parameter. We are going to use the following measure of distance between the true expectation μ_h and its estimator $\hat{\mu}_h$: $d_\kappa(\mu_h, \hat{\mu}_h) = \frac{|\mu_h - \hat{\mu}_h|}{\mu_h + \hat{\mu}_h + \kappa}$.

This measure corresponds to a relative error when approximating μ_h by $\hat{\mu}_h$. Indeed, let $\varepsilon > 0$ be our approximation ratio, and put $d_\kappa(\mu_h, \hat{\mu}_h) < \varepsilon$. This easily yields

$$|\mu_h - \hat{\mu}_h| < \frac{2\varepsilon}{1-\varepsilon} \cdot \mu_h + \frac{\varepsilon}{1-\varepsilon} \cdot \kappa\,. \tag{4}$$

In other words, this implies that $|\mu_h - \hat{\mu}_h| < O(\varepsilon)\big(\mu_h + \kappa\big)$.

Let us fix a parameter $0 < \delta < 1$. Assume $\mathcal{C}$ is a set of $\{0,1\}$ valued functions on $\mathcal{X}$ of VC dimension d. Li et al. (2000) show that if one samples

$$m = O\left(\varepsilon^{-2}\kappa^{-1}(d\log\kappa^{-1} + \log\delta^{-1})\right)$$

examples as above (with a sufficiently large constant of proportionality), then (4) holds uniformly for all $h \in \mathcal{C}$ with probability at least $1 - \delta$.

We consider the range space $(\mathcal{X}, \mathcal{C}^*)$, defined by

$$\mathcal{C}^* = \left(\bigcup_{h'\in\mathcal{C}} \big\{\{X \in \mathcal{X} : h'(X) = 0\}\big\}\right) \cup \left(\bigcup_{h'\in\mathcal{C}} \big\{\{X \in \mathcal{X} : h'(X) = 1\}\big\}\right)\,.$$

In words, $\mathcal{C}^*$ is the collection of all subsets $S \subseteq \mathcal{X}$, whose elements $X \in S$ are mapped to the same value (0 or 1) by h', for some $h' \in \mathcal{C}$. Assume $(\mathcal{X}, \mathcal{C}^*)$ has VC dimension[3] d, and fix $h \in \mathcal{C}$. Let $L = \lceil \log \mu^{-1} \rceil$, and recall the definition of the disagreement sets $\mathcal{X}_i$.

We now apply this definition of *relative ε-approximations*, and the corresponding results within our context. For any h', we define the following four sets of instances:

$$\begin{aligned}
R^{++}_{h'} &= \{X \in \mathcal{X} :\ h'(X) = Y(X) = 1, \text{ and } h(X) = 0\}\,,\\
R^{+-}_{h'} &= \{X \in \mathcal{X} :\ h'(X) = 1, \text{ and } h(X) = Y(X) = 0\}\,,\\
R^{-+}_{h'} &= \{X \in \mathcal{X} :\ h'(X) = 0, \text{ and } h(X) = Y(X) = 1\}\,,\\
R^{--}_{h'} &= \{X \in \mathcal{X} :\ h'(X) = Y(X) = 0, \text{ and } h(X) = 1\}\,.
\end{aligned}$$

Observe that the set $\{X \in \mathcal{X} : h(X) \neq h'(X)\}$ equals the disjoint union of $R^{++}_{h'}$, $R^{+-}_{h'}$, $R^{-+}_{h'}$ and $R^{--}_{h'}$. For each $i = 0, \ldots, L$ and $b \in \{++, +-, -+, --\}$ let $R^b_{h',i} = R^b_{h'} \cap \mathcal{X}_i$. Let $\mathcal{R}_i^{\,b} = \{R^b_{h',i} : h' \in \mathcal{C}\}$. It is easy to verify that the VC dimension of the range spaces $\big(\mathcal{X}_i, \mathcal{R}_i^{\,b}\big)$ is at most d. Each set in $\mathcal{R}_i^{\,b}$ is an intersection of a set in $\mathcal{C}^*$ with some fixed set.

For any $R \subseteq \mathcal{X}_i$ let $\rho_i(R) = \Pr_{X\sim\mathcal{D}_\mathcal{X}|\mathcal{X}_i}[X \in R]$, and $\hat{\rho}_i(R) = m^{-1}\sum_{j=1}^m \mathbf{1}_{X_{i,j}\in R}$. Note that $\hat{\rho}_i(R)$ is an unbiased estimator of $\rho_i(R)$.

3. The VC dimension of $(\mathcal{X}, \mathcal{C}^*)$ is the maximum cardinality of a subset $A \subseteq \mathcal{X}$ for which $\{A \cap r : r \in \mathcal{C}^*\}$ contains all subsets of A.

By the choice of m, inequality (4), and the assumptions on θ and ν we have that with probability at least $1-\delta/L$, for all $R \subseteq \mathcal{R}_i^{++} \cup \mathcal{R}_i^{+-} \cup \mathcal{R}_i^{-+} \cup \mathcal{R}_i^{--}$,

$$|\rho_i(R) - \hat{\rho}_i(R)| = O(\varepsilon) \cdot \left(\rho_i(R) + \theta^{-1}\right), \tag{5}$$

and by the probability union bound we obtain that this uniformly holds for all $i = 0, \ldots, L$ with probability at least $1-\delta$.

Now fix $h' \in \mathcal{C}$ and let $r = \mathrm{dist}(h, h')$. Let $i_r = \lceil \log(r/\mu) \rceil$. By the definition of $\mathcal{X}_i$, $h(X) = h'(X)$ for all $X \in \mathcal{X}_i$ whenever $i > i_r$. We can therefore decompose $\mathrm{reg}_h(h')$ as:

$$\begin{aligned}
\mathrm{reg}_h(h') &= \mathrm{er}_{\mathcal{D}}(h') - \mathrm{er}_{\mathcal{D}}(h) \\
&= \sum_{i=0}^{L} \eta_i \cdot \left(\Pr_{X \sim \mathcal{D}_{\mathcal{X}} | \mathcal{X}_i}[Y(X) \neq h'(X)] - \Pr_{X \sim \mathcal{D}_{\mathcal{X}} | \mathcal{X}_i}[Y(X) \neq h(X)] \right) \\
&= \sum_{i=0}^{i_r} \eta_i \cdot \left(\Pr_{X \sim \mathcal{D}_{\mathcal{X}} | \mathcal{X}_i}[Y(X) \neq h'(X)] - \Pr_{X \sim \mathcal{D}_{\mathcal{X}} | \mathcal{X}_i}[Y(X) \neq h(X)] \right) \\
&= \sum_{i=0}^{i_r} \eta_i \cdot \left(-\rho_i(R_{h'}^{++}) + \rho_i(R_{h'}^{+-}) + \rho_i(R_{h'}^{-+}) - \rho_i(R_{h'}^{--}) \right).
\end{aligned}$$

On the other hand, we similarly have that

$$f(h') = \sum_{i=0}^{i_r} \eta_i \cdot \left(-\hat{\rho}_i(R_{h'}^{++}) + \hat{\rho}_i(R_{h'}^{+-}) + \hat{\rho}_i(R_{h'}^{-+}) - \hat{\rho}_i(R_{h'}^{--}) \right).$$

Combining, we conclude using (5) that

$$|\mathrm{reg}_h(h') - f(h')| \leq O\left(\varepsilon \sum_{i=0}^{i_r} \eta_i \cdot \left(\rho_i(R_{h'}^{++}) + \rho_i(R_{h'}^{+-}) + \rho_i(R_{h'}^{-+}) + \rho_i(R_{h'}^{--}) + 4\theta^{-1} \right) \right). \tag{6}$$

But now notice that $\sum_{i=0}^{i_r} \eta_i \cdot \left(\rho_i(R_{h'}^{++}) + \rho_i(R_{h'}^{+-}) + \rho_i(R_{h'}^{-+}) + \rho_i(R_{h'}^{--}) \right)$ equals r, since it corresponds to those elements $X \in \mathcal{X}$ on which h, h' disagree. Also note that $\sum_{i=0}^{i_r} \eta_i$ is at most $2 \max\left\{ \Pr_{\mathcal{D}_{\mathcal{X}}}\left[\mathrm{DIS}\left(\mathcal{B}(h, r)\right)\right], \Pr_{\mathcal{D}_{\mathcal{X}}}\left[\mathrm{DIS}\left(\mathcal{B}(h, \mu)\right)\right] \right\}$. By the definition of θ, this implies that the RHS of (6) is bounded by $\varepsilon(r + \mu)$, as required by the definition of (ε, μ)-SRRA.[4] ■

Corollary 6 *An (ε, μ)-SRRA with respect to h can be constructed, with probability at least $1-\delta$, using at most*

$$m\left(1 + \lceil \log(1/\mu) \rceil\right) = O\left(\theta \varepsilon^{-2} \left(\log(1/\mu)\right) \left(d \log \theta + \log(\delta^{-1} \log(1/\mu)) \right) \right)$$

label queries.

4. The O-notation disappeared because we assume that the constants are properly chosen in the definition of the sample size m.

Combining Corollaries 4 and 6 (Algorithm 1), we obtain an active learning algorithm in the ERM setting, with query complexity depending on the uniform disagreement coefficient and the VC dimension. Assume δ is a constant. If we are interested in excess risk of order at least that of the optimal error ν, then we may take ε to be, say, $1/5$ and achieve the sought bound by constructing $(1/5, \nu)$-SRRA's using $O(\theta d(\log(1/\nu))(\log\theta))$, once for each of $O(\log(1/\nu))$ iterations of Algorithm 1. If we seek a solution with error $(1+\varepsilon)\nu$, we would need to construct (ε, ν)-SRRA's using $O(\theta d\varepsilon^{-2}(\log(1/\nu))(\log\theta))$ query labels, one for each of $O(\log(1/\nu))$ iterations of the algorithm. The total label query complexity is $O(\theta d(\log^2(1/\nu))(\log\theta))$, which is $O(\log(1/\nu))$ times the best known bounds using disagreement coefficient and VC dimension bounds only (e.g., Dasgupta et al., 2007; Beygelzimer et al., 2009).

Remark 7 (Using the uniform disagreement coefficient) *We first note that in known arguments bounding query complexity using the disagreement coefficient, the disagreement coefficient θ_{h^*} with respect to the optimal hypothesis h^* is used in the analysis, and not the uniform coefficient θ. Also note that in both previously known results bounding query complexity using disagreement coefficient and VC dimension bounds as well as our result, the slight improvement described in Remark 1 applies. In other words, all arguments remain valid if we replace the supremums in (1) and (2) with* $\sup_{r\geq\nu}$.

Remark 8 (Computing the $\mathcal{X}_i$'s) *Note that we show how to compute $\mathcal{X}_i$* exactly *in polynomial time when dealing with linear hypotheses spaces. This is shown in Section 6.1.2 in the context of a ranking problem. Yet, it indicates that in certain cases $\mathcal{X}_i$ can be computed efficiently. Additionally note that the sets $\mathcal{X}_i$ are defined w.r.t. a pivot (i.e., given) hypothesis; thus, it is possible to estimate it, for example, using ideas similar to the nice works of Balcan et al. (2006) and Dasgupta et al. (2007).*

4. Application #1: Learning to Rank from Pairwise Preferences (LRPP)

"Learning to Rank" takes various forms in theory and practice of learning, as well as in combinatorial optimization. In such problems, the goal is to order a set V based on constraints.

A large body of learning literature considers the following scenario: For each $v \in V$ there is a label on some discrete ordinal scale, and the goal is to learn how to order V so as to respect induced pairwise preferences. For example, a scale of $\{1, 2, 3, 4, 5\}$, as in hotel/restaurant star quality; where, if u has a label of 5 ("very good") and v has a label of 1 ("very bad"), then any ordering that places v ahead of u is penalized. Note that even if the labels are noisy, the induced pairwise preferences here are always transitive, hence no combinatorial problem arises. Our work does not deal with this setting.

When the basic unit of information consists of preferences over pairs $u, v \in V$, then the problem becomes combinatorially interesting. In case all quadratically many pairwise preferences are given for free, the corresponding optimization problem is known as *Minimum Feedback Arc-Set in Tournaments* (MFAST).[5] MFAST has been shown to be NP-hard (Alon, 2006). Recently, Kenyon-Mathieu and Schudy (2007) showed a PTAS for this (passive

5. A maximization version of this problem exists as well.

learning) problem. Several important recent works address the challenge of approximating the minimum feedback arc-set problem (Ailon et al., 2008; Braverman and Mossel, 2008; Coppersmith et al., 2010).

Here we consider a query efficient variant of the problem, in which each preference comes at a cost, and the goal is to produce a competitive solution while reducing the preference-query overhead. Other very recent work consider similar settings (Jamieson and Nowak, 2011; Ailon, 2012). Jamieson and Nowak (2011) consider a common scenario in which the alternatives can be characterized in terms of d real-valued features and the ranking obeys the structure of the Euclidean distances between such embeddings. They present an active learning algorithm that requires, using *average case analysis*, as few as $O(d \log n)$ labels in the noiseless case, and $O(d \log^2 n)$ labels under a certain *parametric* noise model. Our work uses worst-case analysis, and assumes an adversarial noise model. In this Section we analyze the pure combinatorial problem (not assuming any feature embeddings). In Section 6 we tackle the problem with linearly induced permutation over feature space embeddings.

Ailon (2012) consider the same setting as ours. Our main result Corollary 13 is a slight improvement over the main result of Ailon (2012) in query complexity, but it provides another significant improvement. Ailon (2012) uses a querying method that is based on a divide and conquer strategy. The weakness of such a strategy can be explained by considering an example in which we want to search a restricted set of permutations (e.g., the setting of Section 6.1). When dividing and conquering, the algorithm in Ailon (2012) is doomed to search a Cartesian product of two permutations spaces (left and right). There is no guarantee that there even exists a permutation in the restricted space that respects this division. In our querying algorithm this limitation is lifted.

4.1 Problem Definition

Let V be a set of n elements (alternatives). The instance space $\mathcal{X}$ is taken to be the set of all distinct pairs of elements in V, namely $V \times V \setminus \big\{(u,u) : u \in V\big\}$. The distribution $\mathcal{D}_{\mathcal{X}}$ is uniform on $\mathcal{X}$. The label function $Y : \mathcal{X} \mapsto \{0,1\}$ encodes a preference function satisfying $Y\big((u,v)\big) = 1 - Y\big((v,u)\big)$ for all $u, v \in V$.[6] By convention, we think of $Y\big((u,v)\big) = 1$ as a stipulation that u is preferred over v. For convenience we will drop the double-parentheses in what follows.

The class of solution functions $\mathcal{C}$ we consider is all $h : \mathcal{X} \to \{0,1\}$ such that it is *skew-symmetric*: $h(u,v) = 1 - h(v,u)$, and *transitive*: $h(u,z) \leq h(u,v) + h(v,z)$ for all distinct $u, v, z \in V$. This is equivalent to the space of permutations over V, and we will use the notation $\pi, \sigma, \ldots$ instead of $h, h', \ldots$ in the remainder of the section. We also use notation $u \prec_\pi v$ as a predicate equivalent to $\pi(u,v) = 1$. Endowing $\mathcal{X}$ with the uniform measure, $\mathrm{dist}(\pi,\sigma)$ turns out to be (up to normalization) the well known kendall-τ distance: $\mathrm{dist}(\pi,\sigma) = N^{-1} \sum_{u \neq v} \mathbf{1}_{\pi(u,v) \neq \sigma(u,v)}$, where $N = n(n-1)$ is the number of all ordered pairs.

4.2 The Weakness of Using Disagreement Coefficient Arguments

We first demonstrate the weakness of a disagreement coefficient approach with respect to this problem. It has been shown in Ailon (2012) that the uniform disagreement coefficient

6. Note that we could have defined $\mathcal{X}$ to be unordered pairs of elements in V without making any assumption on Y. We chose this definition for convenience in what follows.

of $\mathcal{C}$ is $\Omega(n)$. To see this simple fact, notice that if we start from some permutation π and swap the positions of *any* two elements $u, v \in V$, then we obtain a permutation of distance at most $O(1/n)$ away from π, hence the disagreement region of the ball of radius $O(1/n)$ around π is the entire space $\mathcal{X}$. It is also known that the VC dimension of $\mathcal{C}$ is $n-1$ (e.g., Radinsky and Ailon, 2011). This is simply because there is always a labeling $Y(\cdot)$ over any set of $n = |V|$ *pairs* that defines preference cycles. Thus, the set of permutations $\mathcal{C}$ cannot shatter n pairs. Using Corollary 6, we conclude that we would need $\Omega(n^2)$ preference labels to obtain an (ε, μ)-SRRA for any meaningful pair (ε, μ). This is uninformative because the cardinality of $\mathcal{X}$ is $O(n^2)$. A similar bound is obtained using any known active learning bound using disagreement coefficient and VC-dimension bounds only.

Remark 9 *A slight improvement can be obtained using the refined definition of disagreement coefficients of Remark 1. Observe that the uniform disagreement coefficient, as well as the disagreement coefficient at the optimal solution h^* becomes $\theta = \theta_{h^*} = O(1/\nu)$, if $\nu \geq \frac{1}{n}$.*[7] *This improves the query complexity bound to $O(n\nu^{-1})$. If ν tends to n^{-1} from above, in the limit this becomes a quadratic (in n) query complexity.*

Remark 10 *A natural question in this context is why the optimal hypothesis cannot be approximated by sampling preference-pairs uniformly at random. In other words, is it sufficient for our setting to apply a passive learning method? Do we really need to apply the more sophisticated active-learning machinery? Ailon (2012, Section 2) shows that applying plain Empirical Risk Minimization approach is doomed to query the entire pool. Moreover, he shows that even when the noise is zero the uniform sampling approach will w.h.p. have to query all $O(n^2)$ possible labels.*

We next show how to construct more useful (in terms of query complexity) SRRA's for LRPP, *for arbitrarily small ν*.

4.3 Better SRRA for LRPP

Consider the following idea for creating an ε-SRRA for LRPP,[8] with respect to some fixed $\pi \in \mathcal{C}$. We start by defining the following sample size parameter:

$$p = O\left(\varepsilon^{-3} \log^3 n\right) \ . \tag{7}$$

For all $u \in V$ and for all $i = 0, 1, \ldots, \lceil \log n \rceil$, let $I_{u,i}$ Denote the set of elements v such that $(2^i - 1)p < |\pi(u) - \pi(v)| < 2^{i+1}p$ where, abusing notation, $\pi(u)$ is the position of u in π. For example, $\pi(u)$ is 1 if u beats all other elements, and n if it is beaten by all others. From this set, choose a random sequence $R_{u,i} = (v_{u,i,1}, v_{u,i,2}, \ldots, v_{u,i,p})$ of p elements, each chosen uniformly and independently from $I_{u,i}$.[9]

7. Due to symmetry, the uniform disagreement coefficient here equals θ_h for any $h \in \mathcal{C}$.
8. Note that we can neglect the parameter μ because taking its value equal $1/n$ tantamount to zero.
9. A variant of this sampling scheme is as follows: For each pair (u, v), add it to the query-set with probability proportional to $\min\{1, p/|\pi(u) - \pi(v)|\}$. A similar scheme can be found in Ailon et al. (2007), Halevy and Kushilevitz (2007) and Ailon (2012) but the strong properties proven here were not known.

For distinct $u, v \in V$ and a permutation $\sigma \in \mathcal{C}$, let $\mathrm{cost}_{u,v}(\sigma)$ denote the contribution of the pair u, v to $\mathrm{er}_{\mathcal{D}}(\sigma)$, namely: $\mathrm{cost}_{u,v}(\sigma) = N^{-1}\mathbf{1}_{\sigma(u,v)\neq Y(u,v)}$. Let $\mathrm{reg}_{u,v|\sigma}$ denote the contribution of $\{u, v\} \in \mathcal{X}$ to $\mathrm{reg}_{\pi}(\sigma)$, that is

$$\mathrm{reg}_{u,v|\sigma} = 2\left(\mathrm{cost}_{u,v}(\sigma) - \mathrm{cost}_{u,v}(\pi)\right) . \tag{8}$$

Note the notation discards the dependency on π because it is assumed to be fixed. The use of factor 2 is because $\mathrm{cost}_{u,v} \equiv \mathrm{cost}_{v,u}$.

Our estimator $f(\sigma)$ of $\mathrm{reg}_{\pi}(\sigma) = \mathrm{er}_{\mathcal{D}}(\sigma) - \mathrm{er}_{\mathcal{D}}(\pi)$ is defined as

$$f(\sigma) = \frac{1}{2}\sum_{u\in V}\sum_{i=0}^{\lceil \log n\rceil}\frac{|I_{u,i}|}{p}\sum_{t=1}^{p}\mathrm{reg}_{u,v_{u,i,t}|\sigma} .$$

Clearly, $f(\sigma)$ is an unbiased estimator of $\mathrm{reg}_{\pi}(\sigma)$ for any σ. Our goal is to prove that $f(\sigma)$ is an ε-SRRA.

Theorem 11 *With probability at least $1 - n^{-3}$, the function f is an ε-SRRA with respect to π.*

Proof The main idea is to *decompose* the difference $|f(\sigma) - \mathrm{reg}_{\pi}(\sigma)|$ vis-a-vis corresponding pieces of $\mathrm{dist}(\sigma, \pi)$. The first half of the proof is devoted to definition of such distance "pieces." Then using counting and standard deviation-bound arguments we show that the decomposition is, with high probability, an ε-SRRA.

We start with a few definitions. Recall that for any $\pi \in \mathcal{C}$ and $u \in V$, $\pi(u)$ denotes the position of u in the unique permutation that π defines. For example, $\pi(u) = 1$ if u beats all other alternatives: $\pi(u, v) = 1$ for all $v \neq u$; Similarly $\pi(u) = n$ if u is beaten by all other alternatives. For any permutation $\sigma \in \mathcal{C}$, we define the corresponding *profile* of σ as the vector:[10]

$$\mathrm{prof}(\sigma) = \big(\sigma(u_1) - \pi(u_1), \sigma(u_2) - \pi(u_2), \ldots, \sigma(u_n) - \pi(u_n)\big).$$

Note that $\|\mathrm{prof}(\sigma)\|_1$ is $d_{\mathrm{SF}}(\sigma, \pi)$, the Spearman footrule distance between σ and π. For a subset V' of V, we let $\mathrm{prof}(\sigma)[V']$ denote the restriction of the vector $\mathrm{prof}(\sigma)$ to V'. Namely, the vector obtained by zeroing in $\mathrm{prof}(\sigma)$ all coordinates $v \notin V'$.

Now fix $\sigma \in \mathcal{C}$ and two distinct $u, v \in V$. Assume u, v is an inversion in σ with respect to π, and that $|\pi(u) - \pi(v)| = b$ for some integer b. Then either $|\pi(u) - \sigma(u)| \geq b/2$ or $|\pi(v) - \sigma(v)| \geq b/2$. We will "charge" the inversion to $\mathrm{argmax}_{z\in\{u,v\}}\left\{|\pi(z) - \sigma(z)|\right\}$.[11] For any $u \in V$, let $\mathrm{charge}_{\sigma}(u)$ denote the set of elements $v \in V$ such that (u, v) is an inversion in σ with respect to π, which is charged to u based on the above rule. The function $\mathrm{reg}_{\pi}(\sigma)$ can now be written as

$$\mathrm{reg}_{\pi}(\sigma) = \sum_{u\in V}\sum_{v\in\mathrm{charge}_{\sigma}(u)}\mathrm{reg}_{u,v|\sigma} ,$$

10. For the sake of definition assume an arbitrary indexing such that $V = \{u_i : i = 1, \ldots, n\}$.

11. Breaking ties using some canonical rule, for example, charge to the greater of u, v viewed as integers.

where $\mathrm{reg}_{u,v|\sigma}$ is defined in Equation 8. Indeed, any pair that is not inverted contributes nothing to the difference. Similarly, our estimator $f(\sigma)$ can be written as

$$f(\sigma) = \sum_{u \in U} \sum_{i=0}^{\lceil \log n \rceil} \frac{|I_{u,i}|}{p} \sum_{t=1}^{p} \mathrm{reg}_{u,v_{u,i,t}|\sigma} \cdot \mathbf{1}_{v_{u,i,t} \in \mathrm{charge}_\sigma(u)} \ .$$

Observe that we dropped the factor $1/2$ above because we count each pair $\{u, v\}$ only once.

For any even integer M let $U_{\sigma,M}$ denote the set of all elements $u \in V$ such that

$$M/2 < |\pi(u) - \sigma(u)| \leq M \ .$$

Let $U_{\sigma,\leq M}$ denote:

$$\bigcup_{M' \leq M} U_{\sigma,M'} \ .$$

From now on, we shall remove the subscript π, because it is held fixed. Consider the following restrictions of $\mathrm{reg}(\sigma)$ and $f(\sigma)$:

$$\mathrm{reg}(\sigma, M) = \sum_{u \in U_{\sigma,M}} \sum_{v \in \mathrm{charge}_\sigma(u)} \mathrm{reg}_{u,v|\sigma} \ , \tag{9}$$

$$f(\sigma, M) = \sum_{u \in U_{\sigma,M}} \sum_{i=0}^{\lceil \log n \rceil} \sum_{t=1}^{p} \frac{|I_{u,i}|}{p} \left(\mathrm{reg}_{u,v_{u,i,t}|\sigma} \cdot \mathbf{1}_{v_{u,i,t} \in \mathrm{charge}_\sigma(u)} \right) \ . \tag{10}$$

Clearly, $f(\sigma, M)$ is an unbiased estimator of $\mathrm{reg}(\sigma, M)$. Let $T_{\sigma,M}$ denote the set of all elements $u \in V$ such that $|\pi(u) - \sigma(u)| \leq \varepsilon M$. We further split the expressions in (9)-(10) as follows:

$$\mathrm{reg}(\sigma, M) = A(\sigma, M) + B(\sigma, M), \text{ and } f(\sigma, M) = \hat{A}(\sigma, M) + \hat{B}(\sigma, M),$$

where,

$$A(\sigma, M) = \sum_{u \in U_{\sigma,M}} \sum_{v \in \mathrm{charge}_\sigma(u) \cap \overline{T_{\sigma,M}}} \mathrm{reg}_{u,v|\sigma} \ ,$$

$$\hat{A}(\sigma, M) = \sum_{u \in U_{\sigma,M}} \sum_{i=0}^{\lceil \log n \rceil} \frac{|I_{u,i}|}{p} \sum_{t=1}^{p} \mathrm{reg}_{u,v_{u,i,t}|\sigma} \cdot \mathbf{1}_{v_{u,i,t} \in \mathrm{charge}_\sigma(u) \cap \overline{T_{\sigma,M}}} \ ,$$

$\overline{(\cdot)}$ is set complement in V, and $B(\sigma, M), \hat{B}(\sigma, M)$ are analogous with $T_{\sigma,M}$ instead of $\overline{T_{\sigma,M}}$, as follows:

$$B(\sigma, M) = \sum_{u \in U_{\sigma,M}} \sum_{v \in \mathrm{charge}_\sigma(u) \cap T_{\sigma,M}} \mathrm{reg}_{u,v|\sigma} \ ,$$

$$\hat{B}(\sigma, M) = \sum_{u \in U_{\sigma,M}} \sum_{i=0}^{\lceil \log n \rceil} \frac{|I_{u,i}|}{p} \sum_{t=1}^{p} \mathrm{reg}_{u,v_{u,i,t}|\sigma} \cdot \mathbf{1}_{v_{u,i,t} \in \mathrm{charge}_\sigma(u) \cap T_{\sigma,M}} \ .$$

We now estimate the deviation of $\hat{A}(\sigma, M)$ from $A(\sigma, M)$. Fix M. Notice that the expression $A(\sigma, M)$ is completely determined by non-zero elements of the vector $\mathrm{prof}(\sigma)[U_{\sigma,\leq M} \cap \overline{T_{\sigma,M}}]$. Let $J_{\sigma,M}$ denote the number of nonzeros in $\mathrm{prof}(\sigma)[U_{\sigma,M}]$. Each nonzero coordinate of $\mathrm{prof}(\sigma)[U_{\sigma,\leq M} \cap \overline{T_{\sigma,M}}]$ is bounded below by εM in absolute value by definition. Let $P(d, M)$ denote the number of possibilities for the vector $\mathrm{prof}(\sigma)[\overline{T_{\sigma,M}}]$ for σ running over all permutations satisfying $d_{\mathrm{SF}}(\sigma, \pi) = d$. We claim that

$$P(d, M) \leq n^{2d/(\varepsilon M)} \ .$$

Indeed, there can be at most $d/(\varepsilon M)$ nonzeros in $\mathrm{prof}(\sigma)[\overline{T_{\sigma,M}}]$, and each nonzero coordinate can trivially take at most n values. The bound follows.

Now fix integers d and J, and consider the subspace of permutations σ such that $J_{\sigma,M} = J$ and $d_{\mathrm{SF}}(\sigma, \pi) = d$. Define for each $u \in U_{\sigma,M}$, $i \in \lceil \log n \rceil$ and $t = 1, \ldots, p$ a random variable $X_{u,i,t}$ as follows

$$X_{u,i,t} = \frac{|I_{u,i}|}{p} \, \mathrm{reg}_{u, v_{u,i,t}|\sigma} \cdot \mathbf{1}_{v_{u,i,t} \in \mathrm{charge}_\sigma(u) \cap \overline{T_{\sigma,M}}} \ .$$

Clearly $\hat{A}(\sigma, M) = \sum_{u \in U_{\sigma,M}} X_{u,i,t}$. For any $u \in V$, let $i_u = \mathrm{argmax}_i \{|I_{u,i}| \leq 4M\}$, and observe that, by our charging scheme, $X_{u,i,t} = 0$ almost surely, for all $i > i_u$ and $t = 1 \ldots p$. Also observe that for all u, i, t, $|X_{u,i,t}| \leq 2N^{-1}|I_{u,i}|/p \leq 2^{i+1}/p$ almost surely. For a random variable X, we denote by $\|X\|_\infty$ the infimum over numbers α such that $X \leq \alpha$ almost surely. We conclude:

$$\sum_{u \in U_{\sigma,M}} \sum_{i=0}^{i_u} \sum_{t=1}^{p} \|X_{u,i,t}\|_\infty^2 \leq \sum_{u \in U_{\sigma,M}} \sum_{i=0}^{i_u} N^{-2} p 2^{2i+2}/p^2 \leq c_2 p^{-1} N^{-2} J M^2$$

for some global $c_2 > 0$. (We used a bound on the sum of a geometric series.) Using Hoeffding bound, we conclude that the probability that $\hat{A}(\sigma, M)$ deviates from its expected value of $A(\sigma, M)$ by more than some $s > 0$ is at most $\exp\{-s^2 p/(2c_2 J M^2 N^{-2})\}$. We conclude that the probability that $A(\sigma, M)$ deviates from its expected value by more than $\varepsilon d/(N \log n)$ is at most $\exp\{-c_1 \varepsilon^2 d^2 p/(J M^2 \log^2 n)\}$, for some global $c_1 > 0$. Hence, by taking $p = O(\varepsilon^{-3} d^{-1} M J \log^3 n)$, by union bounding over all $P(d, M)$ possibilities for $\mathrm{prof}(\sigma)[\overline{T_{\sigma,M}}]$, with probability at least $1 - n^{-7}$ simultaneously for all σ satisfying $J_{\sigma,M} = J$ and $d_{\mathrm{SF}}(\sigma, \pi) = d$,

$$|A(\sigma, M) - \hat{A}(\sigma, M)| \leq \varepsilon d/(N \log n) \ . \tag{11}$$

But note that, trivially, $JM \leq d$, hence our choice of p in (7) is satisfactory. Finally, union bound over the $O(n^3 \log n)$ possibilities for the values of J and d and $M = 1, 2, 4, ..$ to conclude that (11) holds for all permutations σ simultaneously, with probability at least $1 - n^{-3}$.

Consider now $\hat{B}(\sigma, M)$ and $B(\sigma, M)$. We will need to further decompose these two expressions as follows. For $u \in U_{\sigma,M}$, we define a disjoint cover $(T^1_{u,\sigma,M}, T^2_{u,\sigma,M})$ of $\mathrm{charge}_\sigma(u) \cap T_{\sigma,M}$ as follows. If $\pi(u) < \sigma(u)$, then

$$T^1_{u,\sigma,M} = \{v \in T_{\sigma,M} : \pi(u) + \varepsilon M < \pi(v) < \sigma(u) - \varepsilon M\} \ .$$

Otherwise,

$$T^1_{u,\sigma,M} = \{v \in T_{\sigma,M} : \sigma(u) + \varepsilon M < \pi(v) < \pi(u) - \varepsilon M\} \ .$$

Note that by definition, $T^1_{u,\sigma,M} \subseteq \mathrm{charge}_\sigma(u)$. The set $T^2_{u,\sigma,M}$ is thus taken to be

$$T^2_{u,\sigma,M} = (\mathrm{charge}_\sigma(u) \cap T_{\sigma,M}) \setminus T^1_{u,\sigma,M} \ .$$

The expressions $B(\sigma, M), \hat{B}(\sigma, M)$ now decompose as $B^1(\sigma, M)+B^2(\sigma, M)$ and $\hat{B}^1(\sigma, M)+\hat{B}^2(\sigma, M)$, respectively, as follows:

$$\begin{aligned}
B^1(\sigma,M) &= \sum_{u\in U_{\sigma,M}} \sum_{v\in T^1_{u,\sigma,M}} \mathrm{reg}_{u,v|\sigma} \ , \\
B^2(\sigma,M) &= \sum_{u\in U_{\sigma,M}} \sum_{v\in T^2_{u,\sigma,M}} \mathrm{reg}_{u,v|\sigma} \ , \\
\hat{B}^1(\sigma,M) &= \sum_{u\in U_{\sigma,M}} \sum_{i=0}^{\lceil \log n\rceil} \frac{|I_{u,i}|}{p} \sum_{t=1}^{p} \mathrm{reg}_{u,v_{u,i,t}|\sigma} \cdot \mathbf{1}_{v_{u,i,t}\in T^1(u,\sigma,M)} \ , \\
\hat{B}^2(\sigma,M) &= \sum_{u\in U_{\sigma,M}} \sum_{i=0}^{\lceil \log n\rceil} \frac{|I_{u,i}|}{p} \sum_{t=1}^{p} \mathrm{reg}_{u,v_{u,i,t}|\sigma} \cdot \mathbf{1}_{v_{u,i,t}\in T^2(u,\sigma,M)} \ .
\end{aligned}$$

Now notice that $B^1(\sigma, M)$ can be uniquely determined from $\mathrm{prof}(\sigma)[\overline{T_{\sigma,M}}]$. Indeed, in order to identify $T^1_{u,\sigma,M}$ for some $u \in U_{\sigma,M}$, it suffices to identify zeros in a subset of coordinates of $\mathrm{prof}(\sigma)[\overline{T_{\sigma,M}}]$, where the subset depends only on $\mathrm{prof}(\sigma)[\{u\}]$. Additionally, the value of $C_{u,v}(\sigma) - C_{u,v}(\pi)$ can be "read" from $\mathrm{prof}(\sigma)[\overline{T_{\sigma,M}}]$ (and, of course, $Y(u, v)$) if $v \in T^1_{u,\sigma,M}$. Hence, a Hoeffding bound and a union bound similar to the one used for bounding $|\hat{A}(\sigma, M) - A(\sigma, M)|$ can be used to bound (with high probability) the difference $|\hat{B}^1(\sigma, M) - B^1(\sigma, M)|$ uniformly for all σ and $M = 1, 2, 4, ...$, as well.

Bounding $|\hat{B}^2(\sigma, M) - B^2(\sigma, M)|$ can be done using the following simple claim.

Claim 12 *For $u \in V$ and an integer q, we say that the sampling is successful at (u, q) if the random variable*

$$\left|\left\{(i,t) : \pi(v_{u,i,t}) \in [\pi(u) + (1-\varepsilon)q,\ \pi(u) + (1+\varepsilon)q] \cup [\pi(u) - (1+\varepsilon)q,\ \pi(u) - (1-\varepsilon)q]\right\}\right|$$

is at most twice its expected value. We say that the sampling is successful if it is successful at all $u \in V$ and $q \le n$. If the sampling is successful, then uniformly for all σ and all $M = 1, 2, 4, ...$,

$$|\hat{B}^2(\sigma, M) - B^2(\sigma, M)| = O(\varepsilon J_{\sigma,M} M/N) \, .$$

The sampling is successful with probability at least $1 - n^{-3}$ if $p = O(\varepsilon^{-1} \log n)$.

The last assertion in the claim follows from Chernoff bounds. Note that our bound (7) on p is satisfactory, in virtue of the claim.

Summing up the errors $|\hat{A}(\sigma, M) - A(\sigma, M)|$, $|\hat{B}(\sigma, M) - B(\sigma, M)|$ over all M gives us the following assertion: With probability at least $1 - n^{-2}$, uniformly for all σ,

$$|f(\sigma) - \mathrm{reg}_\pi(\sigma)| \le \varepsilon N^{-1} d_{\mathrm{SF}}(\pi, \sigma) \le 2\varepsilon \mathrm{dist}(\pi, \sigma) \, ,$$

where the last inequality is by Diaconis and Graham (1977). This concludes the proof. ■

Algorithm 2 summarizes our specific ε-SRRA construction for LRPP. Note that by the choice of the sample size p, the number of preference queries needed for computing f is $O(\varepsilon^{-3} n \log^4 n)$. Observe that given a pivot π, our LRPP-SRRA construction is simple to implement, and is (obviously) computationally efficient.

Algorithm 2 SRRA for LRPP

Input: V, $\mathcal{C}$, a pivot $\pi \in \mathcal{C}$, estimation parameter $\epsilon \in (0, 1/5)$

```
p ← O(ε^{-3} log^3 n)
for u ∈ V do
    for i = 0, 1, ..., ⌈log n⌉ do
        I_{u,i} ← {v : (2^i − 1)p < |π(u) − π(v)| < 2^{i+1} p}
        for t = 1, ..., p do
            v_{u,i,t} ← a uniformly and independently sampled alternative from I_{u,i}
        end for
    end for
end for
return f : C ⟶ ℝ, defined by
```

$$f(\sigma) = \sum_{u \in V} \sum_{i=0}^{\lceil \log n \rceil} \frac{|I_{u,i}|}{p} \sum_{t=1}^{p} \left(\mathrm{cost}_{u,v_{u,i,t}}(\sigma) - \mathrm{cost}_{u,v_{u,i,t}}(\pi) \right) ,$$

We can now combine these LRPP ε-SRRA constructions in Algorithm 1, the SRRA's method meta-algorithm defined in Corollary 4. This provides the following bound on the number of preference queries.

Corollary 13 *There exists an active learning algorithm for obtaining a solution $\pi \in \mathcal{C}$ for LRPP with $\mathrm{er}_{\mathcal{D}}(\pi) \leq (1 + O(\varepsilon))\,\nu$ with total query complexity of $O\left(\varepsilon^{-3} n \log^5 n\right)$. The algorithm succeeds with probability at least $1 - n^{-2}$.*

Corollary 13 tells us that the method of SRRA provides a solution of cost $(1 + \varepsilon)\nu$ with query complexity that is slightly above linear in n (for constant ε), regardless of the magnitude of ν. In comparison, we saw in Section 4.2 that any known active learning results that used bounded disagreement coefficient and VC dimension arguments only guaranteed a query complexity of $O(n\nu^{-1})$, tending to the pool size of $n(n-1)$ as ν becomes small. Note that $\nu = o(1)$ is quite realistic for this problem. For example, consider the following noise model. A ground truth permutation π^* exists, $Y(u, v)$ is obtained as a human response to the question of preference between u and v with respect to π^*, and the human errs with probability proportional to $|\pi^*(u) - \pi^*(v)|^{-\rho}$. Namely, closer pairs of item in the ground truth permutation are more prone to confuse a human labeler. The resulting noise is $\nu = n^{-\rho}$ for some $\rho > 0$. (Note, however, our work does not assume Bayesian noise, and we present this scenario for illustration purposes only.)

In terms of query complexity it turns out that our bound provides only a slight improvement over the divide-and-conquer active learning algorithm for LRPP of Ailon (2012).

Specifically, we improve the dependency on ε from ε^{-6} to ε^{-3}. Although our method provides only a minor improvement it still defines the current state-of-the-art for query efficient LRPP. However, more importantly it defines the first query-efficient LRPP algorithm that is applicable over arbitrary set of permutations, $\mathcal{C} \subseteq V!$. We take advantage of this fact in the Section 6.1, where we instantiate ε-SRRA's for the set of permutations induced by half-spaces in $\mathbb{R}^d$.

5. Application #2: Clustering with Side Information

Clustering with side information is a fairly new variant of clustering first described, independently, by Demiriz et al. (1999), and Ben-Dor et al. (1999). In the machine learning community it is also widely known as *semi-supervised clustering*. There are a few alternatives for the form of feedback providing the side-information. The most natural ones are the single item labels (e.g., Demiriz et al., 1999), and the pairwise constraints (e.g., Ben-Dor et al., 1999).

Here we consider pairwise side information: "must"/"cannot" link for pairs of elements $u, v \in V$. Each such information bit comes at a cost, and must be treated frugally. In a combinatorial optimization theoretical setting known as *correlation clustering* there is no input cost overhead, and similarity information for all (quadratically many) pairs is available. The goal there is to optimally clean the noise (nontransitivity). Correlation clustering was defined in Bansal et al. (2004), and also in Shamir et al. (2004) under the name *cluster editing*. Constant factor approximations are known for various minimization versions of this problems (Charikar and Wirth, 2004; Ailon et al., 2008). A PTAS is known for a minimization version in which the number of clusters is fixed to be k (Giotis and Guruswami, 2006), as in our setting.

In machine learning, there are two main approaches for using pairwise side information. In the first approach, this information is used to fine tune or learn a *distance* function, which is then passed on to any standard clustering algorithm such as k-means or k-medians (see, e.g., Klein et al., 2002; Xing et al., 2002; Cohn et al., 2000; Balcan et al., 2009; Shamir and Tishby, 2011; Voevodski et al., 2012). The second approach, which is more related to our work, modifies the clustering algorithms's objective so as to incorporate the pairwise constraints (see, e.g., Basu, 2005; Eriksson et al., 2011; Cesa-Bianchi et al., 2012). Basu (2005) in his thesis, which also serves as a comprehensive survey, has championed this approach in conjunction with k-means, and hidden Markov random field clustering algorithms. In our work we isolate the use of information coming from pairwise clustering constraints, and separate it from the geometry of the problem. In future work it would be interesting to analyze our framework in conjunction with the geometric structure of the input. Interestingly Eriksson et al. (2011) studies active learning for clustering using the geometric input structure. Unlike our setting, they assume either no noise or Bayesian noise.

5.1 Problem Definition

Let V be a set of points of size n. Our goal now is to partition V into k sets (clusters), where k is fixed. In most applications, V is endowed with some metric, and the practitioner uses this metric in order to evaluate the quality of a clustering solution. In some cases, known

as *semi-supervised clustering*, or *clustering with side information*, additional information comes in the form of *pairwise constraints*. Such a constraint tells us for a pair $u, v \in V$ whether they should be in the same cluster or in separate ones. We concentrate on using such information.

Using the notation of our framework, $\mathcal{X}$ denotes the set of distinct pairs of elements in V (same as in Section 4), and $\mathcal{D}_{\mathcal{X}}$ is the corresponding uniform measure. The label $Y\big((u,v)\big) = 1$ means that u and v should be clustered together, and $Y\big((u,v)\big) = 0$ means the opposite. Assume that $Y\big((u,v)\big) = Y\big((v,u)\big)$ for all u, v.[12]

The concept class $\mathcal{C}$ is the set of equivalence relations over V with at most k equivalence classes. More precisely, Every $h \in \mathcal{C}$ is identified with a disjoint cover $V_1, \ldots, V_k$ of V (some V_i's possible empty), with $h\big((u,v)\big) = 1$ if and only if $u, v \in V_j$ for some j. As usual, Y may induce a non-transitive relation. (For example, we could have $Y\big((u,v)\big) = Y\big((v,z)\big) = 1$ and $Y\big((u,z)\big) = 0$.) In what follows, we will drop the double parentheses. Also, we will abuse notation by viewing h as both an equivalence relation and a disjoint cover $\{V_1, \ldots, V_k\}$ of V. We take $\mathcal{D}$ to be the uniform measure on $\mathcal{X}$. The error of $h \in \mathcal{C}$ is given as $\mathrm{er}_{\mathcal{D}}(h) = N^{-1} \sum_{(u,v)\in\mathcal{X}} \mathbf{1}_{h(u,v)\neq Y(u,v)}$ where, as before, $N = |\mathcal{X}| = n(n-1)$. We will define $\mathrm{cost}_{u,v}(h)$ to be the contribution $N^{-1}\mathbf{1}_{h(u,v)\neq Y(u,v)}$ of $(u,v) \in \mathcal{X}$ to $\mathrm{er}_{\mathcal{D}}$. The distance $\mathrm{dist}(h,h')$ is given as $\mathrm{dist}(h,h') = N^{-1} \sum_{(u,v)\in\mathcal{X}} \mathbf{1}_{h(u,v)\neq h'(u,v)}$.

5.2 The Ineffectiveness of Using Disagreement Coefficient Arguments

We demonstrate once again the weakness of a disagreement coefficient approach. It is easy to verify that the uniform disagreement coefficient of $\mathcal{C}$ is $\Theta(n)$. Indeed, starting from any solution $h \in \mathcal{C}$ with corresponding partitioning $\{V_1, \ldots, V_k\}$, consider the partition obtained by moving an element $u \in V$ from its current part V_j to some other part $V_{j'}$ for $j' \neq j$. In other words, consider the clustering $h' \in \mathcal{C}$ given by $\big\{V_{j'} \cup \{u\}, V_j \setminus \{u\}\big\} \cup \bigcup_{i\notin\{j,j'\}} \{V_i\}$. Observe that $\mathrm{dist}(h,h') = O(1/\max\{|V_i|\})$ which for a fixed $k = o(n)$ matches $O(1/n)$. On the other hand, for any $v \in V$ and for any $u \in V$ there is a choice of j' so that h and h' obtained as above would disagree on $(u,v) \in \mathcal{X}$. Hence, $\Pr_{\mathcal{D}_{\mathcal{X}}}\left[\mathrm{DIS}\left(\mathcal{B}\left(h, O(1/n)\right)\right)\right] = 1$.

It is also not hard to see that the VC dimension of $\mathcal{C}$ is $\Theta(n)$. Indeed, any full matching over V constitutes a set which is shattered in $\mathcal{C}$ (as long as $k \geq 2$, of course). On the other hand, any set $S \subseteq \mathcal{X}$ of size n must induce an undirected cycle on the elements of V. Clearly the edges of a cycle cannot be shattered by functions in $\mathcal{C}$, because if $h(u_1,u_2) = h(u_2,u_3) = \cdots = h(u_{\ell-1},u_\ell) = 1$ for $h \in \mathcal{C}$, then also $h(u_1,u_\ell) = 1$.

Using Corollary 6, we conclude that we would need $\Omega(n^2)$ preference labels to obtain an (ε,μ)-SRRA for any meaningful pair (ε,μ). This is uninformative as the cardinality of $\mathcal{X}$ is $O(n^2)$. As in the problem discussed in Section 4, this can be improved using Remark 4 to $\Omega(n\nu^{-1})$, which tends to quadratic in n as ν becomes smaller.

We note that also here, uniform sampling is doomed to query the entire pool; this can be easily shown using similar arguments to the ones appear in Ailon (2012, Section 2.4). We next show how to construct more useful SRRA's for the problem, for arbitrarily small ν.

12. Equivalently, assume that $\mathcal{X}$ contains only unordered distinct pairs without any constraint on Y. For notational purposes we preferred to define $\mathcal{X}$ as the set of ordered distinct pairs.

5.3 Better SRRA for Semi-Supervised k-Clustering

Fix $h \in \mathcal{C}$, with $h = \{V_1, \ldots, V_k\}$ (we allow empty V_i's). Order the V_i's with respect to their sizes so that $|V_1| \geq |V_2| \geq \cdots \geq |V_k|$. We construct an ε-SRRA with respect to h as follows. For each cluster $V_i \in h$ and for each element $u \in V_i$ we draw $k - i + 1$ independent samples $S_{ui}, S_{u(i+1)}, \ldots, S_{uk}$ as follows. Each sample S_{uj} is a subset of V_j of size q (to be defined below), chosen uniformly with repetitions from V_j, where

$$q = c_2 \max\left\{\varepsilon^{-2}k^2, \varepsilon^{-3}k\right\} \log n \tag{12}$$

for some global $c_2 > 0$. Note that the collection of pairs $\{(u, v) \in \mathcal{X} : v \in S_{ui} \text{ for some } i\}$ is, roughly speaking, biased in such a way that pairs containing elements in smaller clusters (with respect to h) are more likely to be selected.

We define our estimator f to be, for any $h' \in \mathcal{C}$,

$$f(h') = \sum_{i=1}^{k} \frac{|V_i|}{q} \sum_{u \in V_i} \sum_{v \in S_{ui}} f_{u,v}(h') + 2 \sum_{i=1}^{k} \sum_{u \in V_i} \sum_{j=i+1}^{k} \frac{|V_j|}{q} \sum_{v \in S_{uj}} f_{u,v}(h') \;, \tag{13}$$

where $f_{u,v}(h') = \mathrm{cost}_{u,v}(h') - \mathrm{cost}_{u,v}(h)$ and $\mathrm{cost}_{u,v}(h) = N^{-1}\mathbf{1}_{h(u,v) \neq Y(u,v)}$. Note that the summations over S_{ui} above takes into account multiplicity of elements in the multiset S_{ui}.

Theorem 14 *With probability at least $1 - n^{-3}$ the function f is an ε-SRRA with respect to h.*

Consider another k-clustering $h' \in \mathcal{C}$, with corresponding partitioning $\{V'_1, \ldots, V'_k\}$ of V. We can write $\mathrm{dist}(h, h')$ as

$$\mathrm{dist}(h, h') = \sum_{(u,v) \in \mathcal{X}} \mathrm{dist}_{u,v}(h, h') \,,$$

where $\mathrm{dist}_{u,v}(h, h') = N^{-1}(\mathbf{1}_{h'(u,v)=1}\mathbf{1}_{h(u,v)=0} + \mathbf{1}_{h(u,v)=1}\mathbf{1}_{h'(u,v)=0})$.

Let n_i denote $|V_i|$, and recall that $n_1 \geq n_2 \geq \cdots \geq n_k$. In what follows, we remove the subscript in reg_h and rename it reg (h is held fixed). The function $\mathrm{reg}(h')$ will now be written as:

$$\mathrm{reg}(h') = \sum_{i=1}^{k} \sum_{u \in V_i} \left(\sum_{v \in V_i \setminus \{u\}} \mathrm{reg}_{u,v}(h') + 2 \sum_{j=i+1}^{k} \sum_{v \in V_j} \mathrm{reg}_{u,v}(h') \right) \,,$$

where

$$\mathrm{reg}_{u,v}(h') = \mathrm{cost}_{u,v}(h') - \mathrm{cost}_{u,v}(h) \;.$$

Clearly for each h' it holds that $f(h')$ from (13) is an unbiased estimator of $\mathrm{reg}(h')$. We now analyze its error. For each $i, j \in [k]$ let V_{ij} denote $V_i \cap V'_j$. This captures exactly the set of elements in the i'th cluster in h and the j'th cluster in h'. The distance $\mathrm{dist}(h, h')$ can be written as follows:

$$\mathrm{dist}(h, h') = N^{-1} \left(\sum_{i=1}^{k} \sum_{j=1}^{k} |V_{ij} \times (V_i \setminus V_{ij})| + 2 \sum_{j=1}^{k} \sum_{1 \leq i_1 < i_2 \leq k} |V_{i_1 j} \times V_{i_2 j}| \right) \;. \tag{14}$$

We call each Cartesian set product in (14) a *distance contributing rectangle*. Note that unless a pair (u,v) appears in one of the distance contributing rectangles, we have $\mathrm{reg}_{u,v}(h') = f_{u,v}(h') = 0$. Hence we can decompose $\mathrm{reg}(h')$ and $f(h')$ in correspondence with the distance contributing rectangles, as follows:

$$\mathrm{reg}(h') = \sum_{i=1}^{k}\sum_{j=1}^{k} G_{i,j}(h') + 2\sum_{j=1}^{k}\sum_{1\le i_1<i_2\le k} G_{i_1,i_2,j}(h')\,, \tag{15}$$

$$f(h') = \sum_{i=1}^{k}\sum_{j=1}^{k} F_{i,j}(h') + 2\sum_{j=1}^{k}\sum_{1\le i_1<i_2\le k} F_{i_1,i_2,j}(h')\,, \tag{16}$$

where

$$\begin{aligned} G_{i,j}(h') &= \sum_{u\in V_{ij}}\sum_{v\in V_i\setminus V_{ij}} \mathrm{reg}_{u,v}(h')\,,\\ F_{i,j}(h') &= \frac{|V_i|}{q}\sum_{u\in V_{ij}}\sum_{v\in (V_i\setminus V_{ij})\cap S_{ui}} f_{u,v}(h')\,, \qquad (17)\\ G_{i_1,i_2,j}(h') &= \sum_{u\in V_{i_1 j}}\sum_{v\in V_{i_2 j}} f_{u,v}(h')\,,\\ F_{i_1,i_2,j}(h') &= \frac{|V_{i_2}|}{q}\sum_{u\in V_{i_1 j}}\sum_{v\in V_{i_2 j}\cap S_{ui_2}} f_{u,v}(h')\,. \qquad (18) \end{aligned}$$

(Note that the S_{ui}'s are multisets, and the inner sums in (17) and (18) may count elements multiple times.)

Lemma 15 *With probability at least $1-n^{-3}$, the following holds simultaneously for all $h'\in\mathcal{C}$ and all $i,j\in[k]$:*

$$|G_{i,j}(h') - F_{i,j}(h')| \le \varepsilon N^{-1}\cdot|V_{ij}\times(V_i\setminus V_{ij})|\,. \tag{19}$$

Proof The predicate (19) (for a given i,j) depends only on the set $V_{ij}=V_i\cap V'_j$. Given a subset $B\subseteq V_i$, we say that h' (i,j)-realizes B if $V_{ij}=B$.

Now fix i,j and $B\subseteq V_i$. Assume h' (i,j)-realizes B. Let $\beta=|B|$ and $\gamma=|V_i|$. Consider the random variable $F_{i,j}(h')$. Think of the sample S_{ui} as a sequence $S_{ui}(1),\ldots,S_{ui}(q)$, where each $S_{ui}(s)$ is chosen uniformly at random from V_i for $s=1,\ldots,q$. We can now rewrite $F_{i,j}(h')$ as follows:

$$F_{i,j}(h') = \frac{\gamma}{q}\sum_{u\in B}\sum_{s=1}^{q} Z\big(S_{ui}(s)\big)\,,$$

where

$$Z(v) = \begin{cases} f_{u,v}(h') & v\in V_i\setminus V_{ij}\\ 0 & \text{otherwise}\end{cases}\,.$$

For all $s = 1, \dots q$ the random variable $Z\big(S_{ui}(s)\big)$ is bounded by $2N^{-1}$ almost surely, and its moments satisfy:

$$E\left[Z\big(S_{ui}(s)\big)\right] = \frac{1}{\gamma} \sum_{v \in (V_i \setminus V_{ij})} f_{u,v}(h')\,,$$
$$E\left[Z\big(S_{ui}(s)\big)^2\right] \leq \frac{4N^{-2}(\gamma-\beta)}{\gamma}\ .$$

From this we conclude using Bernstein inequality that for any $t \leq 6\,N^{-1}\beta(\gamma-\beta)$,

$$\Pr\left[\left|F_{i,j}(h') - G_{i,j}(h')\right| \geq t\right] \leq \exp\left\{-\frac{qt^2}{16\gamma\beta(\gamma-\beta)N^{-2}}\right\}\,.$$

Plugging in $t = \varepsilon N^{-1}\beta(\gamma-\beta)$, we conclude

$$\Pr\left[\left|F_{i,j}(h') - G_{i,j}(h')\right| \geq \varepsilon N^{-1}\beta(\gamma-\beta)\right] \leq \exp\left\{-\frac{q\varepsilon^2\beta(\gamma-\beta)}{16\gamma}\right\}\,.$$

Now note that the number of possible sets $B \subseteq V_i$ of size β is at most $n^{\min\{\beta,\gamma-\beta\}}$. Using union bound and recalling our choice of q, the lemma follows. ■

Proving the following is more involved.

Lemma 16 *With probability at least $1 - n^{-3}$, the following holds uniformly for all $h' \in \mathcal{C}$ and for all $i_1, i_2, j \in [k]$ with $i_1 < i_2$:*

$$|F_{i_1,i_2,j}(h') - G_{i_1,i_2,j}(h')| \leq \varepsilon N^{-1} \max\left\{|V_{i_1 j} \times V_{i_2 j}|, \frac{|V_{i_1 j} \times (V_{i_1} \setminus V_{i_1 j})|}{k}, \frac{|V_{i_2 j} \times (V_{i_2} \setminus V_{i_2 j})|}{k}\right\}\,. \tag{20}$$

Proof The predicate (20) (for a given i_1, i_2, j) depends only on the sets $V_{i_1 j} = V_{i_1} \cap V'_j$ and $V_{i_2 j} = V_{i_2} \cap V'_j$. Given subsets $B_1 \subseteq V_{i_1}$ and $B_2 \subseteq V_{i_2}$, we say that h' (i_1, i_2, j)-realizes (B_1, B_2) if $V_{i_1 j} = B_1$ and $V_{i_2 j} = B_2$.

We now fix $i_1 < i_2, j$ and $B_1 \subseteq V_{i_1}$, $B_2 \subseteq V_{i_2}$. Assume h' (i_1, i_2, j)-realizes (B_1, B_2). For brevity, denote $\beta_\iota = |B_\iota|$ and $\gamma_\iota = |V_{i_\iota}|$ for $\iota = 1, 2$. Using Bernstein inequality as in Lemma 15, we conclude the following two inequalities:

$$\Pr\left[\left|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')\right| > t\right] \leq \exp\left\{-\frac{c_3 t^2 q}{\beta_1\beta_2\gamma_2 N^{-2}}\right\} \tag{21}$$

for any t in the range $\left[0, N^{-1}\beta_1\beta_2\right]$, and some global $c_3 > 0$. For t in the range $(N^{-1}\beta_1\beta_2, \infty)$ and some global c_4 we have

$$\Pr\left[\left|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')\right| > t\right] \leq \exp\left\{-\frac{c_4 t q}{\gamma_2 N^{-1}}\right\}\ . \tag{22}$$

Consider the following three cases.

1. $\beta_1\beta_2 \geq \max\{\beta_1(\gamma_1-\beta_1)/k, \beta_2(\gamma_2-\beta_2)/k\}$. Hence, $\beta_1 \geq (\gamma_2-\beta_2)/k, \beta_2 \geq (\gamma_1-\beta_1)/k$. In this case, we can plug $t = \varepsilon N^{-1}\beta_1\beta_2$ in (21) to get

$$\Pr\left[\left|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')\right| > \varepsilon N^{-1}\beta_1\beta_2\right] \leq \exp\left\{-\frac{c_3\varepsilon^2\beta_1\beta_2 q}{\gamma_2}\right\} . \tag{23}$$

Consider two subcases: (i) If $\beta_2 \geq \gamma_2/2$ then the RHS of (23) is at most $\exp\left\{-\frac{c_3\varepsilon^2\beta_1 q}{2}\right\}$. The number of possible subsets B_1, B_2 of sizes β_1, β_2 respectively is clearly at most $n^{\beta_1+(\gamma_2-\beta_2)} \leq n^{\beta_1+k\beta_1}$. Therefore, as long as $q = O(\varepsilon^{-2}k\log n)$ then with probability at least $1-n^{-6}$ this case is taken care of in the following sense: Simultaneously for all $j, i_1 < i_2$, all possible $\beta_1 \leq \gamma_1 = |V_{i_1}|$, $\beta_2 \leq \gamma_2 = |V_{i_2}|$ satisfying the assumptions and for all $B_1 \subseteq V_{i_1 j}, B_2 \subseteq V_{i_2 j}$ of sizes β_1, β_2 respectively and for all h' (i_1, i_2, j)-realizing (B_1, B_2) we have that $|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')| \leq \varepsilon\beta_1\beta_2$.

(ii) If $\beta_2 < \gamma_2/2$ then by our assumption, $\beta_1 \geq \gamma_2/2k$. Hence the RHS of (23) is at most $\exp\left\{-\frac{c_3\varepsilon^2\beta_2 q}{2k}\right\}$. The number of sets B_1, B_2 of sizes β_1, β_2 respectively is clearly at most $n^{(\gamma_1-\beta_1)+\beta_2} \leq n^{\beta_2(1+k)}$. Therefore, as long as $q = O(\varepsilon^{-2}k^2\log n)$ then with probability at least $1-n^{-6}$ this case is taken care of in the following sense: Simultaneously for all $j, i_1 < i_2$, all possible $\beta_1 < \gamma_1 = |V_{i_1}|$, $\beta_2 < \gamma_2 = |V_{i_2}|$ satisfying the assumptions and for all $B_1 \subseteq V_{i_1 j}, B_2 \subseteq V_{i_2 j}$ of sizes β_1, β_2 respectively and for all h' (i_1, i_2, j)-realizing (B_1, B_2) we have that $|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')| \leq \varepsilon\beta_1\beta_2$.

The requirement $q = O(\varepsilon^{-2}k\log n)$ is satisfied by our choice, Equation 12.

2. $\beta_2(\gamma_2-\beta_2)/k \geq \max\{\beta_1\beta_2, \beta_1(\gamma_1-\beta_1)/k\}$. We consider two subcases.

   (a) $\varepsilon\beta_2(\gamma_2-\beta_2)/k \leq \beta_1\beta_2$. Using (21), we get

$$\Pr[|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')| > \varepsilon N^{-1}\beta_2(\gamma_2-\beta_2)/k] \leq \exp\left\{-\frac{c_3\varepsilon^2\beta_2(\gamma_2-\beta_2)^2 q}{k^2\beta_1\gamma_2}\right\} . \tag{24}$$

   Again consider two subcases. (i) $\beta_2 \leq \gamma_2/2$. In this case we conclude from Equation 24

$$\Pr[|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')| > \varepsilon N^{-1}\beta_2(\gamma_2-\beta_2)/k] \leq \exp\left\{-\frac{c_3\varepsilon^2\beta_2\gamma_2 q}{4k^2\beta_1}\right\} . \tag{25}$$

   Now note that by our assumption

$$\beta_1 \leq (\gamma_2-\beta_2)/k \leq \gamma_2/k \leq \gamma_1/k , \tag{26}$$

   the last inequality is in virtue of our assumption $\gamma_1 \geq \gamma_2$. Also by assumption,

$$\beta_1 \leq \beta_2(\gamma_2-\beta_2)/(\gamma_1-\beta_1) \leq \beta_2\gamma_2/(\gamma_1-\beta_1) . \tag{27}$$

   Plugging (26) in the RHS of (27), we conclude that

$$\beta_1 \leq \beta_2\gamma_2/(\gamma_1(1-1/k)) \leq 2\beta_2\gamma_2/\gamma_1 \leq 2\beta_2 .$$

   From here we conclude that the RHS of (25) is at most $\exp\left\{-\frac{c_3\varepsilon^2\gamma_2 q}{8k^2}\right\}$.

The number of sets B_1, B_2 of sizes β_1, β_2 respectively is clearly at most $n^{\beta_1+\beta_2} \leq n^{2\beta_2+\beta_2} \leq n^{3\gamma_2}$. Hence, as long as $q = O(\varepsilon^{-2}k^2 \log n)$ (satisfied by our assumption), with probability at least $1 - n^{-6}$ simultaneously for all $j, i_1 < i_2$, all possible $\beta_1 \leq \gamma_1 = |V_{i_1}|$, $\beta_2 \leq \gamma_2 = |V_{i_2}|$ satisfying the assumptions and for all $B_1 \subseteq V_{i_1 j}, B_2 \subseteq V_{i_2 j}$ of sizes β_1, β_2 respectively and for all h' (i_1, i_2, j)-realizing (B_1, B_2) we have that $|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')| \leq \varepsilon\beta_2(\gamma_2 - \beta_2)/k$. In the second subcase (ii) $\beta_2 > \gamma_2/2$. The RHS of (24) is at most $\exp\left\{-\frac{2c_3\varepsilon^2(\gamma_2-\beta_2)^2 q}{k^2\beta_1}\right\}$. By our assumption, $(\gamma_2-\beta_2)/(k\beta_1) \geq 1$, hence this is at most $\exp\left\{-\frac{2c_3\varepsilon^2(\gamma_2-\beta_2)q}{k}\right\}$. The number of sets B_1, B_2 of sizes β_1, β_2 respectively is clearly at most $n^{\beta_1+(\gamma_2-\beta_2)} \leq n^{(\gamma_2-\beta_2)/k+(\gamma_2-\beta_2)} \leq n^{2(\gamma_2-\beta_2)}$. Therefore, as long as $q = O(\varepsilon^{-2}k \log n)$ (satisfied by our assumption), then with probability at least $1 - n^{-6}$, using a similar counting and union bound argument as above, this case is taken care of in the sense that: $|G_{i_1,i_2,j}(h') - G_{i_1,i_2,j}(h')| \leq \varepsilon\beta_2(\gamma_2 - \beta_2)/k$.

(b) $\varepsilon\beta_2(\gamma_2 - \beta_2)/k > \beta_1\beta_2$. We now use (22) to conclude

$$\Pr[|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')| > \varepsilon N^{-1}\beta_2(\gamma_2 - \beta_2)/k] \leq \exp\left\{-\frac{c_4\varepsilon\beta_2(\gamma_2 - \beta_2)q}{k\gamma_2}\right\} . \tag{28}$$

We again consider the cases (i) $\beta_2 \leq \gamma_2/2$ and (ii) $\beta_2 > \gamma_2/2$ as above. In (i), we get that the RHS of (28) is at most $\exp\left\{-\frac{c_4\varepsilon\beta_2 q}{2k}\right\}$. Now notice that by our assumptions,

$$\beta_1 < \varepsilon(\gamma_2 - \beta_2)/k \leq \gamma_2/2 \leq \gamma_1/2 . \tag{29}$$

Also by our assumptions, $\beta_1 < \beta_2(\gamma_2 - \beta_2)/(\gamma_1 - \beta_1)$, which by (29) is at most $2\beta_2\gamma_2/\gamma_1 \leq 2\beta_2$. Hence the number of possibilities for B_1, B_2 is at most $n^{\beta_1+\beta_2} \leq n^{3\beta_2}$. In (ii), we get that the RHS of (28) is at most $\exp\left\{-\frac{c_4\varepsilon(\gamma_2-\beta_2)q}{2k}\right\}$, and the number of possibilities for B_1, B_2 is at most $n^{\beta_1+(\gamma_2-\beta_2)}$ which is bounded by $n^{2(\gamma_2-\beta_2)}$ by our assumptions. For both (i) and (ii) taking $q = O(\varepsilon^{-1}k \log n)$ ensures with probability at least $1-n^{-6}$, using a similar counting and union bounding argument as above, case (b) is taken care of in the sense that: $|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')| \leq \varepsilon N^{-1}\beta_2(\gamma_2 - \beta_2)/k$.

3. $\beta_1(\gamma_1 - \beta_1)/k \geq \max\{\beta_1\beta_2, \beta_2(\gamma_2 - \beta_2)/k\}$. We consider two subcases.

(a) $\varepsilon\beta_1(\gamma_1 - \beta_1)/k \leq \beta_1\beta_2$. Using (21), we get

$$\Pr\left[|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')| > \varepsilon N^{-1}\beta_1(\gamma_1 - \beta_1)/k\right] \leq \exp\left\{-\frac{c_3\varepsilon^2\beta_1(\gamma_1 - \beta_1)^2 q}{k^2\beta_2\gamma_2}\right\} . \tag{30}$$

As before, consider case (i) in which $\beta_2 \leq \gamma_2/2$ and (ii) in which $\beta_2 > \gamma_2/2$. For case (i), we use the fact that $\beta_1(\gamma_1 - \beta_1) \geq \beta_2(\gamma_2 - \beta_2)$ by assumption and notice that the RHS of (30) is at most $\exp\left\{-\frac{c_3\varepsilon^2\beta_2(\gamma_2-\beta_2)(\gamma_1-\beta_1)q}{k^2\beta_2\gamma_2}\right\}$. This is hence at most $\exp\left\{-\frac{c_3\varepsilon^2(\gamma_1-\beta_1)q}{2k^2}\right\}$. The number of possibilities of B_1, B_2 of sizes β_1, β_2 is clearly at most

$n^{(\gamma_1-\beta_1)+\beta_2} \leq n^{(\gamma_1-\beta_1)+(\gamma_1-\beta_1)/k} \leq n^{2(\gamma_1-\beta_1)}$. From this we conclude that $q = O(\varepsilon^{-2}k^2 \log n)$ suffices for this case. For case (ii), we bound the RHS of (30) by $\exp\left\{-\frac{c_3\varepsilon^2\beta_1(\gamma_1-\beta_1)^2 q}{2k^2\beta_2^2}\right\}$. Using the assumption that $(\gamma_1-\beta_1)/\beta_2 \geq k$, the latter expression is upper bounded by $\exp\left\{-\frac{c_3\varepsilon^2\beta_1 q}{2}\right\}$. Again by our assumptions,

$$\beta_1 \geq \beta_2(\gamma_2-\beta_2)/(\gamma_1-\beta_1) \geq (\varepsilon(\gamma_1-\beta_1)/k)(\gamma_2-\beta_2)/(\gamma_1-\beta_1) = \varepsilon(\gamma_2-\beta_2)/k \ . \tag{31}$$

The number of possibilities of B_1, B_2 of sizes β_1, β_2 is clearly at most $n^{\beta_1+(\gamma_2-\beta_2)}$ which by (31) is bounded by $n^{\beta_1+k\beta_1/\varepsilon} \leq n^{2k\beta_1/\varepsilon}$. From this we conclude that as long as $q = O(\varepsilon^{-3}k\log n)$ (satisfied by our choice), this case is taken care of in sense repeatedly explained above.

(b) $\varepsilon\beta_1(\gamma_1-\beta_1)/k > \beta_1\beta_2$. Using (22), we get

$$\Pr\left[\left|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')\right| > \varepsilon N^{-1}\beta_1(\gamma_1-\beta_1)/k\right] \leq \exp\left\{-\frac{c_4\varepsilon\beta_1(\gamma_1-\beta_1)q}{k\gamma_2}\right\} \ . \tag{32}$$

We consider two sub-cases, (i) $\beta_1 \leq \gamma_1/2$ and (ii) $\beta_1 > \gamma_1/2$. In case (i), we have that

$$\begin{aligned}\frac{\beta_1(\gamma_1-\beta_1)}{\gamma_2} &= \frac{1}{2}\frac{\beta_1(\gamma_1-\beta_1)}{\gamma_2} + \frac{1}{2}\frac{\beta_1(\gamma_1-\beta_1)}{\gamma_2} \\ &\geq \frac{1}{2}\frac{\beta_1\gamma_1}{2\gamma_2} + \frac{1}{2}\frac{\beta_2(\gamma_2-\beta_2)}{\gamma_2} \\ &\geq \beta_1/4 + \min\{\beta_2, \gamma_2-\beta_2\}/2 \ .\end{aligned}$$

(The last step used $\gamma_1 \geq \gamma_2$.) Hence, the RHS of (32) is bounded above by

$$\exp\left\{-\frac{c_4\varepsilon q(\beta_1/4 + \min\{\beta_2, \gamma_2-\beta_2\}/2)}{k}\right\} \ .$$

The number of possibilities of B_1, B_2 of sizes β_1, β_2 is clearly at most $n^{\beta_1+\min\{\beta_2,\gamma_2-\beta_2\}}$, hence as long as $q = O(\varepsilon^{-1}k\log n)$ (satisfied by our choice), this case is taken care of in the sense repeatedly explained above. In case (ii), we can upper bound the RHS of (32) by $\exp\left\{-\frac{c_4\varepsilon\gamma_1(\gamma_1-\beta_1)q}{2k\gamma_2}\right\} \geq \exp\left\{-\frac{c_4\varepsilon(\gamma_1-\beta_1)q}{2k}\right\}$. The number of possibilities of B_1, B_2 of sizes β_1, β_2 is clearly at most $n^{(\gamma_1-\beta_1)+\beta_2}$ which, using our assumptions, is bounded above by $n^{(\gamma_1-\beta_1)+(\gamma_1-\beta_1)/k} \leq n^{2(\gamma_1-\beta_1)}$. Hence, as long as $q = O(\varepsilon^{-1}k\log n)$, this case is taken care of in the sense repeatedly explained above.

This concludes the proof of the lemma. ■

As a consequence, we get the following:

Lemma 17 *with probability at least* $1-n^{-3}$*, the following holds simultaneously for all* k*-clusterings* $\mathcal{C}'$*:* $|\mathrm{reg}(h') - f(h')| \leq 5\varepsilon\mathrm{dist}(h', h)$.

Proof By the triangle inequality,

$$|\mathrm{reg}(h') - f(h')| \leq \sum_{i=1}^{k}\sum_{j=1}^{k}|G_{i,j}(h') - F_{i,j}(h')| + 2\sum_{j=1}^{k}\sum_{1\leq i_1<i_2\leq k}|G_{i_1,i_2,j}(h') - F_{i_1,i_2,j}(h')|\,.$$

Using (15)-(16), then Lemmas 15 and 16 (assuming success of a high probability event), and rearranging the sum and finally using (14), we get:

$$\begin{aligned}
|\mathrm{reg}(h') - f(h')| \leq & \sum_{i=1}^{k}\sum_{j=1}^{k}\varepsilon N^{-1}|V_{ij}\times(V_i\setminus V_{ij})| \\
& + 2\varepsilon N^{-1}\sum_{j=1}^{k}\sum_{i_1<i_2}\left(|V_{i_1j}\times V_{i_2j}| + k^{-1}|V_{i_1j}\times(V_{i_1}\setminus V_{i_1j})| + k^{-1}|V_{i_2j}\times(V_{i_2}\setminus V_{i_2j})|\right) \\
\leq & \sum_{i=1}^{k}\sum_{j=1}^{k}\varepsilon N^{-1}|V_{ij}\times(V_i\setminus V_{ij})| + 2\varepsilon N^{-1}\sum_{j=1}^{k}\sum_{i_1<i_2}|V_{i_1j}\times V_{i_2j}| \\
& + 2\varepsilon N^{-1}\sum_{j=1}^{k}\sum_{i_1<i_2}^{k}k^{-1}|V_{i_1j}\times(V_{i_1}\setminus V_{i_1j})| + 2\varepsilon N^{-1}\sum_{j=1}^{k}\sum_{i_1<i_2}^{k}k^{-1}|V_{i_2j}\times(V_{i_2}\setminus V_{i_2j})| \\
\leq & \sum_{i=1}^{k}\sum_{j=1}^{k}\varepsilon N^{-1}|V_{ij}\times(V_i\setminus V_{ij})| + 2\varepsilon N^{-1}\sum_{j=1}^{k}\sum_{i_1<i_2}|V_{i_1j}\times V_{i_2j}| \\
& + 2\varepsilon N^{-1}\sum_{j=1}^{k}\sum_{i_1=1}^{k}kk^{-1}|V_{i_1j}\times(V_{i_1}\setminus V_{i_1j})| + 2\varepsilon N^{-1}\sum_{j=1}^{k}\sum_{i_2=1}^{k}kk^{-1}|V_{i_2j}\times(V_{i_2}\setminus V_{i_2j})| \\
& \leq 5\varepsilon N^{-1}\sum_{i=1}^{k}\sum_{j=1}^{k}|V_{ij}\times(V_i\setminus V_{ij})| + \varepsilon N^{-1}\sum_{j=1}^{k}\sum_{i_1<i_2}|V_{i_1j}\times V_{i_2j}| \leq 5\varepsilon\mathrm{dist}(h,h')\,,
\end{aligned}$$

as required. ■

We conclude that f is an ε-SRRA estimator. Its construction pseudo code is presented for convenience in Algorithm 3. Clearly the number of label queries required for obtaining this ε-SRRA estimator is $O(n\max\{\varepsilon^{-2}k^3,\varepsilon^{-3}k^2\}\log n)$. Combining Theorem 14 with this bound and the iterative algorithm described in Corollary 4 (Algorithm 1), we obtain the following:

Corollary 18 *There exists an active learning algorithm for obtaining a solution $h\in\mathcal{C}$ for semi-supervised k-clustering with $\mathrm{er}_{\mathcal{D}}(h)\leq(1+O(\varepsilon))\,\nu$ with total query complexity of $O\left(n\max\left\{\varepsilon^{-2}k^3,\varepsilon^{-3}k^2\right\}\log^2 n\right)$. The algorithm succeeds with success probability at least $1-n^{-2}$.*

We do not believe the ε^{-3} factor in the corollary is tight, and speculate that it should be reduced to ε^{-2} using more advanced measure concentration tools. Note that we assume k is fixed. Indeed, in practice, k is often taken to be constant (or at most $o(n)$). Thus, the sample complexity of our active learning method using these direct SRRA constructions is almost linear in n. As in the case of Corollary 13 and the ensuing discussion around LRPP, the result in Corollary 18 significantly beats known active learning results depending only on disagreement coefficient and VC dimension bounds, for small ν.

Algorithm 3 SRRA for Semi-Supervised k-Clustering

Input: V, k, $\mathcal{C}$, a pivot $h = \{V_i\}_{i=1}^k \in \mathcal{C}$, estimation parameter $\epsilon \in (0, 1/5)$

$q \leftarrow O(\max\left\{\varepsilon^{-2}k^2, \varepsilon^{-3}k\right\}\log n)$
Index the clusters of h such that $|V_1| \geq |V_2| \geq \ldots \geq |V_k|$
for $u \in V_i,\ i = 1, \ldots, k$ **do**
  for $j = i, \ldots, k$ **do**
    $S_{u,j} \leftarrow$ sample q elements from V_j independently and uniformly with repetitions
  end for
end for
return $f : \mathcal{C} \longrightarrow \mathbb{R}$, defined by

$$f(h') = \sum_{i=1}^{k} \frac{|V_i|}{q} \sum_{u \in V_i} \sum_{v \in S_{ui}} \left(\mathrm{cost}_{u,v}(h') - \mathrm{cost}_{u,v}(h)\right) + 2\sum_{i=1}^{k} \sum_{u \in V_i} \sum_{j=i+1}^{k} \frac{|V_j|}{q} \sum_{v \in S_{uj}} \left(\mathrm{cost}_{u,v}(h') - \mathrm{cost}_{u,v}(h)\right)$$

6. Additional Results and Practical Considerations

We discuss two practical extensions of our results.

6.1 LRPP over Linearly Induced Permutations in Constant Dimensional Feature Space

A special class of interest is known as LRPP over linearly induced permutations in constant dimensional feature space. We use the same definition of $\mathcal{X}$ as in Section 4, except that now each point $v \in V$ is associated with a feature vector, which we denote using bold face: $\mathbf{v} \in \mathbb{R}^d$. The concept space $\mathcal{C}$ now consists only of permutations π such that there exists a vector $\mathbf{w}_\pi \in \mathbb{R}^d$ satisfying

$$\pi(u, v) = 1 \iff \langle \mathbf{w}_\pi, \mathbf{u} - \mathbf{v} \rangle > 0 \ . \tag{33}$$

We are assuming familiarity with the theory of geometric arrangements, and refer the reader to de Berg et al. (2008) for further details. Geometrically, each $(u, v) \in \mathcal{X}$ is viewed as a halfspace $H_{u,v} = \{\mathbf{x} : \langle \mathbf{x}, \mathbf{u} - \mathbf{v} \rangle > 0\}$, whose (closure) supporting hyperplane is $h_{u,v} = \{\mathbf{x} : \langle \mathbf{x}, \mathbf{u} - \mathbf{v} \rangle = 0\}$. Let $\mathcal{H}$ be the collection of these $\binom{n}{2}$ hyperplanes $\{h_{u,v} : (u, v) \in \mathcal{X}\}$.[13] The collection $\mathcal{C}$ corresponds to the maximal dimensional cells in the underlying arrangement $\mathcal{A}(\mathcal{H})$. We thus call $\mathcal{A}(\mathcal{H})$ from now on the *permutation arrangement*, and we naturally identify full dimensional cells with their induced permutations. We denote by $\mathbb{C}_\pi \subseteq \mathbb{R}^d$ the unique cell corresponding to a permutation $\pi \in \mathcal{C}$.

13. Note that $h_{u,v} = h_{v,u}$.

6.1.1 Bounding the VC Dimension and Disagreement Coefficient

Using standard tools from combinatorial geometry, the VC dimension of $\mathcal{C}$ is at most $d-1$. Roughly speaking, this property follows from the fact that in an arrangement of m hyperplanes in d-space, each of which meeting the origin, the overall number of cells is at most $O(m^{d-1})$, see de Berg et al. (2008).

As for the uniform disagreement coefficient, we show below that it is bounded by $O(n)$. Let $\pi \in \mathcal{C}$ be a permutation with a corresponding cell $\mathbb{C}_\pi$ in $\mathcal{A}(\mathcal{H})$. The ball $\mathcal{B}(\pi, r)$ is, geometrically, the closure of the union of all cells corresponding to "realizable" permutations σ satisfying $\text{dist}(\sigma, \pi) \leq r$. The corresponding disagreement region $\text{DIS}(\mathcal{B}(\pi, r))$ corresponds to the set of ordered pairs (halfspaces) intersecting $\mathcal{B}(\pi, r)$. In fact, in this case, all these halfspaces have the property that their bounding hyperplane h meets $\mathcal{B}(\pi, r)$. Indeed, if that hyperplane is located outside $\mathcal{B}(\pi, r)$ then there is a clear agreement among all cells (hypeotheses) of $\mathcal{B}(\pi, r)$ with respect to H, as all of them are located at the same side of h. We next show:

Proposition 19 *The measure of* $\text{DIS}\left(\mathcal{B}(\pi, r)\right)$ *in* $\mathcal{D}_{\mathcal{X}}$ *is at most* $8rn$.

Proof By Diaconis and Graham (1977), the Spearman Footrule distance between any two permutations π and σ is at most twice $N\text{dist}(\pi, \sigma)$, where $N = n(n-1)$. Hence, if $\text{dist}(\pi, \sigma)$ is r, then any element u could only swap locations with a set of elements located up to $2rN$ positions away to the right or to the left. This yields a total of $4rN$ 'swap-candidates' for each u. Thus, at most $4rNn$ inversions are possible. Note that each inversion corresponds to a hyperplane (unordered pair) that we cross, and thus the total number of ordered pairs is at most $8rNn$. The probability measure of this set is at most $8rn$, because we assign equal probability of N^{-1} for each possible pair in $\mathcal{X}$. The result follows. ■

By the proposition we have that the disagreement coefficient θ is always bounded by $O(n)$, establishing our bound. We now invoke Corollary 6 with $\mu = O(1/n^2)$ (which is tantamount to $\mu = 0$ for this problem, because $|\mathcal{X}| = O(n^2)$ and we are using the uniform measure), and conclude:

Theorem 20 *An* ε*-SRRA for LRPP in linearly induced permutations in* d *dimensional feature space can be constructed, with respect to any* $\pi \in \mathcal{C}$*, with probability at least* $1-\delta$*, using at most* $O\left(nd\varepsilon^{-2}\log^2 n + n\varepsilon^{-2}(\log n)\left(\log(\delta^{-1}\log n)\right)\right)$ *label queries.*

Combining Theorem 20, and the iterative algorithm described in Corollary 4:

Corollary 21 *There exists an algorithm for obtaining a solution* $\pi \in \mathcal{C}$ *for LRPP in linearly induced permutations in* d *dimensional feature space with* $\text{er}_{\mathcal{D}}(\pi) \leq (1 + O(\varepsilon))\,\nu$ *with total query complexity of*

$$O\Big(\varepsilon^{-2}nd\log^3 n + n\varepsilon^{-2}(\log^2 n)\left(\log(\delta^{-1}\log n)\right)\Big). \tag{34}$$

The algorithm succeeds with success probability at least $1-\delta$.

We compare this bound to that of Corollary 13. For the sake of comparison, assume $\delta = n^{-2}$, so that (34) takes the simpler form of $O\big(\varepsilon^{-2} nd \log^3 n / \log(1/\varepsilon)\big)$. This bound is better than that of Corollary 13 as long as the feature space dimension d is $O(\varepsilon^{-2}\log^2 n)$. For larger dimensions, Corollary 13 gives a better bound. It would be interesting to obtain a smoother interpolation between the *geometric* structure coming from the feature space and the *combinatorial* structure coming from permutations.

We conclude the discussion with a polynomial time algorithm to construct the disagreement region obtained in this setting.

6.1.2 An Algorithm for Constructing the Disagreement Region

We first recall that the disagreement region $\mathrm{DIS}(\mathcal{B}(\pi, r))$ corresponds to the set of bounding hyperplanes h that meet $\mathcal{B}(\pi, r)$. Our goal is thus to compute this set of hyperplanes, their side of space containing π will then correspond to the desired set of halfspaces.

We next introduce, for the sake of analysis, another (absolute) measure related to our previous definitions: We define the *absolute distance* of a hyperplane h from a point p in a cell $\mathbb{C}_\pi$ of $\mathcal{A}(\mathcal{H})$ to be the smallest number of hyperplanes, whose removal makes h visible to p, implying that h appears on the boundary of the cell containing p in the arrangement of this subset of hyperplanes. In what follows, and with a slight abuse of notation, we will sometimes refer to the absolute distance of h from a permutation π (instead of a point p in the cell $\mathbb{C}_\pi$).

By the definition of the measure r it follows that when a hyperplane h intersects $\mathcal{B}(\pi, r)$ this implies that the absolute distance between h and any point in $\mathbb{C}_\pi$ is at most $\bar{r} := Nr$, where $N = n(n-1)$. We also observe that, by definition, at least one cell in $\mathcal{B}(\pi, r)$ must have h on its boundary.

In order to compute all hyperplanes meeting $\mathcal{B}(\pi, r)$ we construct this set iteratively. At the first iteration, we compute all hyperplanes with (absolute) distance 0 from π—these hyperplanes are precisely those that define the boundary of $\mathbb{C}_\pi$. Then we remove them from further consideration, and recursively compute the next set of hyperplanes at (absolute) distance 0 from π in the arrangement of the residual hyperplanes. We stop after $\bar{r}$ iterations. By definition, at the ith step, we compute the set of hyperplanes at absolute distance i from π. Clearly, the number of iterations is at most $|\mathcal{H}| = \binom{n}{2}$, so it is sufficient to show that each iteration can be performed in polynomial time.

We thus face the following problem. Given a set of hyperplanes $\mathcal{H}$ and a point p, determine in polynomial time the subset of hyperplanes in $\mathcal{H}$ that define the cell $\mathbb{C}_p$ of p in the arrangement $\mathcal{A}(\mathcal{H})$. This task can easily be done by constructing the entire arrangement $\mathcal{A}(\mathcal{H})$ or even just the cell containing p. Nevertheless, these constructions take time $n^{\Theta(d)}$ (see, e.g., Sharir and Agarwal 1995), and thus may be inefficient for large values of d. We thus present below a simple approach that circumvents the need to construct these structures, our key idea is to use *linear programming solvers*, and is a variant of the analysis in Ezra and Fine (2007).

To this end, fix a hyperplane $h \in \mathcal{H}$. In order to determine whether h appears on the boundary of the cell containing p, we proceed as follows. For each hyperplane $h' \in \mathcal{H} \setminus \{h\}$ we assign the corresponding halfspace H' bounded by h' and containing p, so p lies at the same side (positive or negative) of these hyperplanes (after, e.g., applying a proper

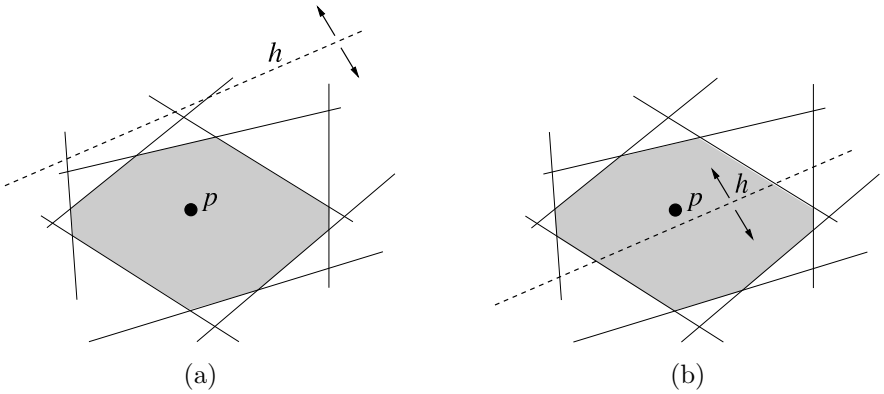


Figure 1: (a) The hyperplane h (depicted by the dashed line in the figure) does not meet the cell containing the point p. (b) The hyperplane h meets that cell. In this case p lies in the portion of that cell located above h.

orientation). Let $\mathcal{H}'$ be the set of these halfspaces. Then we create two additional halfspaces, H^+ and H^- containing the corresponding positive and negative sides of h. We now observe that if h does not appear on the boundary of $\mathbb{C}_p$, then either $\bigcap_{H' \in \mathcal{H}'} H' \bigcap H^+ = \emptyset$ or $\bigcap_{H' \in \mathcal{H}'} H' \bigcap H^- = \emptyset$. Indeed, $\bigcap_{H' \in \mathcal{H}'} H'$ is precisely the cell containing p in $\mathcal{A}(\mathcal{H} \setminus \{h\})$, and if h does not appear on the boundary of $\mathbb{C}_p$ (the cell containing p in $\mathcal{A}(\mathcal{H})$) then we must have that h and $\bigcap_{H' \in \mathcal{H}'} H'$ are disjoint, and thus $\bigcap_{H' \in \mathcal{H}'} H'$ is fully contained in one side of h. If h appears on the boundary of $\mathbb{C}_p$ then both intersections $\bigcap_{H' \in \mathcal{H}'} H' \bigcap H^+$, $\bigcap_{H' \in \mathcal{H}'} H' \bigcap H^-$ must not be empty. In fact, in this case h splits the cell containing p in $\mathcal{A}(\mathcal{H} \setminus \{h\})$ into the two portions $\bigcap_{H' \in \mathcal{H}'} H' \bigcap H^+$, $\bigcap_{H' \in \mathcal{H}'} H' \bigcap H^-$, one of which contains p. See Figure 1 for an illustration. In order to determine if each of these intersections is empty, we construct the corresponding linear programming system, where the constraints are the halfspaces and the objective function is arbitrary, and solve it in polynomial time, using, for example, the ellipsoid method or the interior point method (Grötschel et al., 1988; Karmarkar, 1984; Khačiyan, 1979).

It thus follows that for each hyperplane $h \in H$ we need to solve two linear programming systems in order to determine whether h defines $\mathbb{C}_p$. When we terminate iterating over all $h \in \mathcal{H}$ we have at hand the desired set of hyperplanes of $\mathcal{H}$ that define $\mathbb{C}_p$. Thus the running time of the entire process is polynomial, since at each iteration we spend polynomial time. We have thus obtained:

Theorem 22 *Given a set $\mathcal{H}$ of hyperplanes and a point p in d-space, one can determine in polynomial time the subset of hyperplanes that define the cell in the arrangement $\mathcal{A}(\mathcal{H})$ containing p, where the degree of the polynomial is an absolute constant that does not depend on the dimension d.*

Based on the iterative algorithm presented above, we thus conclude:

Corollary 23 *Given a set $\mathcal{H}$ of hyperplanes in d-space, a permutation π corresponding to a cell $\mathbb{C}_\pi$ in the arrangement $\mathcal{A}(\mathcal{H})$, and a parameter $0 \leq r \leq 1$, one can determine*

in polynomial time the subset of hyperplanes intersecting $\mathcal{B}(\pi, r)$, where the degree of the polynomial is an absolute constant that does not depend on the dimension d.

Remark: We note that since the degree of the polynomial at the running time does not depend on d, we can assume the dimension d is arbitrarily large.

6.2 Convex Relaxations

So far we focused on theoretical ERM aspects only. Doing so, we made no assumptions about the computability of the step $h_i = \operatorname{argmin}_{h' \in \mathcal{C}} f_{h_{i-1}}(h')$ in Corollary 4 (Step 2 in Algorithm 1). Although ERM results are interesting in their own right, we take an additional step and consider convex relaxations.

Instead of optimizing $\mathrm{er}_{\mathcal{D}}(h)$ over the set $\mathcal{C}$, assume we are interested in optimizing $\mathrm{er}_{\mathcal{D}}(\tilde{h})$ over $\tilde{h} \in \widetilde{\mathcal{C}}$, where $\widetilde{\mathcal{C}}$ is a convex set of functions from $\mathcal{X}$ to $\mathbb{R}$. Also assume there is a mapping $\phi : \widetilde{\mathcal{C}} \mapsto \mathcal{C}$ which is used as a "rounding" procedure. For example, in the setting of Section 6.1 the set $\widetilde{\mathcal{C}}$ consists of all vectors $\mathbf{w} \in \mathbb{R}^d$, and the rounding method $\phi : \widetilde{\mathcal{C}} \mapsto \mathcal{C}$ converts $\mathbf{w}$ to a permutation π satisfying (33). When optimizing in $\widetilde{\mathcal{C}}$, one conveniently works with a convex relaxation $\widetilde{\mathrm{er}}_{\mathcal{D}} : \widetilde{\mathcal{C}} \to \mathbb{R}^+$ as surrogate for the discrete loss $\mathrm{er}_{\mathcal{D}}$, defined as follows

$$\widetilde{\mathrm{er}}_{\mathcal{D}}(\tilde{h}) = \mathbf{E}_{(X,Y)\sim\mathcal{D}}\left[\tilde{L}\left(\tilde{h}(X), Y\right)\right] \ . \tag{35}$$

where $\tilde{L} : \mathbb{R} \times \{0,1\} \mapsto \mathbb{R}^+$ is some function convex in the first argument, and satisfying

$$\mathbf{1}_{\left(\phi(\tilde{h})\right)(X) \neq Y} \leq c\tilde{L}\left(\tilde{h}(X), Y\right)$$

for all $\tilde{h} \in \mathcal{C}$ and $X \in \mathcal{X}$, where $c > 0$ is some constant. In words, this means that $\tilde{L}$ upper bounds the discrete loss (up to a factor of c). A typical choice for the example in Section 6.1 would be to define for all $\mathbf{w} \in \widetilde{\mathcal{C}}$ and $x = (u, v) \in \mathcal{X}$: $\mathbf{w}(x) = \langle \mathbf{w}, \mathbf{u} - \mathbf{v} \rangle$, and $\tilde{L}(a, b) = \max\{1 - a(2b-1), 0\}$. Using this choice, optimizing over (35) becomes the famous SVMRank with the hinge loss relaxation (Herbrich et al., 2000; Joachims, 2002):

$$\begin{aligned} \text{Minimize } F(\mathbf{w}, \xi) = \quad & \sum_{u,v} \xi_{u,v} \\ \text{s.t., } \forall u, v, Y(u,v) = 1 : \quad & (\mathbf{u} - \mathbf{v}) \cdot \mathbf{w} \geq 1 - \xi_{u,v} \\ \forall u, v : \quad & \xi_{u,v} \geq 0 \,, \\ & \|\mathbf{w}\| \leq c. \end{aligned}$$

(Note: c is a regularization parameter.)

We now have a natural extension of relative regret: $\widetilde{\mathrm{reg}}_{\tilde{h}}(\tilde{h}') = \widetilde{\mathrm{er}}_{\mathcal{D}}(\tilde{h}') - \widetilde{\mathrm{er}}_{\mathcal{D}}(\tilde{h})$. By our assumptions on convexity, $\widetilde{\mathrm{reg}}_{\tilde{h}} : \widetilde{\mathcal{C}} \mapsto \mathbb{R}^+$ can be efficiently optimized. We now say that $f : \tilde{\mathcal{C}} \mapsto \mathbb{R}^+$ is an (ε, μ)-SRRA with respect to $\tilde{h} \in \widetilde{\mathcal{C}}$ if for all $\tilde{h}' \in \widetilde{\mathcal{C}}$,

$$\left|\mathrm{reg}_{\tilde{h}'}(\tilde{h}') - f(\tilde{h}')\right| \leq \varepsilon\left(\mathrm{dist}\big(\phi(\tilde{h}), \phi(\tilde{h}')\big) + \mu\right) \ .$$

If $\mu = 0$ then we simply say that f is an ε-SRRA. The following is an analogue to Corollary 4:

Theorem 24 *Let $\tilde{h}_0, \tilde{h}_1, \tilde{h}_2, \ldots$ be a sequence of hypotheses in $\widetilde{\mathcal{C}}$ such that for all $i \geq 1$, $\tilde{h}_i = \operatorname{argmin}_{\tilde{h}' \in \widetilde{\mathcal{C}}} f_{i-1}(\tilde{h}')$, where f_{i-1} is an (ε, μ)-SRRA with respect to $\tilde{h}_{i-1}$. Then for all $i \geq 1$,*

$$\widetilde{\mathrm{er}}_{\mathcal{D}}(h_i) = (1 + O(\varepsilon))\,\tilde{\nu} + O(\varepsilon^i)\widetilde{\mathrm{er}}_{\mathcal{D}}(h_0) + O(\varepsilon\mu) \ ,$$

where $\tilde{\nu} = \inf_{\tilde{h} \in \widetilde{C}} \widetilde{\mathrm{er}}_{\mathcal{D}}(\tilde{h})$ and the O-notations may hide constants that depend on c.

The proof is very similar to that of Corollary 4, and we omit the details. It turns out that the sampling techniques used for constructing an ε-SRRA from Section 4.3 can be used for constructing an ε-SRRA for the SVMRank relaxed version as well, as long as $\mathcal{C}$ is restricted to bounded vectors $\mathbf{w}$ and all the feature vectors $\mathbf{v}$ corresponding to $v \in V$ are bounded as well. It is easy to confirm that under such bounded-norm setting all arguments of Section 4.3 follows through. The conclusion is that we can solve SVMRank, in polynomial time, to within an error of $(1 + \varepsilon)\tilde{\nu}$ using only $O\big(n \operatorname{poly}(\log n, \varepsilon^{-1})\big)$ preference queries.

6.2.1 Discussion

1. It is worth mentioning the empirically evidence for the success of learning to rank methods that are focused on using relaxed rather than exact constraints on the solution function (see, e.g., Mcfee and Lanckriet, 2010; Long et al., 2010). Another possible empirically viable direction is to make structural assumptions on the preference noise, we refer the reader to the work of Jamieson and Nowak (2011) for a recent result with improved query complexity under certain Bayesian noise assumptions.
2. We note that while disagreement coefficients are brittle, SRRAs are more robust, as demonstrated by the results reported in this paper. Specifically, the application of SRRAs yields efficient solutions to problems on which a disagreement coefficient approach fails, as described in Sections 4.2 and 5.2, and thus we believe SRRAs should be applied more broadly than a stand-alone disagreement coefficient approach.

7. Conclusions and Future Work

In this work we showed that being able to estimate the relative regret function using carefully biased sampling methods can yield query efficient active learning algorithms. We showed that such estimations can be obtained when the only assumptions we make are bounds on the disagreement coefficient and the VC dimension. This leads to active learning algorithms that almost match the best known using the same assumptions. On the other hand, we presented two problems of vast interest (currently in the margin but gradually moving inside the active learning literature), for which a direct analysis of the relative regret function produced better active learning strategies. The two problems we studied are concerned with learning relations over a ground set, where one problem dealt order relations and the other with equivalence relations (with bounded number of equivalence classes). In both problems our query complexity bounds had an undesirable factors of ε^{-3} which we believe should be reduced to ε^{-2} using more advanced measure concentration tools. We leave this to future work. It would also be interesting to identify other problems for which our approach yields active learning algorithms with faster than previously known convergence rates. Immediate candidates are hierarchical clustering and metric learning. Finally, for LRPP, we discussed a practical scenario in which the ground set is endowed with feature

vectors. We showed how to take the underlying geometry into account in our framework. We did not do so for clustering with side information. The work of Eriksson et al. (2011) indicates that incorporating geometric information into our analysis is a fruitful direction to pursue.

Our work made no assumptions on the noise, except maybe for its magnitude. Another promising future research direction would be to incorporate various standard noise assumptions known to improve active learning rates (especially the model of Mammen and Tsybakov, 1999; Tsybakov, 2004) within our setting.

Acknowledgments

We thank Alekh Agarwal, Nina Balcan, Miroslav Dudik, Ran El-Yaniv, Sariel Har-Peled, John Langford, Rob Schapire, Masashi Sugiyama, and Yair Weiner for helpful discussions. Nir Ailon acknowledges the support of a Marie Curie International Reintegration Grant PIRG07-GA-2010-268403. Esther Ezra acknowledges the support of a National Science Foundation Grant CCF-12-16689.

References

Nir Ailon. An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity. *Journal of Machine Learning Research*, 13:137–164, 2012.

Nir Ailon, Bernard Chazelle, Seshadhri Comandur, and Ding Liu. Estimating the distance to a monotone function. *Random Struct. Algorithms*, 31(3):371–383, 2007.

Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *Journal of the ACM*, 55(5):23:1–23:27, October 2008.

Noga Alon. Ranking tournaments. *SIAM Journal on Discrete Mathematics*, 20, 2006.

Francis 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.

Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. In *ICML*, pages 65–72, 2006.

Maria-Florina Balcan, Andrei Z. Broder, and Tong Zhang. Margin based active learning. In *COLT*, pages 35–50, 2007.

Maria-Florina Balcan, Steve Hanneke, and Jennifer Wortman. The true sample complexity of active learning. In *COLT*, pages 45–56, 2008.

Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Approximate clustering without the approximation. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 1068–1077, 2009.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56:89–113, 2004.

Sugato Basu. *Semi-supervised Clustering: Probabilistic Models, Algorithms and Experiments*. PhD thesis, Department of Computer Sciences, University of Texas at Austin, 2005.

Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 2009.

Alina Beygelzimer, Daniel Hsu, John Langford, and Tong Zhang. Agnostic active learning without constraints. In *NIPS*, 2010.

Mark Braverman and Elchanan Mossel. Noisy sorting without resampling. In *SODA*, pages 268–276, 2008.

Rui Castro and Robert Nowak. Minimax bounds for active learning. *IEEE Transactions on Information Theory*, 54(5):2339–2353, 2008.

Rui Castro, Rebecca Willett, and Robert Nowak. Faster rates in regression via active learning. In *NIPS*, 2005.

Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Linear classification and selective sampling under low noise conditions. In *NIPS*, pages 249–256, 2008.

Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Learning noisy linear classifiers via adaptive and selective sampling. *Machine Learning*, 83(1):71–102, 2011.

Nicolò Cesa-Bianchi, Claudio Gentile, Fabio Vitale, and Giovanni Zappella. Active learning on trees and graphs. In *COLT*, pages 320–332, 2010.

Nicolò Cesa-Bianchi, Claudio Gentile, Fabio Vitale, and Giovanni Zappella. A correlation clustering approach to link classification in signed networks. In *Proceedings of the 25th Annual Conference on Learning Theory. JMLR Workshop and Conference Proceedings*, volume 23 of *JMLR Workshop and Conference Proceedings*, pages 34.1–34.20, 2012.

Moses Charikar and Anthony Wirth. Maximizing quadratic programs: Extending grothendieck's inequality. In *FOCS*, pages 54–60. IEEE Computer Society, 2004.

David Cohn, Rich Caruana, and Andrew Mccallum. Semi-supervised clustering with user feedback. unpublished manuscript, 2000. URL `http://www.cs.umass.edu/~mccallum/papers/semisup-aaai2000s.ps`.

Don Coppersmith, Lisa K. Fleischer, and Atri Rurda. Ordering by weighted number of wins gives a good ranking for weighted tournaments. *ACM Trans. Algorithms*, 6:55:1–55:13, July 2010.

Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In *NIPS*, 2005.

Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *ICML*, pages 208–215, 2008.

Sanjoy Dasgupta, Daniel Hsu, and Claire Monteleoni. A general agnostic active learning algorithm. In *NIPS*, 2007.

Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 3rd edition, 2008.

Ayhan Demiriz, Kristin Bennett, and Mark J. Embrechts. Semi-supervised clustering using genetic algorithms. In *In Artificial Neural Networks in Engineering (ANNIE-99*, pages 809–814. ASME Press, 1999.

Persi Diaconis and R. L. Graham. Spearman's Footrule as a measure of disarray. *Journal of the Royal Statistical Society*, 39(2):262–268, 1977.

Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11:1605–1641, 2010.

Brian Eriksson, Gautam Dasarathy, Aarti Singh, and Robert D. Nowak. Active clustering: Robust and efficient hierarchical clustering using adaptively selected similarities. *Journal of Machine Learning Research - Proceedings Track*, 15:260–268, 2011.

Esther Ezra and Shai Fine. On the cover of convex polyhedra in d-space. Unpublished manuscript, 2007.

Yoav Freund, Sebastian H. Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, September 1997.

Ioannis Giotis and Venkatesan Guruswami. Correlation clustering with a fixed number of clusters. *Theory of Computing*, 2(1):249–266, 2006.

Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, 1988.

Shirley Halevy and Eyal Kushilevitz. Distribution-free property-testing. *SIAM J. Comput.*, 37(4):1107–1138, 2007.

Steve Hanneke. A bound on the label complexity of agnostic active learning. In *ICML*, 2007.

Steve Hanneke. Adaptive rates of convergence in active learning. In *COLT*, 2009.

Steve Hanneke. Rates of convergence in active learning. *Annals of Statistics*, 39(1):333–361, 2011.

Steve Hanneke and Liu Yang. Negative results for active learning with convex losses. *Journal of Machine Learning Research - Proceedings Track*, 9:321–325, 2010.

David Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Control*, 100(1):78–150, September 1992.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin ranking boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, chapter 7, pages 115–132. The MIT Press, 2000.

Kevin G. Jamieson and Rob Nowak. Active ranking using pairwise comparisons. In *NIPS 24*, pages 2240–2248, 2011.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD*, 2002.

Matti Kääriäinen. Active learning in the non-realizable case. In *ALT*, pages 63–77, 2006.

Narendra Karmarkar. A new polynomial–time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 95–103, 2007.

Leonid Khačiyan. Polynomial algorithm for linear programming. *Soviet Doklady*, 244: 1093–1096, 1979.

Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *ICML*, pages 307–314, 2002.

Vladimir Koltchinskii. Rademacher complexities and bounding the excess risk in active learning. *Journal of Machine Learning Research*, 11:2457–2485, 2010.

Yi Li, Philip M. Long, and Aravind Srinivasan. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences*, 62:2001, 2000.

Bo Long, Olivier Chapelle, Ya Zhang, Yi Chang, Zhaohui Zheng, and Belle Tseng. Active learning for ranking through expected loss optimization. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 267–274, 2010.

Enno Mammen and Alexandre B. Tsybakov. Smooth discrimination analysis. *Annals of Statistics*, 27:1808–1829, 1999.

Brian Mcfee and Gert Lanckriet. Metric learning to rank. In *In Proceedings of the 27th Annual International Conference on Machine Learning (ICML)*, 2010.

Stanislav Minsker. Plug-in approach to active learning. *Journal of Machine Learning Research*, 13:67–90, 2012.

Francesco Orabona and Nicolò Cesa-Bianchi. Better algorithms for selective sampling. In *ICML*, pages 433–440, 2011.

Kira Radinsky and Nir Ailon. Ranking from pairs and triplets: information quality, evaluation methods and query complexity. In *WSDM*, pages 105–114, 2011.

Burr Settles. Active learning literature survey. Technical Report 1648, University of Wisconsin–Madison, 2009.

Ohad Shamir and Naftali Tishby. Spectral clustering on a budget. *Journal of Machine Learning Research - Proceedings Track*, 15:661–669, 2011.

Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discrete Applied Math*, 144:173–182, nov 2004.

Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, 1995.

Masashi Sugiyama. Active learning in approximately linear regression based on conditional expectation of generalization error. *Journal of Machine Learning Research*, 7:141–166, 2006.

Alexandre B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, 32:135–166, 2004.

Konstantin Voevodski, Maria-Florina Balcan, Heiko Röglin, Shang-Hua Teng, and Yu Xia. Active clustering of biological sequences. *Journal of Machine Learning Research*, 13: 203–225, 2012.

Liwei Wang. Smoothness, disagreement coefficient, and the label complexity of agnostic active learning. *Journal of Machine Learning Research*, 12:2269–2292, 2011.

Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, 2002.

Liu Yang, Steve Hanneke, and Jaime G. Carbonell. Bayesian active learning using arbitrary binary valued queries. In *ALT*, pages 50–58, 2010.

Liu Yang, Steve Hanneke, and Jaime G. Carbonell. The sample complexity of self-verifying bayesian active learning. *Journal of Machine Learning Research - Proceedings Track*, 15: 816–822, 2011.

Journal of Machine Learning Research 15 (2014) 921-947 Submitted 11/08, revised 9/13; Published 3/14

An Extension of Slow Feature Analysis for Nonlinear Blind Source Separation

Henning Sprekeler* H.SPREKELER@ENG.CAM.AC.UK
Tiziano Zito TIZIANO.ZITO@BCCN-BERLIN.DE
Laurenz Wiskott† LAURENZ.WISKOTT@INI.RUB.DE

Institute for Theoretical Biology and Bernstein Center for Computational Neuroscience Berlin
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin, Germany

Editor: Aapo Hyvärinen

Abstract

We present and test an extension of slow feature analysis as a novel approach to nonlinear blind source separation. The algorithm relies on temporal correlations and iteratively reconstructs a set of statistically independent sources from arbitrary nonlinear instantaneous mixtures. Simulations show that it is able to invert a complicated nonlinear mixture of two audio signals with a high reliability. The algorithm is based on a mathematical analysis of slow feature analysis for the case of input data that are generated from statistically independent sources.

Keywords: slow feature analysis, nonlinear blind source separation, statistical independence, independent component analysis, slowness principle

1. Introduction

Independent Component Analysis (ICA) as a technique for blind source separation (BSS) has attracted a fair amount of research activity over the past three decades. By now a number of techniques have been established that reliably reconstruct the underlying sources from linear mixtures (Hyvärinen et al., 2001). The key insight for linear BSS is that the statistical independence of the sources is usually sufficient to constrain the unmixing function up to trivial transformations like permutation and scaling. Therefore, linear BSS is essentially equivalent to linear ICA.

An obvious extension of the linear case is the task of reconstructing the sources from nonlinear mixtures. Unfortunately, the problem of nonlinear BSS is much harder than linear BSS, because the statistical independence of the instantaneous values of the estimated sources is no longer a sufficient constraint for the unmixing (Hyvärinen and Pajunen, 1999). For example, arbitrary point-nonlinear distortions of the sources are still statistically independent. Additional constraints are needed to resolve these ambiguities.

∗. H.S. is now also at the Computational and Biological Learning Laboratory, Department of Engineering, University of Cambridge, UK.
†. L.W. is now at the Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany.

One approach is to exploit the temporal structure of the sources (e.g., Harmeling et al., 2003; Blaschke et al., 2007). Blaschke et al. (2007) have proposed to use the tendency of nonlinearly distorted versions of the sources to vary more quickly in time than the original sources. A simple illustration of this effect is the frequency doubling property of a quadratic nonlinearity when applied to a sine wave. This observation opens the possibility of finding the original source (or a good representative thereof) among all the nonlinearly distorted versions by choosing the one that varies most slowly in time. An algorithm that has been specifically designed for extracting slowly varying signals is Slow Feature Analysis (SFA, Wiskott, 1998; Wiskott and Sejnowski, 2002). SFA is intimately related to ICA techniques like TDSEP (Ziehe and Müller, 1998; Blaschke et al., 2006) and differential decorrelation (Choi, 2006) and is therefore an interesting starting point for developing nonlinear BSS techniques.

Here, we extend a previously developed mathematical analysis of SFA (Franzius et al., 2007) to the case where the input data are generated from a set of statistically independent sources. The theory makes predictions as to how the sources are represented by the output signals of SFA, based on which we develop a new algorithm for nonlinear blind source separation. Because the algorithm is an extension of SFA, we refer to it as xSFA.

The structure of the paper is as follows. In Section 2, we introduce the optimization problem for SFA and give a brief sketch of the SFA algorithm. In Section 3, we develop the theory that underlies the xSFA algorithm. In Section 4, we present the xSFA algorithm and evaluate its performance. Limitations and possible reasons for failures are discussed in section 5. Section 6 discusses the relation of xSFA to other nonlinear BSS algorithms. We conclude with a general discussion in Section 7.

2. Slow Feature Analysis

In this section, we briefly present the optimization problem that underlies slow feature analysis and sketch the algorithm that solves it.

2.1 The Optimization Problem

Slow Feature Analysis is based on the following optimization task: For a given multi-dimensional input signal we want to find a set of scalar functions that generate output signals that vary as slowly as possible. To ensure that these signals carry significant information about the input, we require them to be uncorrelated and have zero mean and unit variance. Mathematically, this can be stated as follows:

Optimization problem 1: *Given a function space $\mathcal{F}$ and an N-dimensional input signal $\mathbf{x}(t)$, find a set of J real-valued input-output functions $g_j(\mathbf{x}) \in \mathcal{F}$ such that the output signals $y_j(t) := g_j(\mathbf{x}(t))$ minimize*

$$\Delta(y_j) = \langle \dot{y}_j^2 \rangle_t \tag{1}$$

under the constraints

$$\begin{aligned} \langle y_j \rangle_t &= 0 \quad \textit{(zero mean)}, & (2)\\ \langle y_j^2 \rangle_t &= 1 \quad \textit{(unit variance)}, & (3)\\ \forall i < j : \langle y_i y_j \rangle_t &= 0 \quad \textit{(decorrelation and order)}, & (4) \end{aligned}$$

with $\langle\cdot\rangle_t$ and $\dot{y}$ indicating temporal averaging and the derivative of y, respectively.

Equation (1) introduces the Δ-value, which is small for slowly varying signals $y(t)$. The constraints (2) and (3) avoid the trivial constant solution. The decorrelation constraint (4) forces different functions g_j to encode different aspects of the input. Note that the decorrelation constraint is asymmetric: The function g_1 is the slowest function in $\mathcal{F}$, while the function g_2 is the slowest function that fulfills the constraint of generating a signal that is uncorrelated to the output signal of g_1. The resulting sequence of functions is therefore ordered according to the slowness of their output signals on the training data.

It is important to note that although the objective is the slowness of the output signal, the functions g_j are instantaneous functions of the input, so that slowness cannot be achieved by low-pass filtering. As a side effect, SFA is not suitable for inverting convolutive mixtures.

2.2 The SFA Algorithm

If $\mathcal{F}$ is finite-dimensional, the problem can be solved efficiently by the SFA algorithm (Wiskott and Sejnowski, 2002; Berkes and Wiskott, 2005). The full algorithm can be split in two parts: a nonlinear expansion of the input data, followed by a linear generalized eigenvalue problem.

For the nonlinear expansion, we choose a set of functions $f_i(\mathbf{x})$ that form a basis of the function space $\mathcal{F}$. The optimal functions g_j can then be expressed as linear combinations of these basis functions: $g_j(\mathbf{x}) = \sum_i W_{ji} f_i(\mathbf{x})$. By applying the basis functions to the input data $\mathbf{x}(t)$, we get a new and generally high-dimensional set of signals $z_i(t) = f_i(\mathbf{x}(t))$. Without loss of generality, we assume that the functions f_i are chosen such that the expanded signals z_i have zero mean on the input data $\mathbf{x}$. Otherwise, this can be achieved easily by subtracting the mean.

After the nonlinear expansion, the coefficients W_{ji} for the optimal functions can be found from a generalized eigenvalue problem:

$$\dot{\mathbf{C}}\mathbf{W} = \mathbf{C}\mathbf{W}\Lambda\,. \tag{5}$$

Here, $\dot{\mathbf{C}}$ is the matrix of the second moments of the temporal derivative $\dot{z}_i$ of the expanded signals: $\dot{C}_{ij} = \langle \dot{z}_i(t)\dot{z}_j(t)\rangle_t$. $\mathbf{C}$ is the covariance matrix $\mathbf{C} = \langle z_i(t) z_j(t)\rangle_t$ of the expanded signals (since $\mathbf{z}$ has zero mean), $\mathbf{W}$ is a matrix that contains the weights W_{ji} for the optimal functions and Λ is a diagonal matrix that contains the generalized eigenvalues on the diagonal.

If the function space $\mathcal{F}$ is the set of linear functions, the algorithm reduces to solving the generalized eigenvalue problem (5) without nonlinear expansion. Therefore, the second step of the algorithm is in the following referred to as linear SFA.

3. Theoretical Foundations

In this section we extend previous analytical results for SFA to the case of nonlinear blind source separation, more precisely, to the case where the input data $\mathbf{x}(t)$ are generated from a set of statistically independent sources $\mathbf{s}(t)$ by means of a nonlinear, instantaneous, and invertible (or at least injective) function: $\mathbf{x}(t) = \mathbf{F}(\mathbf{s}(t))$. For readers that are more interested in the algorithm than in its mathematical foundations, a summary of the relevant theoretical results can be found at the end of the section.

3.1 SFA With Unrestricted Function Spaces

The central assumption of the theory is that the function space $\mathcal{F}$ that SFA can access is unrestricted apart from the necessary mathematical requirements of integrability and differentiability.[1] This has important conceptual consequences.

3.1.1 Conceptual Consequences of an Unrestricted Function Space

Let us for the moment assume that the mixture $\mathbf{x} = \mathbf{F}(\mathbf{s})$ has the same dimensionality as the source vector. Let g be an arbitrary function $g \in \mathcal{F}$, which generates an output signal $y(t) = g(\mathbf{x}(t))$ when applied to the mixture $\mathbf{x}(t)$. Then, for every such function g, there is another function $\tilde{g} = g \circ \mathbf{F}$ that generates the same output signal $y(t)$ when applied to the sources $\mathbf{s}(t)$ directly. Because the function space $\mathcal{F}$ is unrestricted, this function $\tilde{g}$ is also an element of the function space $\mathcal{F}$. Because this is true for all functions $g \in \mathcal{F}$, the set of output signals that can be generated by applying the functions in $\mathcal{F}$ to the mixture $\mathbf{x}(t)$ is the same as the set of output signals that can be generated by applying the functions to the sources $\mathbf{s}(t)$ directly. Because the optimization problem of SFA is formulated purely in terms of output signals, the output signals when applying SFA to the mixture are the same as when applied directly to the sources. In other words: For an unrestricted function space, the output signals of SFA are independent of the structure of the mixing function $\mathbf{F}$. This statement can be generalized to the case where the mixture $\mathbf{x}$ has a higher dimensionality than the sources, as long as the mixing function $\mathbf{F}$ is injective.

Given that the output signals are independent of the mixture, we can make analytical predictions about the dependence of the output signals on the sources, when the input signals are not a mixture, but the sources themselves. These predictions generalize to the case where the input signals are a nonlinear mixture of the sources instead.

Of course, an unrestricted function space cannot be implemented in practice. Therefore, in any application the output signals depend on the mixture and on the function space used. Nevertheless, the idealized case provides important theoretical insights, which we use as the basis for the blind source separation algorithm presented later.

3.1.2 Earlier Results for SFA with an Unrestricted Function Space

In a previous article (Franzius et al., 2007, Theorems 1-5), we have shown that the optimal functions $g_j(\mathbf{x})$ for SFA in the case of an unrestricted function space are given by the solutions of an eigenvalue equation for a partial differential operator $\mathcal{D}$

$$\mathcal{D} g_j(\mathbf{x}) = \lambda_j g_j(\mathbf{x})$$

with von Neumann boundary conditions

$$\sum_{\alpha\beta} n_\alpha p_{\mathbf{x}}(\mathbf{x}) K_{\alpha\beta}(\mathbf{x}) \partial_\beta g_j(\mathbf{x}) = 0\,.$$

1. More precisely, we assume that the function space $\mathcal{F}$ is the Sobolev space of functions for which both the functions themselves as well as all their partial derivatives with respect to the input signals are square-integrable with respect to the probability measure of the input signals.

Here, $\mathcal{D}$ denotes the operator

$$\mathcal{D} = -\frac{1}{p_{\mathbf{x}}(\mathbf{x})} \sum_{\alpha,\beta} \partial_\alpha p_{\mathbf{x}}(\mathbf{x}) K_{\alpha\beta}(\mathbf{x}) \partial_\beta \,, \tag{6}$$

$p_{\mathbf{x}}(\mathbf{x})$ is the probability density of the input data $\mathbf{x}$ (which we assumed to be non-zero within the range of $\mathbf{x}$) and ∂_α the partial derivative with respect to the α-th component x_α of the input data. $K_{\alpha\beta}(\mathbf{x}) = \langle \dot{x}_\alpha \dot{x}_\beta \rangle_{\dot{\mathbf{x}}|\mathbf{x}}$ is the matrix of the second moments of the velocity distribution $p(\dot{\mathbf{x}}|\mathbf{x})$ of the input data, conditioned on their value $\mathbf{x}$ and $n_\alpha(\mathbf{x})$ is the α-th component of the normal vector on the boundary point $\mathbf{x}$. Note that the partial derivative ∂_α acts on all terms to its right, so that $\mathcal{D}$ is a partial differential operator of second order. The optimal functions for SFA are the J eigenfunctions g_j with the smallest eigenvalues λ_j.

3.2 Factorization Of The Optimal Functions

As discussed above, the dependence of the output signals on the sources can be studied by using the sources themselves as input data. However, because the sources are assumed to be statistically independent, we have additional knowledge about their probability distribution and consequently also about the matrix $K_{\alpha\beta}$. The joint probability density for the sources and their derivatives factorizes:

$$p_{\mathbf{s},\dot{\mathbf{s}}}(\mathbf{s}, \dot{\mathbf{s}}) = \prod_\alpha p_{s_\alpha, \dot{s}_\alpha}(s_\alpha, \dot{s}_\alpha) \,.$$

Clearly, the marginal probability density $p_{\mathbf{s}}$ also factorizes into the individual probability densities $p_\alpha(s_\alpha)$

$$p_{\mathbf{s}}(\mathbf{s}) = \prod_\alpha p_\alpha(s_\alpha) \,, \tag{7}$$

and the matrix $K_{\alpha\beta}$ of the second moments of the velocity distribution of the sources is diagonal

$$K_{\alpha\beta}(\mathbf{s}) := \langle \dot{s}_\alpha \dot{s}_\beta \rangle_{\dot{\mathbf{s}}|\mathbf{s}} = \delta_{\alpha\beta} K_\alpha(s_\alpha) \quad \text{with} \quad K_\alpha(s_\alpha) := \langle \dot{s}_\alpha^2 \rangle_{\dot{s}_\alpha|s_\alpha} \,. \tag{8}$$

The latter is true, because the mean temporal derivative of 1-dimensional stationary and continuously differentiable stochastic processes vanishes for any s_α for continuity reasons (for a mathematical argument see Appendix), so that $K_{\alpha\beta}$ is not only the matrix of the second moments of the derivatives, but actually the conditional covariance matrix of the derivatives of the sources given the sources. As the sources are statistically independent, their derivatives are uncorrelated and $K_{\alpha\beta}$ has to be diagonal.

We can now insert the specific form (7,8) of the probability distribution $p_{\mathbf{s}}$ and the matrix $K_{\alpha\beta}$ into the definition (6) of the operator $\mathcal{D}$. A brief calculation shows that this leads to a separation of the operator $\mathcal{D}$ into a sum of operators $\mathcal{D}_\alpha$, each of which depends on only one of the sources:

$$\mathcal{D}(\mathbf{s}) = \sum_\alpha \mathcal{D}_\alpha(s_\alpha)$$

with

$$\mathcal{D}_\alpha = -\frac{1}{p_\alpha} \partial_\alpha p_\alpha K_\alpha \partial_\alpha \,. \tag{9}$$

This has the important implication that the solution to the full eigenvalue problem for $\mathcal{D}$ can be constructed from the 1-dimensional eigenvalue problems for the individual sources:

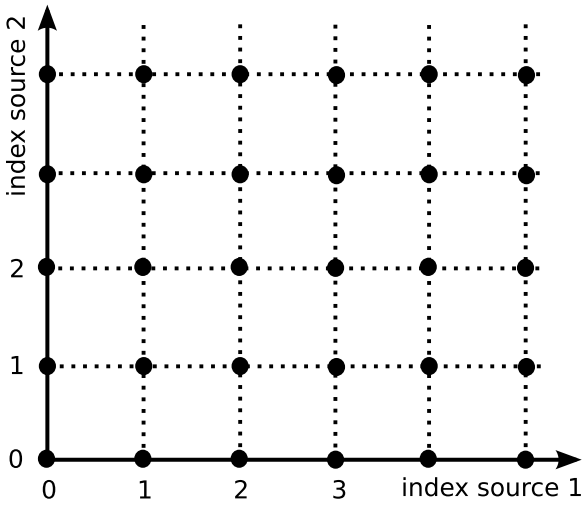


Figure 1: Schematic ordering of the optimal functions for SFA. For an unrestricted function space and statistically independent sources, the optimal functions for SFA are products of harmonics, each of which depends on one of the sources only. In the case of two sources, the optimal functions can therefore be arranged schematically on a 2-dimensional grid, where every grid point represents one function and its coordinates in the grid are the indices of the harmonics that are multiplied to form the function. Because the 0-th harmonic is the constant, the functions on the axes are simply the harmonics themselves and therefore depend on one of the sources only. Moreover, the grid points (1,0) and (0,1) are monotonic functions of the sources and therefore a good representative thereof. It is these solutions that the xSFA algorithm is designed to extract. Note that the scheme also contains an ordering by slowness: All functions to the upper right of a given function have higher Δ-values and therefore vary more quickly.

Theorem 1 *Let $g_{\alpha i}$ ($i \in \mathbb{N}$) be the normalized eigenfunctions of the operators $\mathcal{D}_\alpha$, that is, the set of functions $g_{\alpha i}$ that fulfill the eigenvalue equations*

$$\mathcal{D}_\alpha g_{\alpha i} = \lambda_{\alpha i} g_{\alpha i} \tag{10}$$

with the boundary conditions

$$p_\alpha K_\alpha \partial_\alpha g_{\alpha i} = 0 \tag{11}$$

and the normalization condition

$$(g_{\alpha i}, g_{\alpha i})_\alpha := \langle g_{\alpha i}^2 \rangle_{s_\alpha} = 1 \,.$$

Then, the product functions

$$g_{\mathbf{i}}(\mathbf{s}) := \prod_\alpha g_{\alpha i_\alpha}(s_\alpha)$$

form a set of (normalized) eigenfunctions to the full operator $\mathcal{D}$ with the eigenvalues

$$\lambda_{\mathbf{i}} = \sum_\alpha \lambda_{\alpha i_\alpha}$$

and thus those $g_{\mathbf{i}}$ with the smallest eigenvalues $\lambda_{\mathbf{i}}$ are the optimal functions for SFA. Here, $\mathbf{i} = (i_1, ..., i_S) \in \mathbb{N}^S$ denotes a multi-index that enumerates the eigenfunctions of the full eigenvalue problem.

In the following, we assume that the eigenfunctions $g_{\alpha i}$ are ordered by their eigenvalue and refer to them as the *harmonics* of the source s_α. This is motivated by the observation that in the case where p_α and K_α are independent of s_α, that is, for a uniform distribution, the eigenfunctions $g_{\alpha i}$ are harmonic oscillations whose frequency increases linearly with i (see below). Moreover, we assume that the sources s_α are ordered according to slowness, in this case measured by the eigenvalue $\lambda_{\alpha 1}$ of their lowest non-constant harmonic $g_{\alpha 1}$. These eigenvalues are the Δ-values of the slowest possible nonlinear point transformations of the sources.

The key result of theorem 1 is that in the case of statistically independent sources, the output signals are products of harmonics of the sources. Note that the constant function $g_{\alpha 0}(s_\alpha) = 1$ is an eigenfunction with eigenvalue 0 to all the eigenvalue problems (10). As a consequence, the harmonics $g_{\alpha i}$ of the single sources are also eigenfunctions to the full operator $\mathcal{D}$ (with the index $\mathbf{i} = (0, ..., 0, i_\alpha = i, 0, ..., 0)$) and can thus be found by SFA. Importantly, the lowest non-constant harmonic of the slowest source (i.e., $g_{(1,0,0,...)} = g_{11}$) is the function with the smallest overall Δ-value (apart from the constant) and thus the first function found by SFA. In the next sections, we show that the lowest non-constant harmonics $g_{\alpha 1}$ reconstruct the sources up to a monotonic and thus invertible point transformation and that in the case of sources with Gaussian statistics they even reproduce the sources exactly.

3.3 The First Harmonic Is A Monotonic Function Of The Source

The eigenvalue problem (10,11) has the form of a Sturm-Liouville problem (Courant and Hilbert, 1989) and can easily be rewritten to have the standard form for these problems:

$$\partial_\alpha p_\alpha K_\alpha \, \partial_\alpha g_{\alpha i} + \lambda_{\alpha i} p_\alpha g_{\alpha i} \overset{(10,9)}{=} 0 \,, \tag{12}$$

$$\text{with} \quad p_\alpha K_\alpha \partial_\alpha g_{\alpha i} \overset{(11)}{=} 0 \quad \text{for} \quad s_\alpha \in \{a, b\} \,. \tag{13}$$

Here, we assume that the source s_α is bounded and takes on values on the interval $s_\alpha \in [a, b]$. Note that both p_α and $p_\alpha K_\alpha$ are positive for all s_α. Sturm-Liouville theory states that (i) all eigenvalues are positive (Courant and Hilbert, 1989), (ii) the solutions $g_{\alpha i}, i \in \mathbb{N}^0$ of this problem are oscillatory and (iii) $g_{\alpha i}$ has exactly i zeros on $]a, b[$ if the $g_{\alpha i}$ are ordered by increasing eigenvalue $\lambda_{\alpha i}$ (Courant and Hilbert, 1989, Chapter VI, §6). In particular, $g_{\alpha 1}$ has only one zero $\xi \in]a, b[$. Without loss of generality we assume that $g_{\alpha 1} < 0$ for $s_\alpha < \xi$ and $g_{\alpha 1} > 0$ for $s_\alpha > \xi$. Then Equation (12) implies that

$$\begin{aligned}
& \partial_\alpha p_\alpha K_\alpha \partial_\alpha g_{\alpha 1} = -\lambda_\alpha p_\alpha g_{\alpha 1} < 0 \quad \text{for } s_\alpha > \xi \\
\Longrightarrow \quad & p_\alpha K_\alpha \partial_\alpha g_{\alpha 1} \quad \text{is monotonically decreasing on }]\xi, b] \\
\overset{(13)}{\Longrightarrow} \quad & p_\alpha K_\alpha \, \partial_\alpha g_{\alpha 1} > 0 \text{ on }]\xi, b[\\
\Longrightarrow \quad & \partial_\alpha g_{\alpha 1} > 0 \text{ on }]\xi, b[\,, \text{ because } p_\alpha K_\alpha > 0 \\
\Longleftrightarrow \quad & g_{\alpha 1} \quad \text{is monotonically increasing on }]\xi, b[\,.
\end{aligned}$$

A similar consideration for $s < \xi$ shows that $g_{\alpha 1}$ is also monotonically increasing on $]a, \xi[$. Thus, $g_{\alpha 1}$ is monotonic and invertible on the whole interval $[a, b]$. Note that the monotony of $g_{\alpha 1}$ is important in the context of blind source separation, because it ensures that not only some of the output signals of SFA depend on only one of the sources (the harmonics), but that there should actually be some (the lowest non-constant harmonics) that are very similar to the source itself.

3.4 Gaussian Sources

We now consider the situation that the sources are reversible Gaussian stochastic processes, (i.e., that the joint probability density of $s(t)$ and $s(t+\mathrm{d}t)$ is Gaussian and symmetric with respect to $s(t)$ and $s(t+\mathrm{d}t)$). In this case, the instantaneous values of the sources and their temporal derivatives are statistically independent, that is, $p_{\dot{s}_\alpha|s_\alpha}(\dot{s}_\alpha|s_\alpha) = p_{\dot{s}_\alpha}(\dot{s}_\alpha)$. Thus, K_α is independent of s_α, that is, $K_\alpha(s_\alpha) = K_\alpha = \text{const}$. Without loss of generality we assume that the sources have unit variance. Then the probability density of the source is given by

$$p_\alpha(s_\alpha) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-s_\alpha^2/2}$$

and the eigenvalue Equations (12) for the harmonics can be written as

$$\partial_\alpha e^{-s_\alpha^2/2} \partial_\alpha g_{\alpha i} + \frac{\lambda_{\alpha i}}{K_\alpha} e^{-s_\alpha^2/2} g_{\alpha i} = 0\,.$$

This is a standard form of Hermite's differential equation (see Courant and Hilbert, 1989, Chapter V, § 10). Accordingly, the harmonics $g_{\alpha i}$ are given by the (appropriately normalized) Hermite polynomials H_i of the sources:

$$g_{\alpha i}(s_\alpha) = \frac{1}{\sqrt{2^i i!}} H_i\left(\frac{s_\alpha}{\sqrt{2}}\right)\,.$$

The Hermite polynomials can be expressed in terms of derivatives of the Gaussian distribution:

$$H_n(x) = (-1)^n \mathrm{e}^{x^2} \partial_x^n \mathrm{e}^{-x^2}\,.$$

It is clear that Hermite polynomials fulfill the boundary condition

$$\lim_{s_\alpha \to \infty} K_\alpha p_\alpha \partial_\alpha g_{\alpha i} = 0\,,$$

because the derivative of a polynomial is again a polynomial and the Gaussian distribution decays faster than polynomially as $|s_\alpha| \to \infty$. The eigenvalues depend linearly on the index i:

$$\lambda_{\alpha i} = i K_\alpha\,. \tag{14}$$

The most important consequence is that the lowest non-constant harmonics simply reproduce the sources: $g_{\alpha 1}(s_\alpha) = 1/\sqrt{2} H_1(s_\alpha/\sqrt{2}) = s_\alpha$. Thus, for Gaussian sources, some of the output signals of SFA with an unrestricted function space reproduce the sources exactly.

3.5 Uniformly Distributed Sources

Another canonical example for which the eigenvalue Equation (10) can be solved analytically is the case of uniformly distributed sources, that is, the case where the probability distribution $p_{s,\dot{s}}$ is independent of s on a finite interval and zero elsewhere. Consequently, neither $p_\alpha(s_\alpha)$ nor $K_\alpha(s_\alpha)$ can depend on s_α, that is, they are constants. Note that such a distribution may be difficult to implement by a real differentiable process, because the velocity distribution should be different at boundaries that cannot be crossed. Nevertheless, this case provides an approximation to cases, where the distribution is close to homogeneous.

Let s_α take values in the interval $[0, L_\alpha]$. The eigenvalue Equation (12) for the harmonics is then given by

$$K_\alpha \partial_\alpha^2 g_{\alpha i} + \lambda_{\alpha i} g_{\alpha i} = 0$$

and readily solved by harmonic oscillations:

$$g_{\alpha i}(s_\alpha) = \sqrt{2} \cos\left(i\pi \frac{s_\alpha}{L_\alpha} \right) .$$

The Δ-value of these functions is given by

$$\Delta(g_{\alpha i}) = \lambda_{\alpha i} = K_\alpha \left(\frac{\pi}{L_\alpha} i \right)^2 .$$

Note the similarity of these solutions with the optimal free responses derived by Wiskott (2003b).

3.6 Summary: Results Of The Theory

The following key results of the theory form the basis of the xSFA algorithm:

- For an unrestricted function space, the output signals generated by the optimal functions of SFA are independent of the nonlinear mixture, given the same original sources.
- The optimal functions of SFA are products of functions $g_{\alpha i}(s_\alpha)$, each of which depends on only one of the sources. We refer to the function $g_{\alpha i}$ as the i-th harmonic of the source s_α.
- The slowest non-constant harmonic is a monotonic function of the associated source. It can therefore be considered a good representative of the source.
- If the sources have stationary Gaussian statistics, the harmonics are Hermite polynomials of the sources. In particular, the lowest harmonic is then simply the source itself.
- The slowest function found by SFA is the lowest harmonic of the slowest source and therefore a good representative thereof.

4. An Algorithm For Nonlinear Blind Source Separation

According to the theory, some of the output signals of SFA should be very similar to the sources. Therefore, the problem of nonlinear BSS can be reduced to selecting those output signals of SFA that correspond to the first non-constant harmonics of the sources. In this section, we propose and test an algorithm that should ideally solve this problem. In the following, we sometimes refer to the first non-constant harmonics simply as the "sources", because they should ideally be very similar.

4.1 The xSFA Algorithm

The extraction of the slowest source is rather simple: According to the theory, it is well represented by the first (i.e., slowest) output signal of SFA. Unfortunately, extracting the second source is more complicated, because higher order harmonics of the first source may vary more slowly that the second source.

The idea behind the algorithm we propose here is that once we know the first source, we also know all its possible nonlinear transformations, that is, its harmonics. We can thus remove all aspects of the first source from the SFA output signals by projecting the latter to the space that is uncorrelated to all nonlinear versions of the first source. In the grid arrangement shown in Figure 1, this corresponds to removing all solutions that lie on one of the axes. The remaining signals must have a dependence on the second or even faster sources. The slowest possible signal in this space is then generated by the first harmonic of the second source, which we can therefore extract by means of linear SFA. Once we know the first two sources, we can proceed by calculating all the harmonics of the second source and all products of the harmonics of the first and the second source and remove those signals from the data. The slowest signal that remains then is the first harmonic of the third source. Iterating this scheme should in principle yield all the sources.

The structure of the algorithm is the following (see also Figure 2):

1. Start with the first source: $i = 1$.
2. Apply a polynomial expansion of degree N^{SFA} to the mixture to obtain the expanded mixture $\mathbf{z}$.
3. Apply linear SFA to the expanded mixture $\mathbf{z}$ and store the slowest output signal as an estimate $\tilde{s}_i$ of source i.
4. Stop if the desired number of sources has been extracted ($i = S$).
5. Apply a polynomial expansion of degree N^{nl} to the estimated sources $\tilde{s}_{1,\ldots,i}$ and whiten the resulting signals. We refer to the resulting nonlinear versions of the first sources as $n_k,\ k \in \{1, ..., N^{\mathrm{exp}}\}$, where N^{exp} denotes the dimension of a polynomial expansion of degree N^{nl} of i signals.
6. Remove the nonlinear versions of the first i sources from the expanded mixture $\mathbf{z}$

$$z_j(t) \leftarrow z_j(t) - \sum_{k=1}^{N^{\mathrm{exp}}} \mathrm{cov}(z_j, n_k)\, n_k(t)$$

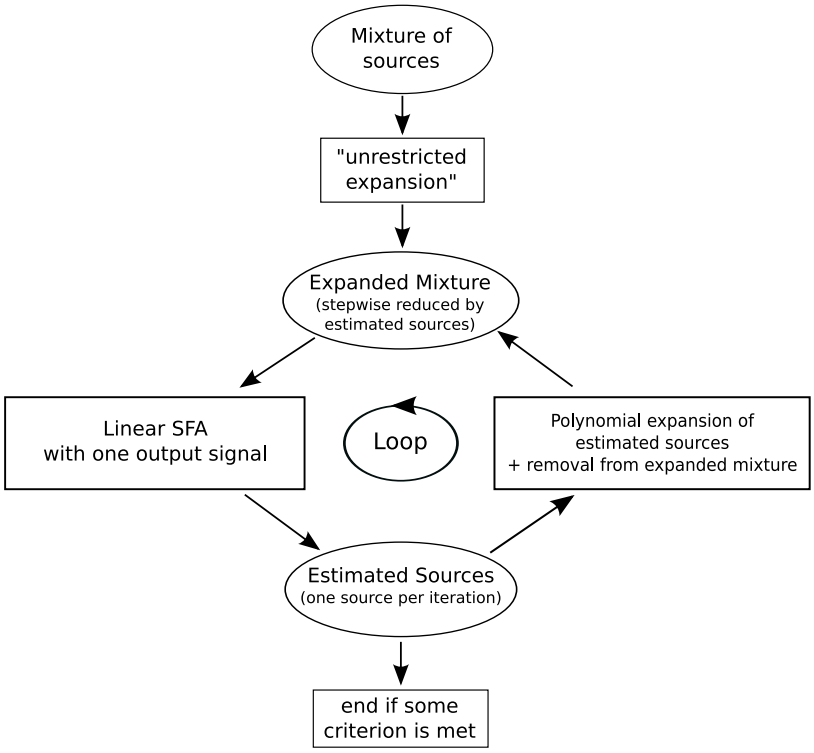


Figure 2: Illustration of the xSFA algorithm. The mixture of the input signals is first subjected to a nonlinear expansion that should be chosen sufficiently powerful to allow (a good approximation of) the inversion of the mixture. An estimate of the first source is then obtained by applying linear SFA to the expanded data. The remaining sources are estimated iteratively by removing nonlinear versions of the previously estimated sources from the expanded data and reapplying SFA. If the number of sources is known, the algorithm terminates when estimates of all sources have been extracted. If the number of sources is unknown, other termination criteria might be more suitable (not investigated here).

and remove principal components with a variance below a given threshold ϵ.

7. To extract the next source, increase i by one and go to step 2, using the new expanded signals $\mathbf{z}$.

Note that the algorithm is a mere extension of SFA in that it does not include new objectives or constraints. We therefore term it xSFA for *eXtended SFA*.

4.2 Simulations

We test the algorithm on two different tasks. The first one is the separation of two audio signals that are subject to a rather complicated mixture. In the second task, we test if the algorithm is able to separate more than two sources.

4.2.1 Sources

Audio signals: We first evaluated the performance of the algorithm on two different test sets of audio signals. Data set A consists of excerpts from 14 string quartets by Béla Bartók. Note that these sources are from the same CD and the same composer and contain the same instruments. They can thus be expected to have similar statistics. Differences in the Δ-values should mainly be due to short-term nonstationarities. This data set provides evidence that the algorithm is able to distinguish between signals with similar global statistics based on short-term fluctuations in their statistics.

Data set B consists of 20 excerpts from popular music pieces from various genres, ranging from classical music over rock to electronic music. The statistics of this set is more variable in their Δ-values, in particular they remain different even for long sampling times.

All sources were sampled at 44,100 Hz and 16 bit, that is, with CD-quality. The length of the samples was varied to assess how the amount of training data affects the performance of the algorithm.

Artificial data: To test how the algorithm would perform in tasks where more than two sources need to be extracted, we generated 6 artificial source signals with different temporal statistics. The sources were colored noise, generated by (i) applying a fast Fourier transform to white noise signals of length T, (ii) multiplying the resulting signals with $\exp(-f^2/2\sigma_i^2)$ (where f denotes the frequency) and (iii) inverting the Fourier transform. The parameter σ_i controls the Δ-values of the sources ($\Delta \approx \sigma_i^2$) and was chosen such that the Δ-values were roughly equidistant: $\sigma_i = \sqrt{i}\frac{T}{50} + 1$.

4.2.2 Nonlinear Mixtures

Audio signals: We subjected all possible pairs of sources within a data set to a nonlinear invertible mixture that was previously used by Harmeling et al. (2003) and Blaschke et al. (2007):

$$\begin{aligned} x_1(t) &= (s_2(t) + 3s_1(t) + 6)\cos(1.5\pi s_1(t))\,, \\ x_2(t) &= (s_2(t) + 3s_1(t) + 6)\sin(1.5\pi s_1(t))\,. \end{aligned} \tag{15}$$

Figure 3 illustrates the spiral-shaped structure of this nonlinearity. This mixture is only invertible if the sources are bounded between -1 and 1, which is the case for the audio data we used. The mixture (15) is not symmetric in s_1 and s_2. Thus, for every pair of sources, there are two possible mixtures and we have tested both for each source pair.

We have also tested the other nonlinearities that Harmeling et al. (2003) have applied to two sources, as well as post-nonlinear mixtures, that is, linear mixture followed by a point nonlinearity. The performance was similar for all mixtures tested without any tuning of parameters (data not shown). Moreover, the performance remained practically unchanged when we used linear mixtures or no mixture at all. This is in line with the argument that the mixture should be irrelevant to SFA if the function space $\mathcal{F}$ is sufficiently rich (see section 3).

Separation of more than two sources: For the simulations with more than two sources, we created a nonlinear mixture by applying a post-nonlinear mixture twice. The basic post-nonlinear mixture is generated by first applying a random rotation O_{ij} to the sources s_i and then applying a point-nonlinearity to each of the linearly mixed signals. We used an

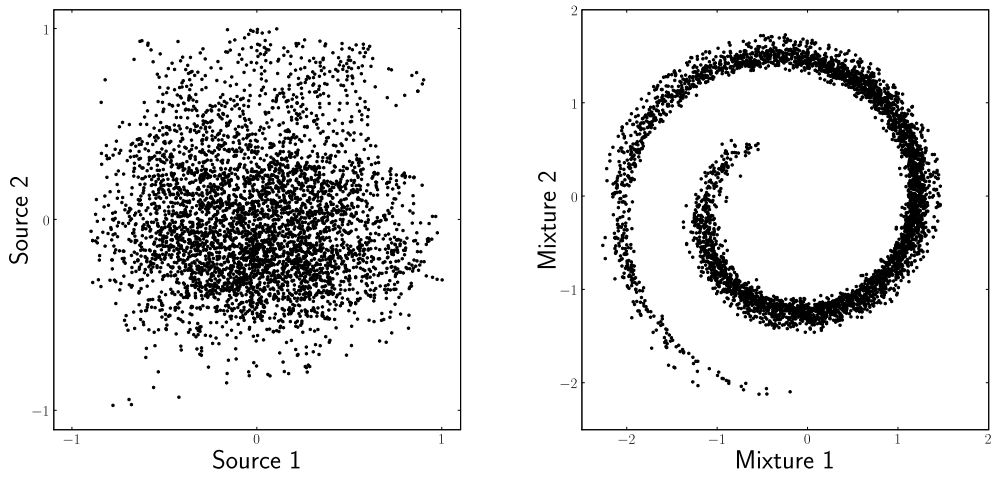


Figure 3: The spiral-shaped structure of the nonlinear mixture. Panel A shows a scatter plot of two sources from data set A. Panel B shows a scatter plot of the nonlinear mixture we used to test the algorithm.

arctangent as a nonlinearity:

$$M_i(\mathbf{s}) = \arctan\left(\zeta^{-1}\left(\sum_j O_{ij} s_j\right)\right) ,$$

with a parameter ζ that controls the strength of the nonlinearity. We normalized the sources to have zero mean and unit variance to ensure that the degree of nonlinearity is roughly the same for all combinations of sources and chose $\zeta = 2$.

This nonlinearity was applied twice, with independently generated rotations, and a normalization step to zero mean and unit variance before each application.

4.2.3 Simulation Parameters

There are three parameters in the algorithm: the degree N^{SFA} of the expansion used for the first SFA step, the degree N^{nl} of the expansion for the source removal and the threshold ϵ for the removal of directions with negligible variance.

Degree of the expansion in the first SFA step: For the simulations with two sources, we used a polynomial expansion of degree $N^{\mathrm{SFA}} = 7$, because it has previously been shown that this function space is sufficient to invert the mixture (15) (Blaschke et al., 2007). For 2-dimensional input signals, this expansion generates a 35-dimensional function space. We kept all $J = 35$ output signals of SFA. It is worth noting that the success rate of the algorithm is practically unchanged when polynomials of higher order are used. From the theoretical perspective, this is not surprising, because once the function space is sufficiently rich to extract the first harmonics of the sources, the system performs just as good as it could with an unrestricted function space.

For the simulations with more than two sources, we used a polynomial expansion of degree $N^{\mathrm{SFA}} = 3$.

Degree of the expansion for source removal: For the simulations with two sources, we expanded the estimate for the first source in polynomials of degree $N^{\mathrm{nl}} = 20$, that is, we projected out 20 nonlinear versions of the first source. Using fewer nonlinear versions does not alter the results significantly, as long as the expansion is sufficiently complex to remove those harmonics of the first source that have smaller Δ-values than the second source. Using higher expansion degrees sometimes leads to numerical instabilities, which we accredit to the extremely sparse distribution that results from the application of very high monomials.

For the separation of more than two sources, all polynomials of degree $N^{\mathrm{nl}} = 4$ of the already estimated sources were projected out.

Variance threshold: After the removal of the nonlinear versions of the first source, there is at least one direction with vanishing variance. To avoid numerical problems caused by singularities in the covariance matrices, directions with variance below $\epsilon = 10^{-7}$ were removed. For almost all source pairs, the only dimension that had a variance below ϵ after the removal was the trivial direction of the first estimated source.

The simulations were done in Python using the modular toolkit for data processing (MDP) developed by Zito et al. (2008). The xSFA algorithm is included in the current version of MDP (http://mdp-toolkit.sourceforge.net).

4.2.4 Performance Measure

For stationary Gaussian sources, the theory predicts that the algorithm should reconstruct the sources exactly. In most applications, however, the sources are neither Gaussian nor stationary (at least not on the time scales we used for training). In this case the algorithm cannot be expected to find the sources themselves, but rather a nonlinearly transformed version of the sources, ideally their lowest harmonics. Thus, the correlation between the output signals of the algorithm and the sources is not necessarily the appropriate measure for the quality of the source separation. Therefore, we also calculated the lowest harmonics $g_{\alpha 1}$ of the sources by applying SFA with a polynomial expansion of degree 11 to the individual sources separately and then calculated the correlations between the output signals of the algorithm and both the output signals of the harmonics $y_{\alpha 1}(t) = g_{\alpha 1}(s_\alpha(t))$ and the sources themselves. In addition to the correlation coefficient, we also calculated the signal-to-noise ratio.

4.2.5 Simulation Results

Figure 4 shows the performance of the algorithm depending on the duration of the training data. To provide an idea of the statistics of the performance, we plot the median as well as the 25th and 75th percentile of the distribution of the correlation coefficient and the signal-to-noise ratio. For data set A, the algorithm requires on the order of 0.5s of training data to extract the first source with a median signal-to-noise ratio (SNR) of about 18 (corresponding to a correlation coefficient (CC) larger than 0.99) and the second source with a median SNR of about 13 (CC$>$ 0.95). Although the median performance increases slowly as the duration of the training data increases to several seconds, the growing distance between the percentiles indicates a larger inhomogeneity in the results, suggesting that for

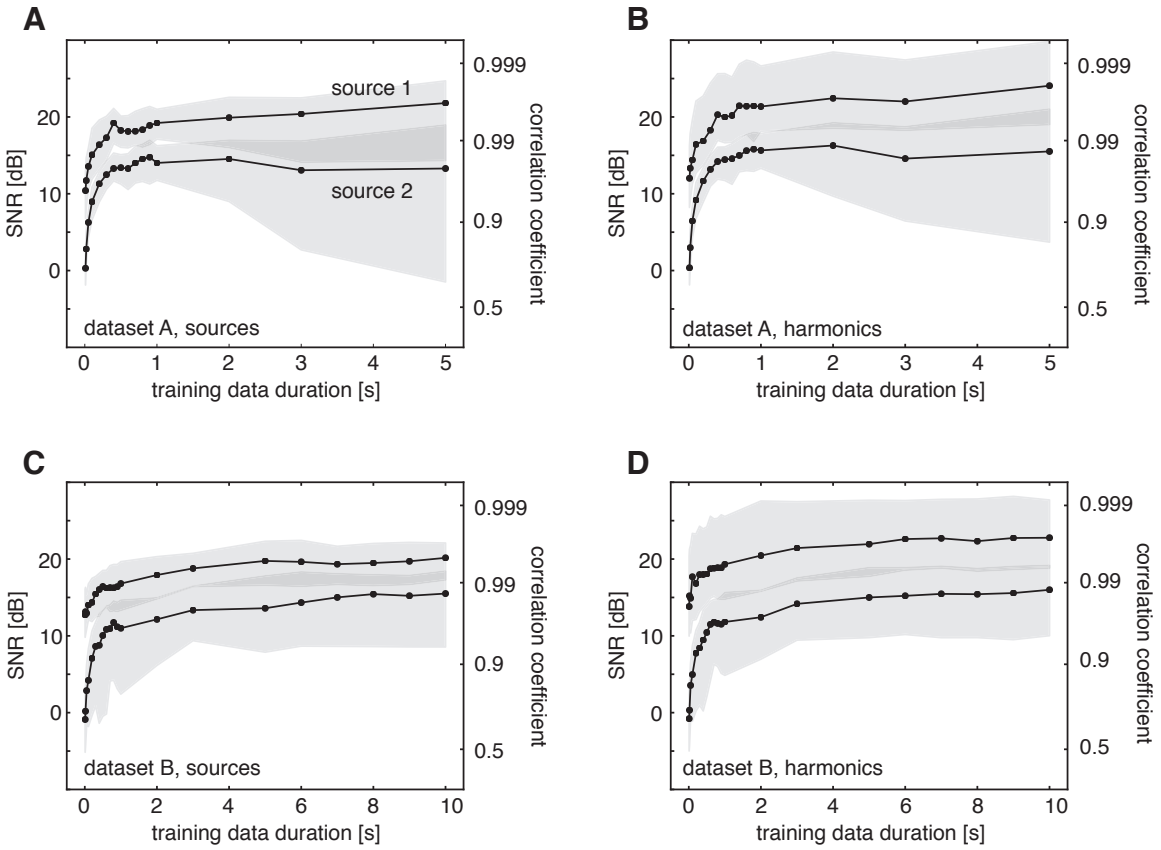


Figure 4: Performance of the algorithm as a function of the duration of the training data. The curves show the median of the distribution of correlation coefficients between the reconstructed and the original sources, as well as the corresponding signal-to-noise ratio (SNR). Grey-shaded areas indicate the region between the 25th and the 75th percentile of the distribution of the correlation/SNR. Statistics cover all possible source pairs that can be simulated (data set A: 14 sources $\rightarrow$ 182 source pairs, data set B: 20 sources $\rightarrow$ 380 source pairs). Panels A and B show results for data set A, panels C and D for data set B. Panels A and C show the ability of the algorithm to reconstruct the sources themselves, while B and D show the performance when trying to reconstruct the slowest harmonics of the sources. Note the difference in time scales.

long durations, the algorithm either performs very well or fails completely. We attribute this behavior to the fact that all sources were string quartets whose temporal statistics are relatively similar in the long term. The SNR is only slightly higher when comparing the extracted sources to the slowest harmonics of the original sources. This may serve as an indication that the sources were close to Gaussian, so that the harmonics and the sources were similar.

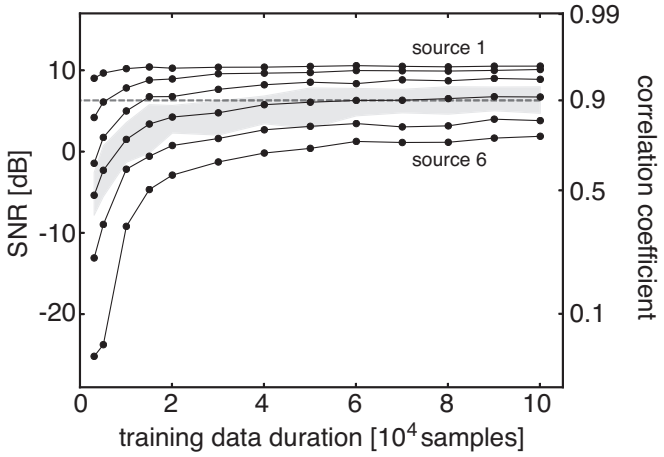


Figure 5: Performance of the algorithm for multiple sources. The curves show the median of the distribution of correlation coefficients between the reconstructed and the orginal six sources, as well as the corresponding signal-to-noise ratio (SNR). The grey-shaded area indicates the region between the 25th and the 75th percentile of the distribution of the correlation/SNR for the 4th source. The percentiles for the other sources are similar but not shown for reasons of graphical clarity. Statistics cover 50 repetitions with independently generated sources. The dashed grey line indicates the performance of a linear regression.

For data set B, longer training times of at least 2s were necessary to reach a similar performance as for data set A. Further research is necessary to assess the reasons for this. Again, the estimated sources are more similar to the slowest harmonics of the sources than to the sources themselves. The reconstruction performance increases with the duration of the training data. For this data set, the prominent divergence of the percentiles for data set A is not observed.

The performance of xSFA is significantly better than that of independent slow feature analysis (ISFA; Blaschke et al., 2007), which also relies on temporal correlations and was reported to reconstruct both sources with CC$>$ 0.9 for about 70% of the source pairs. For our data sets, both sources were reconstructed with a correlation of more than 0.9 for more than 90% of the source pairs, if the duration of the training data was sufficiently large. Moreover, it is likely that the performance of xSFA can be further improved, for example, by using more training data or different function spaces.

The algorithm is relatively fast: On a notebook with 1.7GHz, the simulation of the 182 source pairs for data set A with 0.2s training sequences takes about 380 seconds, which corresponds to about 2.1s for the unmixing of a single pair.

Figure 5 shows the performance of the algorithm for the problem where six artificial sources were to be extracted from a nonlinear mixture. The slowest source is extracted with the highest SNR, and the SNR decreases with increasing Δ-value of the sources. This is most likely due to an accumulation of error that arises from the iterative structure of the xSFA algorithm (see discussion in section 5). The performance increases monotonically

with increasing amount of training data. For 10^5 training data points, four of six estimated sources have a correlation coefficient with the original source that is larger than 0.9. The performance of a supervised linear regression between the sources and the mixture happens to be close to 0.9. For the first four extracted sources, xSFA is thus performing better than any linear technique could.

5. Practical Limitations

There are several reasons why the algorithm can fail, because some of the assumptions underlying the theory are not necessarily fulfilled in simulations. In the following, we discuss some of the reasons for failures. The main insights are summarized at the end of the section.

5.1 Limited Sampling Time

The theory predicts that some of the output signals reproduce the harmonics of the sources exactly. However, problems can arise if eigenfunctions have (approximately) the same eigenvalue. For example, assume that the sources have the same temporal statistics, so that the Δ-value of their slowest harmonics $g_{\mu 1}$ is equal. Then, there is no reason for SFA to prefer one signal over the other.

Of course, in practice, two signals are very unlikely to have exactly the same Δ-value. However, the difference may be so small that it cannot be resolved because of limited sampling. To get a feeling for how well two sources can be distinguished, assume there were only two sources that are drawn independently from probability distributions with Δ-values Δ and $\Delta + \delta$. Then linear SFA should ideally reproduce the sources exactly. However, if there is only a finite amount of data, say of total duration T, the Δ-values of the signals can only be estimated with finite precision. Qualitatively, we can distinguish the sources when the standard deviation of the estimated Δ-value is smaller than the difference δ in the "exact" Δ-values. It is clear that this standard deviation depends on the number of data points roughly as $1/\sqrt{T}$. Thus the smallest difference $\delta_{\min}$ in the Δ-values that can be resolved has the functional dependence

$$\delta_{\min} \sim \Delta^{\alpha} \frac{1}{\sqrt{T}} \,.$$

The reason why the smallest distinguishable difference δ must depend on the Δ-value is that subsequent data points are not statistically independent, because the signals have a temporal structure. For slow signals, that is, signals with a small Δ-values, the estimate of the Δ-value is less precise than for quickly varying signals, because the finite correlation time of the signals impairs the quality of the sampling. For dimensionality reasons, the exponent α has to take the value $\alpha = 3/4$, yielding the criterion

$$\frac{\delta_{\min}}{\Delta} \sim \frac{1}{\sqrt{T\sqrt{\Delta}}} \,.$$

For an interpretation of this equation note that the Δ-value can be interpreted as a (quadratic) measure for the width of the power spectrum of a signal (assuming a roughly unimodal power

spectrum centered at zero):

$$\Delta(y) = \frac{1}{T}\int \dot{y}^2 \mathrm{d}t = \frac{1}{T}\int \omega^2 |y(\omega)|^2 \mathrm{d}\omega\,, \tag{16}$$

where $y(\omega)$ denotes the Fourier transform of $y(t)$. However, the inverse width of the power spectrum is an operative measure for the correlation time τ of the signal, leaving us with a correlation time $\tau \sim 1/\sqrt{\Delta}$. With this in mind, the criterion (16) takes a form that is much easier to interpret:

$$\frac{\delta_{\min}}{\Delta} \sim \sqrt{\frac{\tau}{T}} = \frac{1}{\sqrt{N_\tau}}\,. \tag{17}$$

The correlation time τ characterizes the time scale on which the signal varies, so intuitively, we can cut the signal into $N_\tau = T/\tau$ "chunks" of duration τ, which are approximately independent. Equation (17) then states that the smallest relative difference in the Δ-value that can be resolved is inversely proportional to the square root of the number N_τ of independent data "chunks".

If the difference in the Δ-value of the predicted solutions is smaller than $\delta_{\min}$, SFA is likely not to find the predicted solutions but rather an arbitrary mixture thereof, because the removal of random correlations and not slowness is the essential determinant for the solution of the optimization problem. Equation (17) may serve as an estimate of how much training time is needed to distinguish two signals. Note however, that the validity of (17) is questionable for nonstationary sources, because the statistical arguments used above are not valid.

Using these considerations, we can estimate the order of magnitude of training data that is needed for the data sets we used to evaluate the performance of the algorithm. For both data sets, the Δ-values of the sources were on the order of 0.01, which corresponds to an autocorrelation time of approximately $1/\sqrt{0.01} = 10$ samples. Those sources of data set A that were most similar differed in Δ-value by $\delta/\Delta \sim 0.05$, which requires $N_\tau = (1/0.05)^2 = 400$. This corresponds to ~ 4000 samples that are required to distinguish the sources, which is similar to what was observed in simulations. In data set B, the problem is not that the sources are too similar, but rather that they are too different in Δ-value, which makes it difficult to distinguish between the products of the second source and harmonics of the first and the second source alone. The Δ-values often differ by a factor of 20 or more, so that the relative difference between the relevant Δ-values is again on the order of 5%. In theory, the same amount of training data should therefore suffice. However, if the sources strongly differ in Δ-value, many harmonics need to be projected out before the second source is accessible, which presumably requires a higher precision in the estimate of the first source. This might be one reason why significantly more training data is needed for data set B.

5.2 Sampling Rate

The theory is derived under the assumption that all signals are continuous in time. Real data are generally discretized. Therefore, the theory is only valid if the data are sampled at a sampling rate sufficient to generate quasi-continuous data. As the sampling rate decreases, so do the correlations between subsequent data points. In the limit of extremely

low sampling rates this renders techniques like SFA that are based on short-term temporal correlations useless.

For discrete data, the temporal derivative is usually replaced by a difference quotient:

$$\dot{y}(t) \approx \frac{y(t+\Delta t) - y(t)}{\Delta t}\,,$$

where $y(t+\Delta t)$ and $y(t)$ are neighboring sample points and Δt is given by the inverse of the sampling rate r. The Δ-value can then be expressed in terms of the variance of the signal and its autocorrelation function:

$$\Delta(y) = \langle \dot{y}^2 \rangle_t \approx \frac{2}{\Delta t^2} \left(\langle y^2 \rangle_t - \langle y(t+\Delta t) y(t) \rangle_t \right) = 2r^2 \left(\langle y^2 \rangle_t - \langle y(t+\Delta t) y(t) \rangle_t \right) . \qquad (18)$$

If the sampling is too low, the signal effectively becomes white noise. In this case, the term that arises from the time-delayed correlation vanishes, while the variance remains constant. Thus, for small sampling rates, the Δ-value depends quadratically on the sampling rate, while it saturates to its "real" value if the sampling rate is increased. This behavior is illustrated in Figure 6A. Note that two signals with different Δ-values for sufficient sampling rate may have very similar Δ-value when the sampling is decreased too drastically. Intuitively, this is the case if the sampling rate is so low that both signals are (almost) white noise. In this case, there are no temporal correlations that could be exploited, so that SFA returns a random mixture of the signals.

The number of samples N that can be used for training is limited by the working memory of the computer and/or the available CPU time. Thus, for a fixed maximal number of training samples N, the sampling rate implicitly determines the maximal training time $T = N/r$. The training time, in turn, determines the minimal relative difference in Δ-value that can be distinguished (cf. Equation (17)). Thus, for a fixed number of sample points, the minimal relative difference in Δ-value that can be resolved is proportional to $1/\sqrt{T} \sim \sqrt{r}$. But why do low sampling rates lead to a better resolution? The reason is that for high sampling rates, neighboring data points have essentially the same value. Thus, they do not help in estimating the Δ-value, because they do not carry new information.

In summary, the sampling rate should ideally be in an intermediate regime. If the sampling rate is too low, the signals become white noise and cannot be distinguished, while too high sampling rates lead to high computational costs without delivering additional information. This is illustrated in Figure 6B.

5.3 Density Of Eigenvalues

The problem of getting random mixtures instead of the optimal solutions is of course most relevant in the case where the sources, or more precisely, the slowest non-constant harmonics of the sources have similar Δ-values. However, even when the sources are sufficiently different, this problem eventually arises for the higher-order solutions. To quantify the expected differences in Δ-value between the solutions, we define a *density* $\rho(\Delta)$ of the Δ-values as the number of eigenvalues expected in an interval $[\Delta, \Delta + \delta]$, divided by the interval length δ. A convenient way to determine this density is to calculate the number $R(\Delta)$ of solutions with eigenvalues smaller than Δ and then take the derivative with respect to Δ.

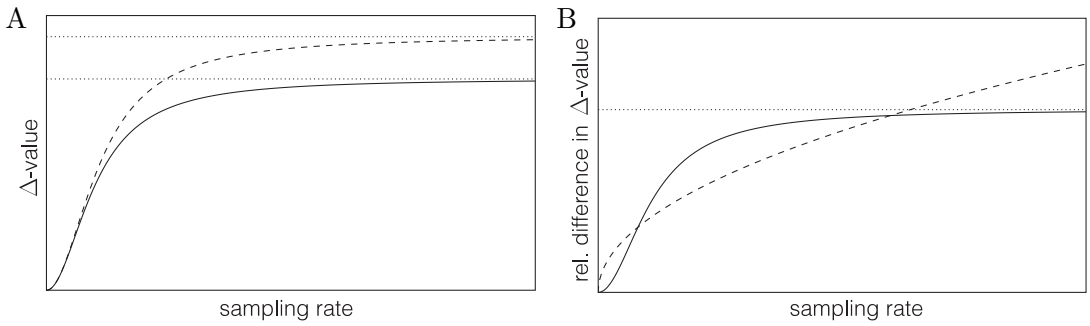


Figure 6: Influence of the sampling rate. (A) Qualitative dependence of the Δ-value of two different signals on the sampling rate. For very low sampling rates, both signals become white noise and the Δ-value quadratically approaches zero. Signals that have different Δ-values for sufficiently high sampling rates may therefore not be distinguished if the sampling rate is too low. The dotted lines indicate the "real" Δ-values of the signals. Note: It may sound counterintuitive that the Δ-value drops to zero with decreasing sampling rate, as white noise should be regarded as a quickly varying signal. This arises from taking the sampling rate into account in the temporal derivative (18). If the derivative is simply replaced by the difference between adjacent data points, the Δ-value approaches 2 as the sampling rate goes to zero and decreases with the inverse square of the sampling rate as the sampling rate becomes large. (B) Sampling rate dependence of the "resolution" of the algorithm for a fixed number of training samples. The solid line shows the qualitative dependence of the relative difference in Δ-value of two signals as a function of the sampling rate and the dashed line shows the qualitative behavior of the minimal relative difference in Δ-value that can be resolved. The signals can only be separated by SFA if the resolvable difference (dashed) is below the expected relative difference (solid). Therefore an intermediate sampling is more efficient. The dotted line indicates the "real" ratio of the Δ-values.

In the Gaussian approximation, the Δ-values of the harmonics are equidistantly spaced, cf. Equation (14). As the Δ-value $\Delta_{\mathbf{i}}$ of the full product solution $g_{\mathbf{i}}$ is the sum of the Δ-values of the harmonics, the condition $\Delta_{\mathbf{i}} < \Delta$ restricts the index $\mathbf{i}$ to lie below a hyperplane with the normal vector $\mathbf{n} = (\lambda_{11}, ..., \lambda_{S1}) \in \mathbb{R}^S$:

$$\sum_{\mu} i_\mu \lambda_{\mu 1} = \mathbf{i} \cdot \mathbf{n} < \Delta \,. \tag{19}$$

Because the indices are homogeneously distributed in index space with density one, the expected number of solutions with $\Delta < \Delta_0$ is simply the volume of the subregion in index space for which Equation (19) is fulfilled:

$$R(\Delta) = \frac{1}{S!} \prod_{\mu=1}^{S} \frac{\Delta}{\lambda_{\mu 1}} \,.$$

The density of the eigenvalues is then given by

$$\rho(\Delta) = \frac{\partial R(\Delta)}{\partial \Delta} = \frac{1}{(S-1)!}\left[\prod_{\mu}\frac{1}{\lambda_{\mu 1}}\right]\Delta^{S-1}\,.$$

As the density of the eigenvalues can be interpreted as the inverse of the expected distance between the Δ-values, the distance and thus the separability of the solutions with a given amount of data declines as $1/\Delta^{S-1}$. In simulations, we can expect to find the theoretically predicted solutions only for the slowest functions, higher order solutions tend to be linear mixtures of the theoretically predicted functions. This is particularly relevant if there are many sources, that is, if S is large.

If the sources are not Gaussian, the dependence of the density on the Δ-value may have a different dependence on Δ (e.g., for uniformly distributed sources $\rho(\Delta) \sim \Delta^{S/2-1}$). The problem of decreasing separability, however, remains.

5.4 Function Space

An assumption of the theory is that the function space accessible to SFA is unlimited. However, any application has to restrict the function space to a finite dimensionality. If the function space is ill-chosen in that it cannot invert the mixture that generated the input data from the sources, it is clear that the theory can no longer be valid.

Because the nature of the nonlinear mixture is not known *a priori*, it is difficult to choose an appropriate function space. We used polynomials with relatively high degree. A problem with this choice is that high polynomials generate extremely sparse data distributions. Depending on the input data at hand, it may be more robust to use other basis functions such as radial basis functions or kernel approaches (Böhmer et al., 2012), although for SFA, these tend to be computationally more expensive.

The suitability of the function space is one of the key determinants for the quality of the estimation of the first source. If this estimate is not accurate but has significant contributions from other sources, the nonlinear versions of the estimate that are projected out are not accurate, either. The projection step may thus remove aspects of the second source and thereby impair the estimate of the second source. For many sources, these errors accumulate so that estimates for faster sources will not be trustworthy, an effect that is clearly visible in the simulations with more sources. This problem might be further engraved by the increasing eigenvalue density discussed above.

5.5 Summary

In summary, we have discussed four factors that have an influence on simulation results:

- **Limited sampling time:** Whether the algorithm can distinguish two sources with similar Δ-values depends on the amount of data that is available. More precisely, to separate two sources with Δ-values Δ and $\Delta + \delta$, the duration T of the training data should be on the order of $T \sim \tau\,(\Delta/\delta)^2$ or more. Here, τ is the autocorrelation time of the signals, which can be estimated from the Δ-value of the sources: $\tau \approx 1/\sqrt{\Delta}$.
- **Sampling rate:** Because the algorithm is based on temporal correlations, the sampling rate should of course be sufficiently high to have significant correlations between

subsequent data points. If the number T of samples that can be used is limited by the memory capacity of the computer, very high sampling rates can be a disadvantage, because the correlation time τ (measured in samples) of the data is long. Consequently, the number T/τ of "independent data chunks" is smaller than with lower sampling rates, which may impair the ability of the algorithm to separate sources with similar Δ-values (see previous point).

- **Density of eigenvalues:** The problem of similar Δ-values is not only relevant when the sources are similar, because the algorithm also needs to distinguish the faster sources from products of these sources with higher-order harmonics of the lower sources. To estimate how difficult this is, we have argued that, for the case of Gaussian sources, the expected difference between the Δ-values of the output of SFA declines as $1/\Delta^{S-1}$, where S is the number of sources. Separating a source from the product solutions of lower-order sources therefore becomes more difficult with increasing number of sources.

- **Function space:** Another important influence on the performance of the system is the choice of the function space $\mathcal{F}$ for SFA. Of course, $\mathcal{F}$ has to be chosen sufficiently rich to allow the inversion of the nonlinear mixture. According to the theory additional complexity of the function spaces should not alter the results and we have indeed found that the system is rather robust to the particular choice of $\mathcal{F}$, as long as it is sufficiently complex to invert the mixture. We expect, however, that an extreme increase in complexity leads to (a) numerical instabilities (in particular for polynomial expansions as used here) and (b) overfitting effects.

6. Relation To Other Nonlinear BSS Algorithms

Because xSFA is based on temporal correlations, in a very similar way as the kernel-TDSEP (kTSDEP) algorithm presented by Harmeling et al. (2003), one could expect the two algorithms to have similar performance. By using the implementation of the kTDSEP algorithm made available by the authors,[2] we compared kTDSEP with xSFA on the audio signals from data set A in the case of the spiral mixture (15). For the best parameter setting we could identify, kTDSEP was able to recover both sources (with correlation >0.9) for only 20% of the signal pairs, while xSFA recovered both sources for more than 90% of the source pairs with the same training data. This result was obtained using a training data duration of 0.9 s, 25 time-shifted covariance matrices, a polynomial kernel of degree 7, and k-means clustering with a maximum of 10000 points considered. Results depended strongly but not systematically on training data duration. A regression analysis for a few of the failure cases revealed that the sources were present among the extracted components, but not properly selected, suggesting that the poor performance was primarily caused by a failure of the automatic source selection approach of Harmeling et al. (2003). The kTDSEP algorithm would resemble xSFA even more if kernel PCA were used instead of k-means clustering for finding a basis in the kernel feature space. Using kernel PCA, however, yields worse results: both sources were recovered at best in 5% of the signal pairs. The influence of the kernel

2. The code is available on `http://people.kyb.tuebingen.mpg.de/harmeling/code/ktdsep-0.2.tar`.

choice (kernel PCA/k-means) on the performance could be due to numerical instabilities and small eigenvalues, which we avoid in xSFA by using singular value decomposition with thresholding in the SFA dimensionality reduction step.

Almeida (2003) has suggested a different approach (MISEP) that uses a multilayer perceptron to extend the maximum entropy ansatz of Bell and Sejnowski (1995) to the nonlinear case. MISEP has been shown to work in an application to real data (Almeida, 2005). In our hands, MISEP was not able to solve the spiral-shaped nonlinear mixture described in Section 4, however, exactly because of the problems described by Hyvärinen and Pajunen (1999): it converges to a nonlinear mixture of the sources that generates statistically independent output signals. Conversely, xSFA fails to solve the image unmixing problem on which MISEP was successful (Almeida, 2005), probably because of low-frequency components that introduce correlations between the images (Ha Quang and Wiskott, 2013). Whether an information-theoretic ansatz like MISEP or a temporal approach like xSFA is more suitable therefore seems to depend on the problem at hand.

Zhang and Chan (2008) have suggested that the indeterminacies of the nonlinear BSS problem could be solved by a minimal nonlinear distortion (MND) principle, which assumes that the mixing function is smooth. To exploit this, they added a regularization term to common nonlinear ICA objective functions (including that of MISEP). They investigated both a global approach that punishes deviations of the unmixing nonlinearity from the best linear solution and a local approach that favors locally smooth mappings. The latter is remotely related to xSFA, which also tries to enforce smooth mappings, but measures smoothness in time rather than directly in the unmixing function. The MND ansatz applies to arbitrary functions, while xSFA is limited to time-varying data. On the other hand, the temporal smoothness constraint of xSFA could extend to problems where the original sources are smooth, but the mixing function is not.

A nonlinear BSS approach that is even more akin to SFA is the diffusion-map ansatz of Singer and Coifman (2008). Diffusion maps and Laplacian eigenmaps are closely related to SFA (Sprekeler, 2011). A key difference lies in the choice of the local metrics of the data, which is dictated by the temporal structure for SFA (the matrix $K_{\alpha\beta}$ can be thought of as an inverse metric tensor), but hand-chosen for diffusion maps. Singer & Coifman made a data-driven choice for the metric tensor through local inspection of the data manifold, and showed that the resulting diffusion maps can reconstruct the original sources in a toy example (Singer and Coifman, 2008) and extract slowly varying manifolds in time series data (Singer et al., 2009).[3]

7. Discussion

In this article, we have extended previous theoretical results on SFA to the case where the input data are generated from a set of statistically independent sources. The theory shows that (a) the optimal output of SFA consists of products of signals, each of which depends on a single source only and that (b) some of these harmonics should be monotonic functions of the sources themselves. Based on these predictions, we have introduced the xSFA algorithm to iteratively reconstruct the sources, in theory from arbitrary invertible mixtures. Simulations have shown that the performance of xSFA is substantially higher

3. An SFA-based approach to a similar problem has been suggested by Wiskott (2003a).

than the performance of independent slow feature analysis (ISFA; Blaschke et al., 2007) and kTDSEP (Harmeling et al., 2003), other algorithms for nonlinear BSS that also rely on temporal correlations.

xSFA is relatively robust to changes of parameters. Neither the degree of the expansion before the first SFA step nor the number of removed nonlinear versions of the first source need to be finely tuned, though both need to be within a certain range, so that the BSS problem can be solved without running into the overfitting or error accumulation problems discussed above. It should be noted, moreover, that polynomial expansions - as used here - become problematic if the degree of the expansion is too high. The resulting expanded data contain directions with very sparse distributions, which can lead (a) to singularities in the covariance matrix (e.g., for Gaussian signals with limited sampling, x^{20} and x^{22} are almost perfectly correlated) and (b) to sampling problems for the estimation of the required covariances because the data are dominated by few data points with high values. Note, that this problem is not specific to the algorithm itself, but rather to the expansion type used. Other expansions such as radial basis functions may be more robust. The relative insensitivity of xSFA to parameters is a major advantage over ISFA, whose performance depended crucially on the right choice of a trade-off parameter between slowness and independence.

Many algorithms for nonlinear blind source separation are designed for specific types of mixtures, for example, for post-nonlinear mixtures (for an overview of methods for post-nonlinear mixtures see Jutten and Karhunen, 2003). In contrast, our algorithm should work for arbitrary instantaneous mixtures. As previously mentioned, we have performed simulations for a set of instantaneous nonlinear mixtures and the performance was similar for all mixtures. The only requirements are that the sources are distinguishable based on their Δ-value and that the function space accessible to SFA is sufficiently complex to invert the mixture. Note that the algorithm is restricted to instantaneous mixtures. It cannot invert convolutive mixtures because SFA processes its input instantaneously and is thus not suitable for a deconvolution task.

It would be interesting to see if the theory for SFA can be extended to other algorithms. For example, given the close relation of SFA to TDSEP (Ziehe and Müller, 1998), a variant of the theory may apply to the kernel version of TDSEP (Harmeling et al., 2003). In particular, it would be interesting to see whether the theory would suggest an alternative source selection algorithm for kTDSEP that is more robust.

In summary, we have presented a new algorithm for nonlinear blind source separation that is (a) independent of the mixture type, (b) robust to parameters, (c) underpinned by a rigorous mathematical framework, and (d) relatively reliable, as shown by the reconstruction performance for the examined cases.

Acknowledgments

We want to thank Stefan Harmeling for his valuable help in the comparison of xSFA with kTDSEP. This work was supported by the Volkswagen Foundation through a junior research group to L.W.. H.S. was supported by the German ministry for Science and Education (grant no. 01GQ1201).

Appendix A. Proof Of Theorem 1

The proof that all product functions $g_{\mathbf{i}} = \prod_\alpha g_{\alpha i_\alpha}(s_\alpha)$ are eigenfunctions of the operator $\mathcal{D} = \sum_\beta \mathcal{D}_\beta$ can be carried out directly:

$$\begin{aligned}
\mathcal{D}g_{\mathbf{i}}(\mathbf{s}) &= \left(\sum_\beta \mathcal{D}_\beta\right) \prod_\alpha g_{\alpha i_\alpha}(s_\alpha) \\
&= \sum_\beta \mathcal{D}_\beta \prod_\alpha g_{\alpha i_\alpha}(s_\alpha) \\
&= \sum_\beta \left(\mathcal{D}_\beta g_{\beta i_\beta}(s_\beta)\right) \prod_{\alpha\neq\beta} g_{\alpha i_\alpha}(s_\alpha) \\
&\quad \text{(because } \mathcal{D}_\beta \text{ is a differential operator w.r.t. } s_\beta \text{ only)} \\
&= \sum_\beta \lambda_{\beta i_\beta} g_{\beta i_\beta}(s_\beta) \prod_{\alpha\neq\beta} g_{\alpha i_\alpha}(s_\alpha) \\
&= \left(\sum_\beta \lambda_{\beta i_\beta}\right) \prod_\alpha g_{\alpha i_\alpha}(s_\alpha) \\
&= \lambda_{\mathbf{i}} g_{\mathbf{i}}(\mathbf{s})\,.
\end{aligned}$$

Because the product functions are eigenfunctions of the full operator $\mathcal{D}$, the theory of Franzius et al. (2007, Theorems 1-5) applies, stating that the J product functions with the smallest eigenvalue, ordered by their eigenvalue, are the solutions of the optimization problem of SFA. The proof of this theory requires that the eigenfunctions form a complete set. Because the set of eigenfunctions for the individual operators $\mathcal{D}_\alpha$ form a complete set for the individual Sobolev space of functions depending on s_α only, however (Courant and Hilbert, 1989, §14), the product set $g_{\mathbf{i}}$ is also a complete set for the product space.

Appendix B. Proof That $K_{\alpha\beta}$ Is Diagonal

To prove that the matrix $K_{\alpha\beta}(\mathbf{s})$ is diagonal, we first need to prove that the mean temporal derivative of any 1-dimensional signal given its value vanishes: $\langle \dot{s} \rangle_{\dot{s}|s} = \int \dot{s} p(\dot{s}|s) \mathrm{d}\dot{s} = 0$. To do so, we assume that the distribution of the signal is stationary and that the signal is continuously differentiable. Because of the stationarity, the probability that the signal is smaller than a given value s_0 is constant:

$$\begin{aligned}
0 &= \frac{d}{dt} \int_{-\infty}^{s_0} \int_{-\infty}^{\infty} p(s,\dot{s}) \mathrm{d}s \mathrm{d}\dot{s} \\
&= \int_{-\infty}^{s_0} \int_{-\infty}^{\infty} \partial_t p(s,\dot{s}) \mathrm{d}s \mathrm{d}\dot{s} \\
&= -\int_{-\infty}^{s_0} \int_{-\infty}^{\infty} \partial_s \left[\dot{s} p(s,\dot{s})\right] + \partial_{\dot{s}} \left[\ddot{s} p(s,\dot{s}))\right] \mathrm{d}s \mathrm{d}\dot{s}
\end{aligned}$$

where we used the continuity equation $\partial_t p(s,\dot{s}) + \partial_s \left[\dot{s} p(s,\dot{s})\right] + \partial_{\dot{s}} \left[\ddot{s} p(s,\dot{s}))\right] = 0$. Using the divergence theorem and assuming that the probability distribution vanishes as $\dot{s} \to \infty$ for

all $s < s_0$, we get the desired result:

$$\begin{aligned} 0 &= -\int_{-\infty}^{\infty} \dot{s} p(s, \dot{s}) \mathrm{d}\dot{s} \\ &= -p(s) \langle \dot{s} \rangle_{\dot{s}|s} \,. \end{aligned}$$

Because the mean temporal derivative $\langle \dot{s}_\alpha \rangle_{\dot{s}_\alpha | s_\alpha}$ is zero for each signal, the matrix $K_{\alpha\beta} = \langle \dot{s}_\alpha \dot{s}_\beta \rangle_{\dot{\mathbf{s}}|\mathbf{s}}$ is not only the matrix of the second moments of the velocity distribution given the signal values $\mathbf{s}$ but its covariance matrix. Because the signals are statistically independent, they are necessarily uncorrelated, that is, their covariance matrix is diagonal.

References

L. Almeida. MISEP: Linear and nonlinear ICA based on mutual information. *Journal of Machine Learning Research*, 4:1297–1318, 2003.

L. Almeida. Separating a real-life nonlinear image mixture. *Journal of Machine Learning Research*, 6:1199–1229, 2005.

A. Bell and T. Sejnowski. An information maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

P. Berkes and L. Wiskott. Slow feature analysis yields a rich repertoire of complex cells. *Journal of Vision*, 5(6):579–602, 2005.

T. Blaschke, P. Berkes, and L. Wiskott. What is the relation between slow feature analysis and independent component analysis? *Neural Computation*, 18(10):2495–2508, 2006.

T. Blaschke, T. Zito, and L. Wiskott. Independent slow feature analysis and nonlinear blind source separation. *Neural Computation*, 19(4):994–1021, 2007.

W. Böhmer, S. Grünewälder, H. Nickisch, and K. Obermayer. Generating feature spaces for linear algorithms with regularized sparse kernel slow feature analysis. *Machine Learning*, 89:67–86, 2012.

S. Choi. Differential learning algorithms for decorrelation and independent component analysis. *Neural Networks*, 19(10):1558–1567, Dec 2006.

R. Courant and D. Hilbert. *Methods of Mathematical Physics, Part I.* Wiley, 1989.

M. Franzius, H. Sprekeler, and L. Wiskott. Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):e166, 2007.

M. Ha Quang and L. Wiskott. Multivariate slow feature analysis and decorrelation filtering for blind source separation. *Image Processing, IEEE Transactions on*, 22(7):2737–2750, 2013.

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

A. Hyvärinen and P. Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley, 2001.

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.

A. Singer and R. Coifman. Non-linear independent component analysis with diffusion maps. *Applied and Computational Harmonic Analysis*, 25(2):226–239, 2008.

A. Singer, R. Erban, I. G. Kevrekidis, and R. R. Coifman. Detecting intrinsic slow variables in stochastic dynamical systems by anisotropic diffusion maps. *Proceedings of the National Academy of Sciences*, 106(38):16090–16095, 2009.

H. Sprekeler. On the relation of slow feature analysis and Laplacian eigenmaps. *Neural Computation*, 23:3287–3302, 2011.

L. Wiskott. Learning invariance manifolds. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks, ICANN'98, Skövde*, Perspectives in Neural Computing, pages 555–560, London, Sept. 1998. Springer. ISBN 3-540-76263-9.

L. Wiskott. Estimating driving forces of nonstationary time series with slow feature analysis. arXiv.org e-Print archive, `http://arxiv.org/abs/cond-mat/0312317/`, Dec. 2003a.

L. Wiskott. Slow feature analysis: A theoretical analysis of optimal free responses. *Neural Computation*, 15(9):2147–2177, 2003b.

L. Wiskott and T. Sejnowski. Slow feature analysis: unsupervised learning of invariances. *Neural Computation*, 14:715–770, 2002.

K. Zhang and L. Chan. Minimal nonlinear distortion principle for nonlinear independent component analysis. *Journal of Machine Learning Research*, 9:2455–2487, 2008.

A. Ziehe and K.-R. Müller. TDSEP–an efficient algorithm for blind separation using time structure. *Proc. Int. Conf. on Artificial Neural Networks (ICANN '98)*, pages 675–680, 1998.

T. Zito, N. Wilbert, L. Wiskott, and P. Berkes. Modular toolkit for data processing (MDP): A python data processing framework. *Frontiers in Neuroinformatics*, 2:8, 2008.

Journal of Machine Learning Research 15 (2014) 949-980 Submitted 4/11; Revised 5/13; Published 3/14

Natural Evolution Strategies

Daan Wierstra DAAN@DEEPMIND.COM
Tom Schaul TOM@DEEPMIND.COM
DeepMind Technologies Ltd.
Fountain House, 130 Fenchurch Street
London, United Kingdom

Tobias Glasmachers TOBIAS.GLASMACHERS@INI.RUB.DE
Institute for Neural Computation
Universitätsstrasse 150
Ruhr-University Bochum, Germany

Yi Sun YI@IDSIA.CH
Google Inc.
1600 Amphitheatre Pkwy
Mountain View, United States

Jan Peters MAIL@JAN-PETERS.NET
Intelligent Autonomous Systems Institute
Hochschulstrasse 10
Technische Universität Darmstadt, Germany

Jürgen Schmidhuber JUERGEN@IDSIA.CH
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
University of Lugano (USI)/SUPSI
Galleria 2
Manno-Lugano, Switzerland

Editor: Una-May O'Reilly

Abstract

This paper presents Natural Evolution Strategies (NES), a recent family of black-box optimization algorithms that use the natural gradient to update a parameterized search distribution in the direction of higher expected fitness. We introduce a collection of techniques that address issues of convergence, robustness, sample complexity, computational complexity and sensitivity to hyperparameters. This paper explores a number of implementations of the NES family, such as general-purpose multi-variate normal distributions and separable distributions tailored towards search in high dimensional spaces. Experimental results show best published performance on various standard benchmarks, as well as competitive performance on others.

Keywords: natural gradient, stochastic search, evolution strategies, black-box optimization, sampling

1. Introduction

Many real world optimization problems are too difficult or complex to model directly. Therefore, they might best be solved in a 'black-box' manner, requiring no additional information

on the objective function (i.e., the 'fitness' or 'cost') to be optimized besides fitness evaluations at certain points in parameter space. Problems that fall within this category are numerous, ranging from applications in health and science (Winter et al., 2005; Shir and Bäck, 2007; Jebalia et al., 2007) to aeronautic design (Hasenjäger et al., 2005; Klockgether and Schwefel, 1970) and control (Hansen et al., 2009).

Numerous algorithms in this vein have been developed and applied in the past fifty years, in many cases providing good and even near-optimal solutions to hard tasks, which otherwise would have required domain experts to hand-craft solutions at substantial cost and often with worse results. The near-infeasibility of finding globally optimal solutions resulted in a fair amount of heuristics in black-box optimization algorithms, leading to a proliferation of complicated yet frequently highly performant methods.

In this paper, we introduce Natural Evolution Strategies (NES), a novel black-box optimization framework which boasts a relatively clean derivation, yet achieves state-of-the-art performance (with the help of well-chosen heuristic methods). The core idea, similar to the framework of estimation of distribution algorithms (EDAs) (Mühlenbein and Paass, 1996; Larrañaga, 2002; Pelikan et al., 2000) and many evolution strategies approaches (e.g., Ostermeier et al. 1994), is to maintain and iteratively update a search distribution from which search points are drawn and subsequently evaluated. However, NES updates the search distribution in the direction of higher expected fitness using the natural gradient (whereas EDAs, for example, typically use maximum likelihood methods to *fit* the distribution of search points).

1.1 Continuous Black-Box Optimization

The problem of black-box optimization has spawned a wide variety of approaches. A first class of methods was inspired by classic optimization methods, including simplex methods such as Nelder-Mead (Nelder and Mead, 1965), as well as members of the quasi-Newton family of algorithms. Simulated annealing (Kirkpatrick et al., 1983), a popular method introduced in 1983, was inspired by thermodynamics, and is in fact an adaptation of the Metropolis-Hastings algorithm. Other methods, such as those inspired by evolution, have been developed from the early 1950s on. These include the broad class of genetic algorithms (Holland, 1975; Goldberg, 1989), differential evolution (Storn and Price, 1997), estimation of distribution algorithms (Larrañaga, 2002; Pelikan et al., 2000; Bosman and Thierens, 2000; Bosman et al., 2007; Pelikan et al., 2006), particle swarm optimization (Kennedy and Eberhart, 2001), and the cross-entropy method (Rubinstein and Kroese, 2004).

Evolution strategies (ES), introduced by Ingo Rechenberg and Hans-Paul Schwefel in the 1960s and 1970s (Rechenberg and Eigen, 1973; Schwefel, 1977), were designed to cope with high-dimensional continuous-valued domains and have remained an active field of research for more than four decades (Beyer and Schwefel, 2002). ESs involve evaluating the fitness of real-valued genotypes in batch ('generation'), after which the best genotypes are kept, while the others are discarded. Survivors then procreate (by slightly mutating all of their genes) in order to produce the next batch of offspring. This process, after several generations, was shown to lead to reasonable to excellent results for many difficult optimization problems. The algorithm framework has been developed extensively over the years to include the representation of correlated mutations by the use of a full covariance matrix. This allowed

the framework to capture interrelated dependencies by exploiting the covariances while 'mutating' individuals for the next generation. The culminating algorithm, the covariance matrix adaptation evolution strategy (CMA-ES; Hansen and Ostermeier, 2001), has proven successful in numerous studies (e.g., Friedrichs and Igel, 2005; Muller et al., 2002; Shepherd et al., 2006). While evolution strategies have shown to be effective at black-box optimization, analyzing the actual dynamics of the procedure turns out to be difficult, the considerable efforts of various researchers notwithstanding (Beyer, 2001; Jägersküpper, 2007; Jebalia et al., 2010; Auger, 2005; Schaul, 2012f).

1.2 The NES Family

Natural Evolution Strategies (NES) are a family of evolution strategies which iteratively update a search distribution by using an estimated gradient on its distribution parameters.

The general procedure is as follows: the parameterized search distribution is used to produce a batch of search points, and the fitness function is evaluated at each such point. The distribution's parameters (which include strategy parameters) allow the algorithm to adaptively capture the (local) structure of the fitness function. For example, in the case of a Gaussian distribution, this comprises the mean and the covariance matrix. From the samples, NES estimates a search gradient on the parameters towards higher expected fitness. NES then performs a gradient ascent step along the *natural gradient*, a second-order method which, unlike the plain gradient, renormalizes the update w.r.t. uncertainty. This step is crucial, since it prevents oscillations, premature convergence, and undesired effects stemming from a given parameterization (see Section 2.3 and Figure 2 for an overview on how the natural gradient addresses those issues). The entire process reiterates until a stopping criterion is met.

All members of the 'NES family' operate based on the same principles. They differ in the type of distribution and the gradient approximation method used. Different search spaces require different search distributions; for example, in low dimensionality it can be highly beneficial to model the full covariance matrix. In high dimensions, on the other hand, a more scalable alternative is to limit the covariance to the diagonal only. In addition, highly multi-modal search spaces may benefit from more heavy-tailed distributions (such as Cauchy, as opposed to the Gaussian). A last distinction arises between distributions where we can analytically compute the natural gradient, and more general distributions where we need to estimate it from samples.

1.3 Paper Outline

This paper builds upon and extends our previous work on Natural Evolution Strategies (Wierstra et al., 2008; Sun et al., 2009a,b; Glasmachers et al., 2010a,b; Schaul et al., 2011), and is structured as follows: Section 2 presents the general idea of search gradients as described in Wierstra et al. (2008), explaining stochastic search using parameterized distributions while doing gradient ascent towards higher expected fitness. The limitations of the plain gradient are exposed in Section 2.2, and subsequently addressed by the introduction of the natural gradient (Section 2.3), resulting in the canonical NES algorithm.

Section 3 then regroups a collection of techniques that enhance NES's performance and robustness. This includes fitness shaping (designed to render the algorithm invariant w.r.t.

order-preserving fitness transformations (Wierstra et al., 2008), Section 3.1), and adaptation sampling which is a novel technique for adjusting learning rates online (Section 3.2). We provide a novel formulation of NES for the whole class of multi-variate versions of distributions with rotation symmetries (Section 3.3). As special cases we summarize techniques for multivariate Gaussian search distributions, constituting the most common case (Section 3.4). Finally, in Section 3.5, we develop the breadth of the framework, motivating its usefulness and deriving a number of NES variants with different search distributions.

The ensuing experimental investigations show the competitiveness of the approach on a broad range of benchmarks (Section 5). The paper ends with a discussion on the effectiveness of the different techniques and types of distributions and an outlook towards future developments (Section 6).

2. Search Gradients

The core idea of Natural Evolution Strategies is to use *search gradients* (first introduced in Berny, 2000, 2001) to update the parameters of the search distribution. We define the search gradient as the sampled gradient of expected fitness. The search distribution can be taken to be a multinormal distribution, but could in principle be any distribution for which we can find derivatives of its log-density w.r.t. its parameters. For example, useful distributions include Gaussian mixture models and the Cauchy distribution with its heavy tail.

If we use θ to denote the parameters of density $\pi(\mathbf{z}\,|\,\theta)$ and $f(\mathbf{z})$ to denote the fitness function for samples $\mathbf{z}$, we can write the expected fitness under the search distribution as

$$J(\theta) = \mathbb{E}_\theta[f(\mathbf{z})] = \int f(\mathbf{z})\ \pi(\mathbf{z}\,|\,\theta)\ d\mathbf{z}. \tag{1}$$

The so-called 'log-likelihood trick' enables us to write

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \int f(\mathbf{z})\ \pi(\mathbf{z}\,|\,\theta)\ d\mathbf{z} \\
&= \int f(\mathbf{z})\ \nabla_\theta \pi(\mathbf{z}\,|\,\theta)\ d\mathbf{z} \\
&= \int f(\mathbf{z})\ \nabla_\theta \pi(\mathbf{z}\,|\,\theta)\ \frac{\pi(\mathbf{z}\,|\,\theta)}{\pi(\mathbf{z}\,|\,\theta)}\ d\mathbf{z} \\
&= \int \Big[f(\mathbf{z})\ \nabla_\theta \log \pi(\mathbf{z}\,|\,\theta)\Big]\ \pi(\mathbf{z}\,|\,\theta)\ d\mathbf{z} \\
&= \mathbb{E}_\theta\left[f(\mathbf{z})\ \nabla_\theta \log \pi(\mathbf{z}\,|\,\theta)\right].
\end{aligned}$$

From this form we obtain the estimate of the search gradient from samples $\mathbf{z}_1 \ldots \mathbf{z}_\lambda$ as

$$\nabla_\theta J(\theta) \approx \frac{1}{\lambda} \sum_{k=1}^{\lambda} f(\mathbf{z}_k)\ \nabla_\theta \log \pi(\mathbf{z}_k\,|\,\theta), \tag{2}$$

where λ is the population size. This gradient on expected fitness provides a search direction in the space of search distributions. A straightforward gradient ascent scheme can thus

iteratively update the search distribution

$$\theta \leftarrow \theta + \eta \nabla_\theta J(\theta),$$

where η is a learning rate parameter. Algorithm 1 provides the pseudocode for this very general approach to black-box optimization by using a search gradient on search distributions.

Algorithm 1: Canonical Search Gradient algorithm

input: f, θ_{init}
repeat
  for $k = 1 \dots \lambda$ **do**
    draw sample $\mathbf{z}_k \sim \pi(\cdot|\theta)$
    evaluate the fitness $f(\mathbf{z}_k)$
    calculate log-derivatives $\nabla_\theta \log \pi(\mathbf{z}_k|\theta)$
  end
  $\nabla_\theta J \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_\theta \log \pi(\mathbf{z}_k|\theta) \cdot f(\mathbf{z}_k)$
  $\theta \leftarrow \theta + \eta \cdot \nabla_\theta J$
until *stopping criterion is met*

Using the search gradient in this framework is similar to evolution strategies in that it iteratively generates the fitnesses of batches of vector-valued samples—the ES's so-called candidate solutions. It is different however, in that it represents this 'population' as a parameterized distribution, and in the fact that it uses a search gradient to update the parameters of this distribution, which is computed using the fitnesses.

2.1 Search Gradient for Gaussian Distributions

In the case of the 'default' d-dimensional multi-variate normal distribution, the parameters of the Gaussian are the mean $\boldsymbol{\mu} \in \mathbb{R}^d$ (candidate solution center) and the covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ (mutation matrix). Let θ denote these parameters: $\theta = \langle \boldsymbol{\mu}, \boldsymbol{\Sigma} \rangle$. To sample efficiently from this distribution we need a square root of the covariance matrix, that is, a matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ fulfilling $\mathbf{A}^\top \mathbf{A} = \boldsymbol{\Sigma}$. Then $\mathbf{z} = \boldsymbol{\mu} + \mathbf{A}^\top \mathbf{s}$ transforms a standard normal vector $\mathbf{s} \sim \mathcal{N}(0, \mathbb{I})$ into a sample $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Here, $\mathbb{I} = \mathrm{diag}(1, \dots, 1) \in \mathbb{R}^{d \times d}$ denotes the identity matrix. Let

$$\begin{aligned} \pi(\mathbf{z} \,|\, \theta) &= \frac{1}{(\sqrt{2\pi})^d |\det(\mathbf{A})|} \cdot \exp\left(-\frac{1}{2} \left\| \mathbf{A}^{-1} \cdot (\mathbf{z} - \boldsymbol{\mu}) \right\|^2 \right) \\ &= \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \cdot \exp\left(-\frac{1}{2} (\mathbf{z} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{z} - \boldsymbol{\mu}) \right) \end{aligned}$$

denote the density of the multinormal search distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

In order to calculate the derivatives of the log-likelihood with respect to individual elements of θ for this multinormal distribution, first note that

$$\log \pi(\mathbf{z}|\theta) = -\frac{d}{2} \log(2\pi) - \frac{1}{2} \log \det \boldsymbol{\Sigma} - \frac{1}{2} (\mathbf{z} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{z} - \boldsymbol{\mu}).$$

We will need its derivatives, that is, $\nabla_{\boldsymbol{\mu}} \log \pi(\mathbf{z}|\theta)$ and $\nabla_{\boldsymbol{\Sigma}} \log \pi(\mathbf{z}|\theta)$. The first is trivially

$$\nabla_{\boldsymbol{\mu}} \log \pi(\mathbf{z}|\theta) = \boldsymbol{\Sigma}^{-1}(\mathbf{z} - \boldsymbol{\mu}), \tag{3}$$

while the latter is

$$\nabla_{\boldsymbol{\Sigma}} \log \pi(\mathbf{z}|\theta) = \frac{1}{2}\boldsymbol{\Sigma}^{-1}(\mathbf{z} - \boldsymbol{\mu})(\mathbf{z} - \boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1} - \frac{1}{2}\boldsymbol{\Sigma}^{-1}. \tag{4}$$

Using these derivatives to calculate $\nabla_\theta J$, we can then update parameters $\theta = \langle \boldsymbol{\mu}, \boldsymbol{\Sigma} \rangle$ as $\theta \leftarrow \theta + \eta \nabla_\theta J$ using learning rate η. This produces a new center $\boldsymbol{\mu}$ for the search distribution, and simultaneously adapts its associated covariance matrix $\boldsymbol{\Sigma}$. To summarize, we provide the pseudocode for following the search gradient in the case of a multinormal search distribution in Algorithm 2.

Algorithm 2: Search Gradient algorithm: Multinormal distribution

input: f, $\boldsymbol{\mu}_{init}$,$\boldsymbol{\Sigma}_{init}$
repeat
  for $k = 1 \ldots \lambda$ **do**
    draw sample $\mathbf{z}_k \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
    evaluate the fitness $f(\mathbf{z}_k)$
    calculate log-derivatives:
    $\nabla_{\boldsymbol{\mu}} \log \pi(\mathbf{z}_k|\theta) = \boldsymbol{\Sigma}^{-1}(\mathbf{z}_k - \boldsymbol{\mu})$
    $\nabla_{\boldsymbol{\Sigma}} \log \pi(\mathbf{z}_k|\theta) = -\frac{1}{2}\boldsymbol{\Sigma}^{-1} + \frac{1}{2}\boldsymbol{\Sigma}^{-1}(\mathbf{z}_k - \boldsymbol{\mu})(\mathbf{z}_k - \boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}$
  end
  $\nabla_{\boldsymbol{\mu}} J \leftarrow \frac{1}{\lambda}\sum_{k=1}^{\lambda} \nabla_{\boldsymbol{\mu}} \log \pi(\mathbf{z}_k|\theta) \cdot f(\mathbf{z}_k)$
  $\nabla_{\boldsymbol{\Sigma}} J \leftarrow \frac{1}{\lambda}\sum_{k=1}^{\lambda} \nabla_{\boldsymbol{\Sigma}} \log \pi(\mathbf{z}_k|\theta) \cdot f(\mathbf{z}_k)$
  $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \eta \cdot \nabla_{\boldsymbol{\mu}} J$
  $\boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma} + \eta \cdot \nabla_{\boldsymbol{\Sigma}} J$
until *stopping criterion is met*

2.2 Limitations of Plain Search Gradients

As the attentive reader will have realized, there exists at least one major issue with applying the search gradient as-is in practice: It is impossible to *precisely locate* a (quadratic) optimum, even in the one-dimensional case. Let $d = 1$, $\theta = \langle \mu, \sigma \rangle$, and samples $z \sim \mathcal{N}(\mu, \sigma)$. Equations (3) and (4), the gradients on μ and σ, become

$$\begin{aligned} \nabla_\mu J &= \frac{z - \mu}{\sigma^2}, \\ \nabla_\sigma J &= \frac{(z-\mu)^2 - \sigma^2}{\sigma^3}, \end{aligned}$$

and the updates, assuming simple hill-climbing (i.e., a population size $\lambda = 1$) read:

$$\begin{aligned} \mu &\leftarrow \mu + \eta\frac{z - \mu}{\sigma^2}, \\ \sigma &\leftarrow \sigma + \eta\frac{(z-\mu)^2 - \sigma^2}{\sigma^3}. \end{aligned}$$

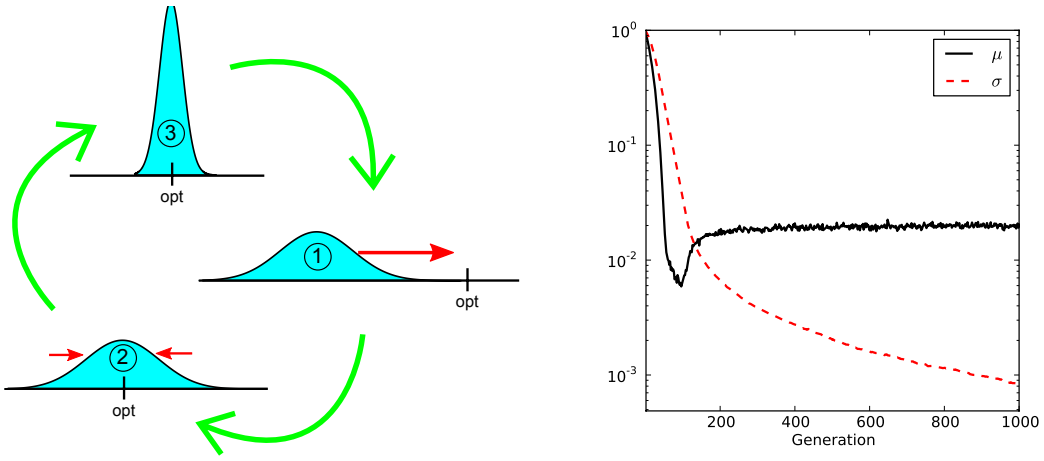


Figure 1: **Left:** Schematic illustration of how the search distribution adapts in the one-dimensional case: from (1) to (2), μ is adjusted to make the distribution cover the optimum. From (2) to (3), σ is reduced to allow for a precise localization of the optimum. The step from (3) to (1) then is the problematic case, where a small σ induces a largely overshooting update, making the search start over again. **Right:** Progression of μ (black) and σ (red, dashed) when following the search gradient towards minimizing $f(\mathbf{z}) = \mathbf{z}^2$, executing Algorithm 2. Plotted are median values over 1000 runs, with a small learning rate $\eta = 0.01$ and $\lambda = 10$, both of which mitigate the instability somewhat, but still show the failure to precisely locate the optimum (for which both μ and σ need to approach 0).

For any objective function f that requires locating an (approximately) quadratic optimum with some degree of precision (e.g., $f(\mathbf{z}) = \mathbf{z}^2$), σ must decrease, which in turn increases the variance of the updates, as $\Delta\mu \propto \frac{1}{\sigma}$ and $\Delta\sigma \propto \frac{1}{\sigma}$ for a typical sample z. In fact, the updates become increasingly unstable, the smaller σ becomes, an effect which a reduced learning rate or an increased population size can only delay but not avoid. Figure 1 illustrates this effect. Conversely, whenever $\sigma \gg 1$ is large, the magnitude of a typical update is severely reduced.

Clearly, this update is not at all *scale-invariant*: Starting with $\sigma \gg 1$ makes all updates minuscule, whereas starting with $\sigma \ll 1$ makes the first update huge and therefore unstable.

This effect need not occur in gradient-based search in general. Here it is rather a consequence of the special situation that the gradient controls both position and variance of a distribution over the same search space dimension. Note that this situation is generic for all translation and scale-invariant families of search distributions. We conjecture that this limitation constitutes one of the main reasons why search gradients have not been developed before: typically, with a naive parameterization, the plain search gradient's performance can be both unstable and unsatisfying; however, the natural gradient extension (introduced in Section 2.3) tackles these issues, and renders search gradients into a viable optimization method by making updates invariant with respect to the particular parameterization used.

2.3 Using the Natural Gradient

Instead of using the plain stochastic gradient for updates, NES follows the *natural gradient*. The natural gradient was first introduced into the field of machine learning by Amari in 1998, and has been shown to possess numerous advantages over the plain gradient (Amari, 1998; Amari and Douglas, 1998). Natural gradients help mitigate the slow convergence of plain gradient ascent in optimization landscapes with ridges and plateaus.

The plain gradient ∇J simply follows the steepest ascent in the space of the actual parameters θ of the distribution. This means that for a given small step-size ε, following it will yield a new distribution with parameters chosen from the hypersphere of radius ϵ and center θ that maximizes J. In other words, the Euclidean distance in parameter space is used to measure the distance between subsequent distributions. Clearly, this makes the update dependent on the particular parameterization of the distribution, therefore a change of parameterization leads to different gradients and different updates. See also Figure 2 for an illustration of how this effectively renormalizes updates w.r.t. uncertainty.

The key idea of the natural gradient is to remove this dependence on the parameterization by relying on a more 'natural' measure of distance $D(\theta'||\theta)$ between probability distributions $\pi(\mathbf{z}|\theta)$ and $\pi(\mathbf{z}|\theta')$. One such natural distance measure between two probability distributions is the Kullback-Leibler divergence (Kullback and Leibler, 1951). The natural gradient can then be formalized as the solution to the constrained optimization problem

$$\begin{aligned} \max_{\delta\theta} J(\theta+\delta\theta) &\approx J(\theta) + \delta\theta^\top \nabla_\theta J, \\ s.t.\ D(\theta+\delta\theta||\theta) &= \varepsilon, \end{aligned} \tag{5}$$

where $J(\theta)$ is the expected fitness of Equation (1), and ε is a small increment size. Now, we have for $\lim \delta\theta \to 0$,

$$D(\theta+\delta\theta||\theta) = \frac{1}{2}\delta\theta^\top \mathbf{F}(\theta)\,\delta\theta,$$

where

$$\begin{aligned} \mathbf{F} &= \int \pi(\mathbf{z}|\theta)\,\nabla_\theta \log \pi(\mathbf{z}|\theta)\,\nabla_\theta \log \pi(\mathbf{z}|\theta)^\top\, d\mathbf{z} \\ &= \mathbb{E}\left[\nabla_\theta \log \pi(\mathbf{z}|\theta)\,\nabla_\theta \log \pi(\mathbf{z}|\theta)^\top\right] \end{aligned}$$

is the *Fisher information matrix* of the given parametric family of search distributions. The solution to the constrained optimization problem in Equation (5) can be found using a Lagrangian multiplier (Peters, 2007), yielding the necessary condition

$$\mathbf{F}\delta\theta = \beta\nabla_\theta J,$$

for some constant $\beta > 0$. The direction of the natural gradient $\widetilde{\nabla}_\theta J$ is given by $\delta\theta$ thus defined. If $\mathbf{F}$ is invertible,[1] the natural gradient amounts to

$$\widetilde{\nabla}_\theta J = \mathbf{F}^{-1}\nabla_\theta J(\theta).$$

1. Care has to be taken because the Fisher matrix estimate may not be (numerically) invertible even if the exact Fisher matrix is.

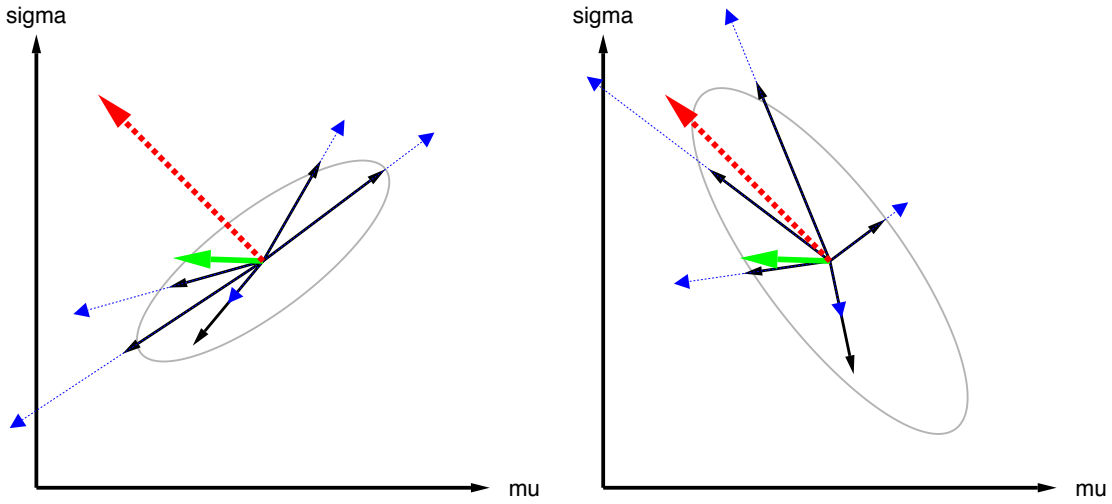


Figure 2: Illustration of plain versus natural gradient in parameter space. Consider two parameters, for example, $\theta = (\mu, \sigma)$, of the search distribution. In the plot on the left, the solid (black) arrows indicate the gradient samples $\nabla_\theta \log \pi(\mathbf{z} \,|\, \theta)$, while the dotted (blue) arrows correspond to $f(\mathbf{z}) \cdot \nabla_\theta \log \pi(\mathbf{z} \,|\, \theta)$, that is, the same gradient estimates, but scaled with fitness. Combining these, the bold (green) arrow indicates the (sampled) fitness gradient $\nabla_\theta J$, while the bold dashed (red) arrow indicates the corresponding natural gradient $\tilde{\nabla}_\theta J$.

Being random variables with expectation zero, the distribution of the black arrows is governed by their covariance, indicated by the gray ellipse. Notice that this covariance is a quantity in *parameter space* (where the θ reside), which is not to be confused with the covariance of the distribution in the *search space* (where the samples $\mathbf{z}$ reside).

In contrast, solid (black) arrows on the right represent $\tilde{\nabla}_\theta \log \pi(\mathbf{z} \,|\, \theta)$, and dotted (blue) arrows indicate the *natural* gradient samples $f(\mathbf{z}) \cdot \tilde{\nabla}_\theta \log \pi(\mathbf{z} \,|\, \theta)$, resulting in the natural gradient (dashed red).

The covariance of the solid arrows on the right hand side turns out to be the inverse of the covariance of the solid arrows on the left. This has the effect that when computing the natural gradient, directions with high variance (uncertainty) are penalized and thus shrunken, while components with low variance (high certainty) are boosted, since these components of the gradient samples deserve more trust. This makes the (dashed red) natural gradient a much more trustworthy update direction than the (green) plain gradient.

The Fisher matrix can be estimated from samples, reusing the log-derivatives $\nabla_\theta \log \pi(\mathbf{z}|\theta)$ that we already computed for the gradient $\nabla_\theta J$. Then, updating the parameters following the natural gradient instead of the steepest gradient leads us to the general formulation of NES, as shown in Algorithm 3.

Algorithm 3: Canonical Natural Evolution Strategies

input: f, θ_{init}
repeat
  for $k = 1 \dots \lambda$ **do**
    draw sample $\mathbf{z}_k \sim \pi(\cdot|\theta)$
    evaluate the fitness $f(\mathbf{z}_k)$
    calculate log-derivatives $\nabla_\theta \log \pi(\mathbf{z}_k|\theta)$
  end
  $\nabla_\theta J \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_\theta \log \pi(\mathbf{z}_k|\theta) \cdot f(\mathbf{z}_k)$
  $\mathbf{F} \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_\theta \log \pi\left(\mathbf{z}_k|\theta\right) \nabla_\theta \log \pi\left(\mathbf{z}_k|\theta\right)^\top$
  $\theta \leftarrow \theta + \eta \cdot \mathbf{F}^{-1} \nabla_\theta J$
until *stopping criterion is met*

3. Performance and Robustness Techniques

In the following we will present and introduce crucial heuristics to improves NES's performance and robustness. Fitness shaping (Wierstra et al., 2008) is designed to make the algorithm invariant w.r.t. arbitrary yet order-preserving fitness transformations (Section 3.1). Adaptation sampling, a novel technique for adjusting learning rates online, is introduced in Section 3.2.

In sections 3.3 and 3.4 we describe two crucial techniques to enhance performance of the NES algorithm as applied to multinormal distributions: Exponential parameterization guarantees that the covariance matrix stays positive-definite, and second, a novel method for changing the coordinate system into a "natural" one is laid out, which makes the algorithm computationally efficient.

3.1 Fitness Shaping

NES uses rank-based fitness shaping in order to render the algorithm *invariant* under monotonically increasing (i.e., rank preserving) transformations of the fitness function. For this purpose, the fitness of the population is transformed into a set of utility values $u_1 \geq \cdots \geq u_\lambda$. Let $\mathbf{z}_i$ denote the i^{th} best individual (the i^{th} individual in the population, sorted by fitness, such that $\mathbf{z}_1$ is the best and $\mathbf{z}_\lambda$ the worst individual). Replacing fitness with utility, the gradient estimate of Equation (2) becomes, with slight abuse of notation,

$$\nabla_\theta J(\theta) = \sum_{k=1}^{\lambda} u_k \, \nabla_\theta \log \pi(\mathbf{z}_k \,|\, \theta).$$

The choice of utility function can in fact be seen as a free parameter of the algorithm. Throughout this paper we will use the following

$$u_k = \frac{\max\left(0, \log(\frac{\lambda}{2}+1) - \log(k)\right)}{\sum_{j=1}^{\lambda} \max\left(0, \log(\frac{\lambda}{2}+1) - \log(j)\right)} - \frac{1}{\lambda},$$

which is directly related to the one employed by CMA-ES (Hansen and Ostermeier, 2001), for ease of comparison. In our experience, however, this choice has not been crucial to performance, as long as it is monotonous and based on ranks instead of raw fitness (e.g., a function which simply increases linearly with rank).

3.2 Adaptation Sampling

To reduce the burden of determining appropriate hyper-parameters such as the learning rate, we develop a new online adaptation or meta-learning technique (Schaul and Schmidhuber, 2010), called *adaptation sampling*, that can automatically adapt the settings.

We model this situation as follows: Let π_θ be a distribution with hyper-parameter θ and $\psi(\mathbf{z})$ a quality measure for each sample $\mathbf{z} \sim \pi_\theta$. Our goal is to adapt θ such as to maximize the quality ψ. A straightforward method to achieve this, henceforth dubbed *adaptation sampling*, is to evaluate the quality of the samples $\mathbf{z}'$ drawn from $\pi_{\theta'}$, where $\theta' \neq \theta$ is a slight variation of θ, and then perform hill-climbing: Continue with the new θ' if the quality of its samples is significantly better (according, for example, to a Mann-Whitney U-test), and revert to θ otherwise. Note that this proceeding is similar to the NES algorithm itself, but applied at a meta-level to algorithm parameters instead of the search distribution. The goal of this adaptation is to maximize the *pace* of progress over time, which is slightly different from maximizing the fitness function itself.

Virtual adaptation sampling is a lightweight alternative to adaptation sampling that is particularly useful whenever evaluating ψ is expensive :

- do importance sampling on the existing samples $\mathbf{z}_i$, according to $\pi_{\theta'}$:

  $$w_i' = \frac{\pi(\mathbf{z}|\theta')}{\pi(\mathbf{z}|\theta)}$$

  (this is always well-defined, because $\mathbf{z} \sim \pi_\theta \Rightarrow \pi(\mathbf{z}|\theta) > 0$).

- compare $\{\psi(\mathbf{z}_i)\}$ with weights $\{w_i = 1, \forall i\}$ and $\{\psi' = \psi(\mathbf{z}_i), \forall i\}$ with weights $\{w_i'\}$, using a weighted generalization of the Mann-Whitney test.

Beyond determining whether θ or θ' is better, choosing a non-trivial confidence level ρ allows us to avoid parameter drift, as θ is only updated if the improvement is significant enough. There is one caveat, however: the rate of parameter change needs to be adjusted such that the two resulting distributions are not too similar (otherwise the difference won't be statistically significant), but also not too different, (otherwise the weights w' will be too small and again the test will be inconclusive). If, however, we explicitly desire large adaptation steps on θ, we have the possibility of interpolating between adaptation sampling and virtual adaptation sampling by drawing a few new samples from the distribution $\pi_{\theta'}$ (each assigned weight 1), where it is overlapping least with π_θ.

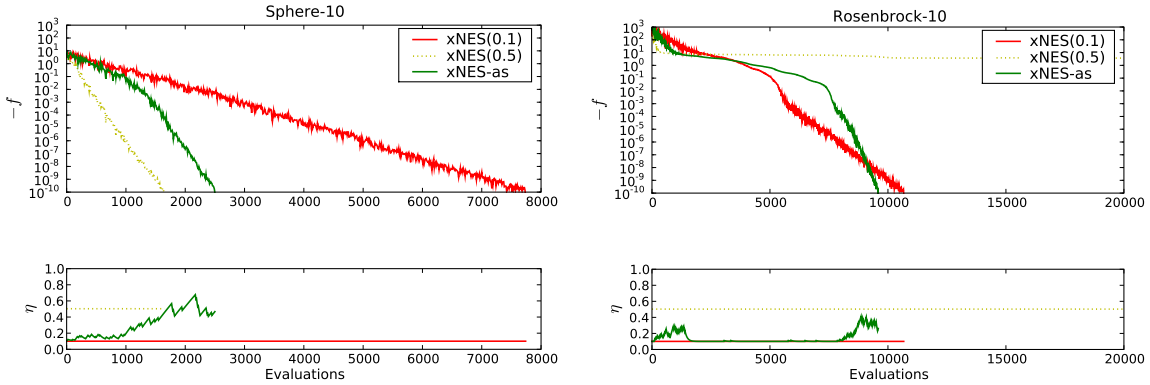


Figure 3: Illustration of the effect of adaptation sampling. We show the increase in fitness during a NES run (above) and the corresponding learning rates (below) on two setups: 10-dimensional sphere function (left), and 10-dimensional Rosenbrock function (right). Plotted are three variants of xNES (Algorithm 5): fixed default learning rate of $\eta = 0.1$ (dashed, red) fixed large learning rate of $\eta = 0.5$ (dotted, yellow), and an adaptive learning rate starting at $\eta = 0.1$ (green). We see that for the (simple) Sphere function, it is advantageous to use a large learning rate, and adaptation sampling automatically finds that one. However, using the overly greedy updates of a large learning rate fails on harder problems (right). Here adaptation sampling really shines: it boosts the learning rate in the initial phase (entering the Rosenbrock valley), then quickly reduces it while the search needs to carefully navigate the bottom of the valley, and boosts it again at the end when it has located the optimum and merely needs to zoom in precisely.

For NES algorithms, the most important parameter to be adapted by adaptation sampling is the learning rate η, starting with a conservative guess. This is because half-way into the search, after a local attractor has been singled out, it may well pay off to increase the learning rate in order to more quickly converge to it.

In order to produce variations η' which can be judged using the above-mentioned U-test, we propose a procedure similar in spirit to Rprop-updates (Riedmiller and Braun, 1993; Igel and Hüsken, 2003), where the learning rates are either increased or decreased by a multiplicative constant whenever there is evidence that such a change will lead to better samples.

More concretely, when using adaptation sampling for NES we test for an improvement with the hypothetical distribution θ' generated with $\eta' = 1.5\eta$. Each time the statistical test is successful with a confidence of at least $\rho = \frac{1}{2} - \frac{1}{3(d+1)}$ (this value was determined empirically) we increase the learning rate by a factor of $1+c'$, up to at most $\eta = 1$. Otherwise we bring it closer to its initial value: $\eta \leftarrow (1 - c')\eta + c'\eta_{init}$. We use $c' = \frac{1}{10}$ (again, an empirically robust choice). The final procedure is summarized in algorithm 4. We append the ending "-as" to denote algorithm variants using adaptation sampling.

Figure 3 illustrates the effect of the virtual adaptation sampling strategy on two different 10-dimensional unimodal benchmark functions, the Sphere function f_1 and the Rosenbrock function f_8 (see Section 5.2 for details). We find that, indeed, adaptation sampling boosts the learning rates to the appropriate high values when quick progress can be made (in the presence of an approximately quadratic optimum), but keeps them at carefully low values otherwise.

Algorithm 4: Adaptation sampling

input : $\eta_{\sigma,t}$,$\eta_{\sigma,\text{init}}$, θ_t, θ_{t-1}, $\{(\mathbf{z}_k, f(\mathbf{z}_k))\}$, c', ρ
output: $\eta_{\sigma,t+1}$
compute hypothetical θ', given θ_{t-1} and using $3/2\eta_{\sigma,t}$
for $k = 1 \dots \lambda$ **do**
  $w'_k = \dfrac{\pi(\mathbf{z}_k|\theta')}{\pi(\mathbf{z}_k|\theta)}$
end
$S \leftarrow \{\text{rank}(\mathbf{z}_k)\}$
$S' \leftarrow \{w'_k \cdot \text{rank}(\mathbf{z}_k)\}$
if *weighted-Mann-Whitney*$(S, S') < \rho$ **then**
  return $(1 - c') \cdot \eta_\sigma + c' \cdot \eta_{\sigma,\text{init}}$
else
  return $\min((1 + c') \cdot \eta_\sigma, 1)$
end

3.3 Rotationally Symmetric Distributions

In this section we derive the computation of the natural gradient for a rather general class of distributions, namely multi-variate distributions arising from linear transformations of rotationally symmetric distributions. For many important cases, such as multi-variate Gaussians, this allows us to obtain the Fisher matrix in closed form. The resulting strategy updates are computationally efficient.

For a sample $\mathbf{z} \in \mathbb{R}^d$ let $r = \|\mathbf{z}\|$ denote its radial component. Let $Q_{\boldsymbol{\tau}}(\mathbf{z})$ be a family of rotationally symmetric distributions with parameter vector $\boldsymbol{\tau}$. From this invariance property we deduce that the density can be written as $Q_{\boldsymbol{\tau}}(\mathbf{z}) = q_{\boldsymbol{\tau}}(r^2)$ for some family of functions $q_{\boldsymbol{\tau}} : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$. In the following we consider classes of search distributions with densities

$$\begin{aligned}\pi\big(\mathbf{z} \,\big|\, \boldsymbol{\mu}, \mathbf{A}, \boldsymbol{\tau}\big) &= \frac{1}{|\det(\mathbf{A})|} \cdot q_{\boldsymbol{\tau}}\big(\| \left(\mathbf{A}^{-1}\right)^\top (\mathbf{z} - \boldsymbol{\mu})\|^2\big) \\ &= \frac{1}{\sqrt{\det(\mathbf{A}^\top \mathbf{A})}} \cdot q_{\boldsymbol{\tau}}\big((\mathbf{z} - \boldsymbol{\mu})^\top (\mathbf{A}^\top \mathbf{A})^{-1} (\mathbf{z} - \boldsymbol{\mu})\big) \qquad (6)\end{aligned}$$

with additional transformation parameters $\boldsymbol{\mu} \in \mathbb{R}^d$ and invertible $\mathbf{A} \in \mathbb{R}^{d \times d}$. If needed, $\mathbf{A}$ can be restricted to any continuous sub-group of the invertible matrices like, for example, diagonal matrices. The function $q_{\boldsymbol{\tau}}$ is the accordingly transformed density of the random variable $\mathbf{s} = \left(\mathbf{A}^{-1}\right)^\top (\mathbf{z} - \boldsymbol{\mu})$. This setting is rather general. It covers many impor-

tant families of distributions and their multi-variate forms, most prominently multi-variate Gaussians. In addition, properties of the radial distribution such as its tail (controlling whether large mutations are common or rare) can be controlled with the parameter $\boldsymbol{\tau}$.

3.3.1 Local "Natural" Coordinates

In principle the computation of the natural gradient, involving the computation of gradient and Fisher matrix, is straight-forward. However, the parameters $\boldsymbol{\mu}$ and $\mathbf{A}$ have d and $d(d+1)/2$ dimensions, which makes a total of $d(d+3)/2 \in \mathcal{O}(d^2)$. Let d' denote the dimensionality of the radial parameters $\boldsymbol{\tau}$, and for simplicity we assume that d' is fixed and does not grow with the dimensionality of the search space. Then the Fisher matrix has $\mathcal{O}(d^4)$ entries, and its inversion costs $\mathcal{O}(d^6)$ operations. It turns out that we can do much better. The following derivation is a generalization of the proceeding found in Glasmachers et al. (2010b).

The above encoding by means of transformations of a rotation invariant normal form of the distribution hints at the introduction of a canonical local coordinate system in which the normal form becomes the current search distribution. It turns out from Equation (6) that the dependency of the distribution on $\mathbf{A}$ is only in terms of the symmetric positive definite matrix $\mathbf{A}^\top\mathbf{A}$. In the Gaussian case this matrix coincides with the covariance matrix. Instead of performing natural gradient steps on the manifolds of invertible or positive definite symmetric matrices we introduce a one-to-one encoding with a vector space representation. The matrix exponential, restricted to the vector space of symmetric matrices, is a global map for the manifold of symmetric positive definite matrices. Thus, we introduce "exponential" local coordinates $(\boldsymbol{\delta}, \mathbf{M}) \mapsto (\boldsymbol{\mu}_{\text{new}}, \mathbf{A}_{\text{new}}) = \left(\boldsymbol{\mu} + \mathbf{A}^\top\boldsymbol{\delta}, \mathbf{A}\exp\left(\frac{1}{2}\mathbf{M}\right)\right)$. These coordinates are local in the sense that the current search distribution is encoded by $(\boldsymbol{\delta}, \mathbf{M}) = (0, 0)$. It turns out that in these coordinates the Fisher matrix takes the rather simple form

$$\mathbf{F} = \begin{pmatrix} \mathbb{I} & v \\ v^\top & c \end{pmatrix} \quad \text{with} \quad v = \frac{\partial^2 \log \pi(\mathbf{z})}{\partial(\boldsymbol{\delta}, \mathbf{M})\partial\boldsymbol{\tau}} \in \mathbb{R}^{(m-d')\times d'} \quad \text{and} \quad c = \frac{\partial^2 \log \pi(\mathbf{z})}{\partial\boldsymbol{\tau}^2} \in \mathbb{R}^{d'\times d'}. \tag{7}$$

Note that for distributions without radial parameters $\boldsymbol{\tau}$, such as Gaussians, we obtain $\mathbf{F} = \mathbb{I}$. Thus, in local coordinates the otherwise computationally intensive operations of computing and inverting the Fisher matrix are trivial, and the vanilla gradient coincides with the natural gradient. For this reason we call the above local coordinates also *natural* exponential coordinates. For non-trivial parameters $\boldsymbol{\tau}$ we use the Woodbury identity to compute the inverse of the Fisher matrix as

$$\mathbf{F}^{-1} = \begin{pmatrix} \mathbb{I} & v \\ v^\top & c \end{pmatrix}^{-1} = \begin{pmatrix} \mathbb{I} + Hvv^\top & -Hv \\ -Hv^\top & H \end{pmatrix}$$

with $H = (c - v^\top v)^{-1}$, and exploiting $H^\top = H$. It remains to compute the gradient. We obtain the three components of derivatives of log-probabilities

$$\nabla_{\boldsymbol{\delta},\mathbf{M},\boldsymbol{\tau}}|_{\boldsymbol{\delta}=0,\mathbf{M}=0} \log \pi\left(\mathbf{z} \,|\, \boldsymbol{\mu}, \mathbf{A}, \boldsymbol{\tau}, \boldsymbol{\delta}, \mathbf{M}\right) = g = \left(g_{\boldsymbol{\delta}}, g_{\mathbf{M}}, g_{\boldsymbol{\tau}}\right),$$

$$
\begin{aligned}
g_{\boldsymbol{\delta}} =& -2 \cdot \frac{q'_{\boldsymbol{\tau}}(\|\mathbf{s}\|^2)}{q_{\boldsymbol{\tau}}(\|\mathbf{s}\|^2)} \cdot \mathbf{s}, \\
g_{\mathbf{M}} =& -\frac{1}{2}\mathbb{I} - \frac{q'_{\boldsymbol{\tau}}(\|\mathbf{s}\|^2)}{q_{\boldsymbol{\tau}}(\|\mathbf{s}\|^2)} \cdot \mathbf{s}\mathbf{s}^\top, \\
g_{\boldsymbol{\tau}} =& \frac{1}{q_{\boldsymbol{\tau}}(\|\mathbf{s}\|^2)} \cdot \nabla_{\boldsymbol{\tau}}\, q_{\boldsymbol{\tau}}(\|\mathbf{s}\|^2),
\end{aligned}
$$

where $q'_{\boldsymbol{\tau}} = \frac{\partial}{\partial(r^2)} q_{\boldsymbol{\tau}}$ denotes the derivative of $q_{\boldsymbol{\tau}}$ with respect to r^2, and $\nabla_{\boldsymbol{\tau}}\, q_{\boldsymbol{\tau}}$ denotes the gradient w.r.t. $\boldsymbol{\tau}$. The sample-wise natural gradient becomes

$$
\mathbf{F}^{-1} \cdot g = \begin{pmatrix} (g_{\boldsymbol{\delta}}, g_{\mathbf{M}}) - Hv(v^\top (g_{\boldsymbol{\delta}}, g_{\mathbf{M}}) - g_{\boldsymbol{\tau}}) \\ H(v^\top (g_{\boldsymbol{\delta}}, g_{\mathbf{M}}) - g_{\boldsymbol{\tau}}) \end{pmatrix},
$$

which can be computed efficiently in only $\mathcal{O}(d^2)$ operations (assuming fixed d'). This is in contrast to $\mathcal{O}(d^6)$ operations required for a naïve inversion of the full Fisher matrix.

3.3.2 Sampling from Radial Distributions

In order to use this class of distributions for search we need to be able to draw samples from it. The central idea is to first draw a sample $\mathbf{s}$ from the 'standard' density $\pi(\mathbf{s} \,|\, \boldsymbol{\mu} = 0, \mathbf{A} = \mathbb{I}, \boldsymbol{\tau})$, which is then transformed into the sample $\mathbf{z} = \mathbf{A}^\top \mathbf{s} + \boldsymbol{\mu}$, corresponding to the density $\pi(\mathbf{z} \,|\, \mathbf{A}, \boldsymbol{\mu}, \boldsymbol{\tau})$. In general, sampling $\mathbf{s}$ can be decomposed into sampling the (squared) radius component $r^2 = \|\mathbf{z}\|^2$ and a unit vector $\mathbf{v} \in \mathbb{R}^d$, $\|\mathbf{v}\| = 1$. The squared radius has the density

$$
\tilde{q}_{\boldsymbol{\tau}}(r^2) = \int\limits_{\|\mathbf{z}\|^2 = r^2} Q_{\boldsymbol{\tau}}(\mathbf{z})\, d\mathbf{z} = \frac{2\pi^{d/2}}{\Gamma(d/2)} \cdot (r^2)^{(d-1)/2} \cdot q_{\boldsymbol{\tau}}(r^2),
$$

where $\Gamma(\cdot)$ denotes the gamma function. In the following we assume that we have an efficient method of drawing samples from this one-dimensional density. Besides the radius we draw a unit vector $\mathbf{u} \in \mathbb{R}^d$ uniformly at random, for example by normalizing a standard normally distributed vector. Then $\mathbf{s} = r \cdot \mathbf{u}$ is effectively sampled from $\pi(\mathbf{s} \,|\, \boldsymbol{\mu} = 0, \mathbf{A} = \mathbb{I}, \boldsymbol{\tau})$, and the composition $\mathbf{z} = r\mathbf{A}^\top \mathbf{u} + \boldsymbol{\mu}$ follows the density $\pi(\mathbf{z} \,|\, \boldsymbol{\mu}, \mathbf{A}, \boldsymbol{\tau})$. In many special cases, however, there are more efficient ways of sampling $\mathbf{s} = r \cdot \mathbf{u}$ directly.

3.4 Techniques for Multinormal Distributions

Multi-variate Gaussians are the most prominent class of search distributions for evolution strategies. An advantageous property of Gaussians is that the Fisher information matrix is known analytically. A large share of the previous work on NES has dealt with the development of efficient techniques for this important special case.

Here we deal with this prominent case in the above introduced framework. The natural gradient is

$$\nabla_{\boldsymbol{\delta}} J = \sum_{k=1}^{\lambda} f(\mathbf{z}_k) \cdot \mathbf{s}_k,$$
$$\nabla_{\mathbf{M}} J = \sum_{k=1}^{\lambda} f(\mathbf{z}_k) \cdot (\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I}),$$

where $\mathbf{s}_k$ is the k-th best sample in the batch in local coordinates, and $\mathbf{z}_k$ is the same sample in task coordinates. The resulting algorithm, outlined in Algorithm 5 is known as exponential NES (xNES). It is demonstrated in the algorithm how the covariance factor $\mathbf{A}$ can be decomposed into a scalar step size $\sigma > 0$ and a normalized covariance factor $\mathbf{B}$ fulfilling $\det(\mathbf{B}) = 1$. This decoupling of shape ($\mathbf{B}$) from scale (σ) information allows for adaptation of the two orthogonal components with independent learning rates. All NES variants for multi-variate Gaussians have a complexity of $\mathcal{O}(d^3)$ computations per covariance matrix update. This complexity can be reduced to $\mathcal{O}(d^2)$ by computing the updates in local non-exponential coordinates.

It was shown that the NES principle is also compatible with elitist selection (Glasmachers et al., 2010a), resulting in the natural gradient hillclimber (1+1)-xNES. We refer to the papers by Sun et al. (2009a,b) and Glasmachers et al. (2010b,a) for further technical details on these algorithms.

Algorithm 5: Exponential Natural Evolution Strategies (xNES) (multinormal case)

input: f, $\boldsymbol{\mu}_{init}$, $\boldsymbol{\Sigma}_{init} = \mathbf{A}^\top \mathbf{A}$

initialize
$\sigma \leftarrow \sqrt[d]{|\det(\mathbf{A})|}$
$\mathbf{B} \leftarrow \mathbf{A}/\sigma$

repeat
  for $k = 1 \dots \lambda$ **do**
    draw sample $\mathbf{s}_k \sim \mathcal{N}(0, \mathbb{I})$
    $\mathbf{z}_k \leftarrow \boldsymbol{\mu} + \sigma \mathbf{B}^\top \mathbf{s}_k$
    evaluate the fitness $f(\mathbf{z}_k)$
  end
  sort $\{(\mathbf{s}_k, \mathbf{z}_k)\}$ with respect to $f(\mathbf{z}_k)$ and compute utilities u_k

  compute gradients
  $\nabla_{\boldsymbol{\delta}} J \leftarrow \sum_{k=1}^{\lambda} u_k \cdot \mathbf{s}_k$ $\quad$ $\nabla_{\mathbf{M}} J \leftarrow \sum_{k=1}^{\lambda} u_k \cdot (\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I})$
  $\nabla_\sigma J \leftarrow \mathrm{tr}(\nabla_{\mathbf{M}} J)/d$ $\quad$ $\nabla_{\mathbf{B}} J \leftarrow \nabla_{\mathbf{M}} J - \nabla_\sigma J \cdot \mathbb{I}$

  update parameters
  $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \eta_{\boldsymbol{\delta}} \cdot \sigma \mathbf{B} \cdot \nabla_{\boldsymbol{\delta}} J$
  $\sigma \leftarrow \sigma \cdot \exp(\eta_\sigma / 2 \cdot \nabla_\sigma J)$
  $\mathbf{B} \leftarrow \mathbf{B} \cdot \exp(\eta_{\mathbf{B}} / 2 \cdot \nabla_{\mathbf{B}} J)$

until *stopping criterion is met*

3.5 Beyond Multinormal Distributions

The large share of literature on the multinormal case does not properly reflect the generality of the NES algorithm. It was shown by Schaul et al. (2011) that certain classes of problems can profit from tailored search distributions.

One simple yet important variant, inspired by Ros and Hansen (2008), is to use separable search distributions to improve the update complexity from cubic or quadratic to linear. This is compatible with the exponential parameterization of xNES. Cheap updates are a prerequisite for search in high-dimensional spaces, for example, for training recurrent neural networks. The resulting algorithm is called separable NES (SNES). Within our framework of linearly transformed rotationally symmetric distributions we obtain this case by restricting $\mathbf{A}$ to the group of (invertible) diagonal transformation matrices.

Another direction is the extension of NES to multivariate Cauchy distributions. These heavy-tailed distributions have undefined expectation and infinite variance. In this case the natural gradient is not defined. Still, the NES principle can be generalized by means of invariance properties. This is because three seemingly unrelated properties coincide for local natural coordinates. Let us assume the absence of radial parameters $\boldsymbol{\tau}$, then (i) due to Equation (7) the Fisher matrix is the identity, (ii) plain and natural gradient coincide, and (iii) by construction the current search distribution is invariant under orthogonal transformations. We obtain the following alternative characterization of NES: The NES algorithm performs its gradient updates based on a local coordinate system in which the orthogonal group (induced by the standard inner product) leaves the search distribution invariant.

This characterization for multi-variate distributions turns out to be robust. The argument stays valid while we iteratively grow the distribution's tail. In the limit of infinite variance the natural gradient interpretation breaks down, while the characterization by invariance is unaffected.

NES with heavy-tailed distributions are most useful as (1+1) hillclimbers: Assume a lucky exploratory sample from the distribution's heavy tail has managed to escape a bad local optimum. In a population-based algorithm this effect would be "corrupted" by weighted averaging and the better attractor may be missed, while a hillclimber can jump to the new position. It has been demonstrated in Schaul et al. (2011) that heavy-tailed distributions can improve the performance of NES on highly multi-modal problems.

4. Connection to CMA-ES

It turns out that xNES is closely related to the seminal Covariance Matrix Adaptation Evolution Strategy (CMA-ES; Hansen and Ostermeier, 2001) algorithm. It has been noticed by Glasmachers et al. (2010b) that in first order Taylor approximation the exponential xNES update coincides with the so-called rank-μ update of CMA-ES. Akimoto et al. (2010) have show rigorously that CMA-ES is in fact following an approximate natural gradient, and thus, arguably, can be seen as a member of the NES family. In the remainder of this section, we will point out similarities and differences between it and xNES.

The relation between CMA-ES and xNES is clearest when considering the CMA-ES variant with rank-μ update (in the terminology of this study, rank-λ-update), since this one does not feature evolution paths. Both xNES and CMA-ES parameterize the search distribution with three functionally different parameters for mean, scale, and shape of the

Algorithm 6: Separable NES (SNES)

input: f, $\boldsymbol{\mu}_{init}$, $\boldsymbol{\sigma}_{init}$
repeat
  for $k = 1 \dots \lambda$ **do**
    draw sample $\mathbf{s}_k \sim \mathcal{N}(0, \mathbb{I})$
    $\mathbf{z}_k \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma}\mathbf{s}_k$
    evaluate the fitness $f(\mathbf{z}_k)$
  end
  sort $\{(\mathbf{s}_k, \mathbf{z}_k)\}$ with respect to $f(\mathbf{z}_k)$ and compute utilities u_k
  compute gradients
  $$\nabla_{\boldsymbol{\mu}} J \leftarrow \sum_{k=1}^{\lambda} u_k \cdot \mathbf{s}_k$$
  $$\nabla_{\boldsymbol{\sigma}} J \leftarrow \sum_{k=1}^{\lambda} u_k \cdot (\mathbf{s}_k^2 - 1)$$
  update parameters
  $$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \eta_{\boldsymbol{\mu}} \cdot \boldsymbol{\sigma} \cdot \nabla_{\boldsymbol{\mu}} J$$
  $$\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} \cdot \exp(\eta_{\boldsymbol{\sigma}}/2 \cdot \nabla_{\boldsymbol{\sigma}} J)$$
until *stopping criterion is met*;

distribution. xNES uses the parameters $\boldsymbol{\mu}$, σ, and $\mathbf{B}$, while the covariance matrix is represented as $\sigma^2 \cdot \mathbf{C}$ in CMA-ES, where $\mathbf{C}$ can be any positive definite symmetric matrix. Thus, the representation of the scale of the search distribution is shared among σ and $\mathbf{C}$ in CMA-ES, and the role of the additional parameter σ is to allow for an adaptation of the step size on a faster time scale than the full covariance update. In contrast, the NES updates of scale and shape parameters σ and $\mathbf{B}$ are decoupled.

The update of the center parameter $\boldsymbol{\mu}$ is very similar to the update of the center of the search distribution in CMA-ES, see Hansen and Ostermeier (2001). The utility function exactly takes the role of the weights in CMA-ES, which assumes a fixed learning rate of one.

For the covariance matrix, the situation is more complicated. We deduce the update rule

$$\begin{aligned}\boldsymbol{\Sigma}_{\text{new}} =& (\mathbf{A}_{\text{new}})^\top \cdot \mathbf{A}_{\text{new}} \\ =& \mathbf{A}^\top \cdot \exp\left(\eta_{\boldsymbol{\Sigma}} \cdot \sum_{k=1}^{\lambda} u_k \left(\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I}\right)\right) \cdot \mathbf{A}\end{aligned}$$

for the covariance matrix, with learning rate $\eta_{\boldsymbol{\Sigma}} = \eta_{\mathbf{A}}$. This term is closely connected to the exponential parameterization of the natural coordinates in xNES, while CMA-ES is formulated in global linear coordinates. The connection of these updates can be shown either by applying the xNES update directly to the natural coordinates without the exponential parameterization (Akimoto et al., 2010), or by approximating the exponential map by its first order Taylor expansion. Akimoto et al. (2010) established the same connection directly in coordinates based on the Cholesky decomposition of $\boldsymbol{\Sigma}$, see Sun et al. (2009a,b). The arguably simplest derivation of the equivalence relies on the invariance of the natural gradient under coordinate transformations, which allows us to perform the computation, w.l.o.g., in natural coordinates. We use the first order Taylor approximation of the matrix

exponential to obtain

$$\exp\left(\eta_{\boldsymbol{\Sigma}} \cdot \sum_{k=1}^{\lambda} u_k \left(\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I}\right)\right) \approx \mathbb{I} + \eta_{\boldsymbol{\Sigma}} \cdot \sum_{k=1}^{\lambda} u_k \left(\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I}\right),$$

so the first order approximate update yields

$$\begin{aligned}
\boldsymbol{\Sigma}'_{new} =& \mathbf{A}^\top \cdot \left(\mathbb{I} + \eta_{\boldsymbol{\Sigma}} \cdot \sum_{k=1}^{\lambda} u_k \left(\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I}\right)\right) \cdot \mathbf{A} \\
=& (1 - U \cdot \eta_{\boldsymbol{\Sigma}}) \cdot \mathbf{A}^\top \mathbf{A} + \eta_{\boldsymbol{\Sigma}} \cdot \sum_{k=1}^{\lambda} u_k \left(\mathbf{A}^\top \mathbf{s}_k\right) \left(\mathbf{A}^\top \mathbf{s}_k\right)^\top \\
=& (1 - U \cdot \eta_{\boldsymbol{\Sigma}}) \cdot \boldsymbol{\Sigma} + \eta_{\boldsymbol{\Sigma}} \cdot \sum_{k=1}^{\lambda} u_k \left(\mathbf{z}_k - \boldsymbol{\mu}\right) \left(\mathbf{z}_k - \boldsymbol{\mu}\right)^\top
\end{aligned}$$

with $U = \sum_{k=1}^{\lambda} u_k$, from which the connection to the CMA-ES rank-μ-update is obvious (see Hansen and Ostermeier, 2001, and note that $U = 1$ for CMA-ES.).

It is interesting to note that both xNES and CMA-ES use different learning rates for the mean and covariance components of the search distribution. Thus, in a strict sense they do not follow the natural gradient. Instead they follow a systematically transformed direction with altered (typically reduced) covariance component. This makes intuitive sense since the d components of the mean can be estimated more robustly from a fixed size sample than the $\mathcal{O}(d^2)$ covariance parameters. The theoretical implications of using differently scaled updates for the two components are yet to be explored.

CMA-ES uses the well-established technique of evolution paths to smooth out random effects over multiple generations. This technique is particularly valuable when working with minimal population sizes, which is the default for both algorithms. Thus, evolution paths are expected to improve stability; further interpretations have been provided by Hansen and Ostermeier (2001). However, the presence of evolution paths complicates matters since the state of the CMA-ES algorithms is not completely described by its search distribution. Another difference between xNES and CMA-ES is the exponential parameterization of the updates in xNES, which results in a multiplicative update equation for the covariance matrix, in contrast to the additive update of CMA-ES. The multiplicative covariance update is coherent with the multiplicative (and also exponential) update of the step size σ.

A valuable perspective offered by the natural gradient updates in xNES is the derivation of the updates of the center $\boldsymbol{\mu}$, the step size σ, and the normalized transformation matrix $\mathbf{B}$, all from the *same* principle of natural gradient ascent. In contrast, the updates applied in CMA-ES result from different heuristics for each parameter. This connection might provide an interesting perspective on some of the methods employed by CMA-ES.

5. Experiments

In this section, we empirically validate the new algorithms, to determine how NES algorithms perform compared to state-of-the-art evolution strategies, identifying specific strengths and limitations of the different variants.

We conduct a broad series of experiments on standard benchmarks, as well as more specific experiments testing special capabilities. In total, four different algorithm variants are tested and their behaviors compared qualitatively as well as quantitatively, w.r.t. different modalities.

We start by detailing and justifying the choices of hyperparameters, then we proceed to evaluate the performance of a number of different variants of NES (with and without adaptation sampling) on a broad collection of benchmarks. We also conduct experiments using the separable variant on high-dimensional problems.

5.1 Experimental Setup and Hyperparameters

Across all NES variants, we distinguish three hyperparameters: the population size λ, the learning rates η and the utility function u (because we always use fitness shaping, see Section 3.1). In particular, for the multivariate Gaussian case (xNES) we have the three learning rates $\eta_{\boldsymbol{\mu}}$, η_{σ}, and $\eta_{\mathbf{B}}$.

It is highly desirable to have good default settings that scale with the problem dimension and lead to robust performance on a broad class of benchmark functions. Table 1 provides such default values as functions of the problem dimension d for xNES. We borrowed several of the settings from CMA-ES (Hansen and Ostermeier, 2001), which seems natural due to the apparent similarity. Both the population size λ and the learning rate $\eta_{\boldsymbol{\mu}}$ are the same as for CMA-ES, even if this learning rate never explicitly appears in CMA-ES. For the utility function we copied the weighting scheme of CMA-ES, but we shifted the values such that they sum to zero, which is the simplest form of implementing a fitness baseline; Jastrebski and Arnold (2006) proposed a similar approach for CMA-ES. The remaining parameters were determined via an empirical investigation, aiming for robust performance. In addition, in the separable case (SNES) the number of parameters in the covariance matrix is reduced from $d(d+1)/2 \in \mathcal{O}(d^2)$ to $d \in \mathcal{O}(d)$, which allows us to increase the learning rate $\eta_{\boldsymbol{\sigma}}$ by a factor of $d/3 \in \mathcal{O}(d)$, a choice which has proven robust in practice (Ros and Hansen, 2008).

The algorithm variants that we will be evaluating below are xNES (Algorithm 5), "xNES-as", that is xNES using adaptation sampling (Section 3.2), and the separable SNES (Algorithm 6). A Python implementation of all these is available within the open-source machine learning library PyBrain (Schaul et al., 2010), and implementations in different languages can be found at `http://www.idsia.ch/~tom/nes.html`.

5.2 Black-box Optimization Benchmarks

For a practitioner it is important to understand how NES algorithms compare to other methods on a wide range black-box optimization scenarios. Thus, we evaluate our algorithm on all the benchmark functions of the 'Black-Box Optimization Benchmarking' collection (BBOB) from the GECCO Workshop for Real-Parameter Optimization. The collection consists of 24 noise-free functions (12 unimodal, 12 multimodal; Hansen et al., 2010a) and 30 noisy functions (Hansen et al., 2010b). In order to make our results fully comparable, we also use the identical setup (Hansen and Auger, 2010), which transforms the pure benchmark functions to make the parameters non-separable (for some) and avoid trivial optima at the origin. The framework permits restarts until the budget of function evaluations ($10^5 d$) is

parameter	default value
λ	$4 + \lfloor 3\log(d) \rfloor$
$\eta_{\boldsymbol{\mu}}$	1
$\eta_\sigma = \eta_{\mathbf{B}}$	$\frac{(9 + 3\log(d))}{5d\sqrt{d}}$
$\eta_{\boldsymbol{\sigma}}$	$\frac{(3 + \log(d))}{5\sqrt{d}}$
u_k	$\frac{\max\left(0, \log(\frac{\lambda}{2} + 1) - \log(i)\right)}{\sum_{j=1}^{\lambda} \max\left(0, \log(\frac{\lambda}{2} + 1) - \log(j)\right)} - \frac{1}{\lambda}$
c'	$\frac{1}{10}$

Table 1: Default parameter values for xNES, xNES-as and SNES (including the utility function) as a function of problem dimension d.

used up, which we trigger whenever the variance of the distribution becomes too small, that is, when $\sqrt[d]{\det \mathbf{\Sigma}} < 10^{-20}$.

The results in this subsection are very similar[2] to those published in the 2012 edition of the BBOB Workshop at GECCO, we refer the interested reader to Schaul (2012b,c,e,d) for additional results and analysis. Figure 4 provides a compounded overview of the results on dimensions 5 and 20, on noisy or noiseless functions, and a direct comparison to all algorithms benchmarked in the 2009 edition of BBOB, which shows that xNES is among the best algorithms, assuming that the budget of function evaluations surpasses 100 times the dimension. We find (Schaul, 2012c) that xNES-as significantly outperforms all algorithms from the BBOB 2009 competition on function f_{115} and for limited budgets on f_{18}, f_{118} and f_{119}, while underperforming compared to the winners on a set of other functions.

Figures 5 (noise-free functions) and Figure 6 (noisy functions) show how performance scales with dimension, on a detailed function-by-function level, for both xNES and xNES-as. Using adaptation sampling improves performance most significantly on simple benchmarks like the sphere function or its noisy siblings (f_1, f_{101}, f_{102}, f_{103}, f_{107}) and only hurts significantly on f_{13}, in high dimensions (see Schaul, 2012e for the full comparison, and result tables with statistical significance tests). While the presented default settings are weak for highly multi-modal functions like f_3, f_4 or f_{15}, larger population sizes and sophisticated restart strategies may alleviate this issue.

Figure 7 compares the loss ratios (in terms of expected running time) given fixed budgets of evaluations, for xNES, xNES-as, and BIPOP-CMA-ES (Hansen, 2009a,b), the winner of the 2009 BBOB edition as reference point. BIPOP-CMA-ES is significantly better on most noisy functions, but the differences are much more subtle on the noiseless ones. In fact, a

2. The difference lies with the stopping criterion used for restarts, which was not compliant in the 2012 results.

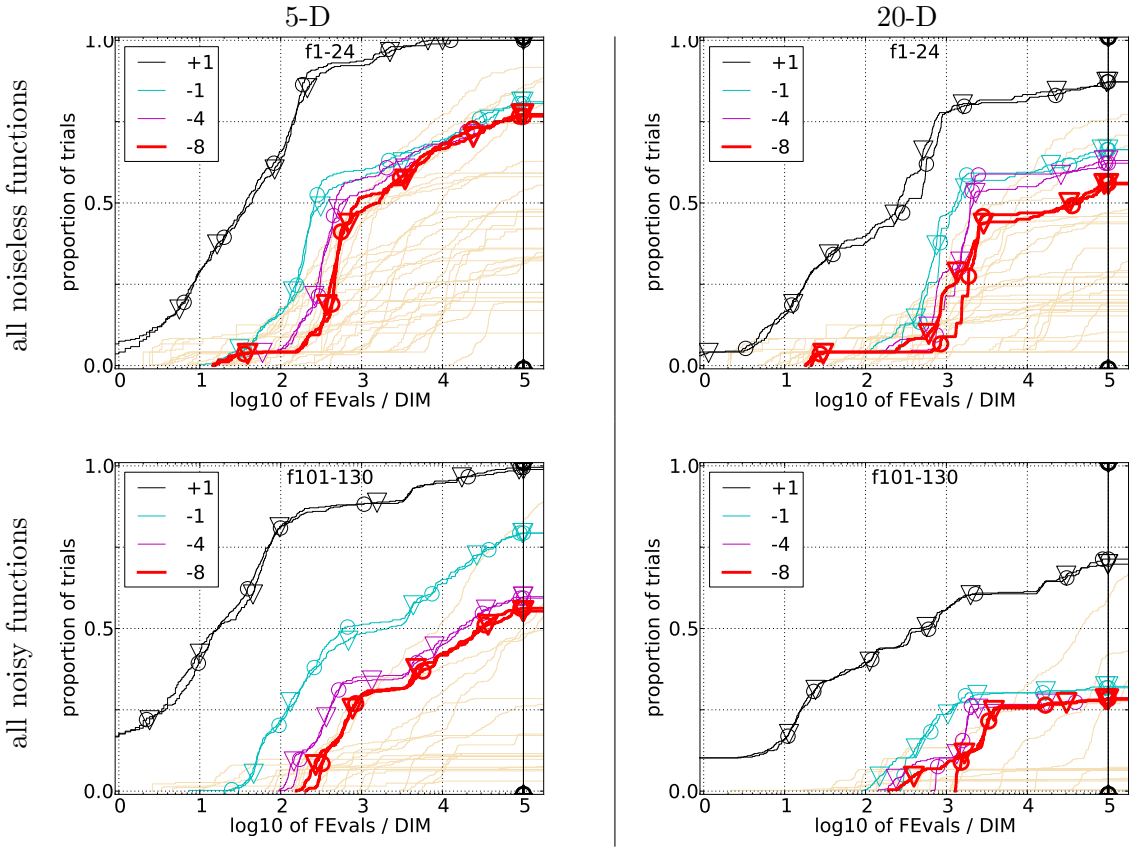


Figure 4: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right), over all noiseless functions (top row) and all noisy functions (bottom row). The ECDF is taken of the number of function evaluations divided by dimension d to reach a target value $f_{\text{opt}} + 10^k$, where $k \in \{1, -1, -4, -8\}$ is given by the first value in the legend, for xNES ($\circ$) and xNES_as ($\triangledown$). Light beige lines show the ECDF for target precision 10^{-8} of all algorithms benchmarked during BBOB-2009. From this high-level perspective, both xNES variants appear among the best tested algorithms, with a small advantage for using adaptation sampling.

direct comparison (Schaul, 2012d) found that xNES-as is close in performance to BIPOP-CMA-ES across a large fraction of the benchmark functions; but there is some diversity as well, with xNES-as being significantly better on 6 of the functions and significantly worse on 18 of them.

5.3 Separable NES for Neuroevolution

The SNES algorithm is expected to perform at least as well as xNES on separable problems, while it should show considerably worse performance in the presence of highly dependent variables. These are indeed the the findings in Schaul (2012a), where SNES was bench-

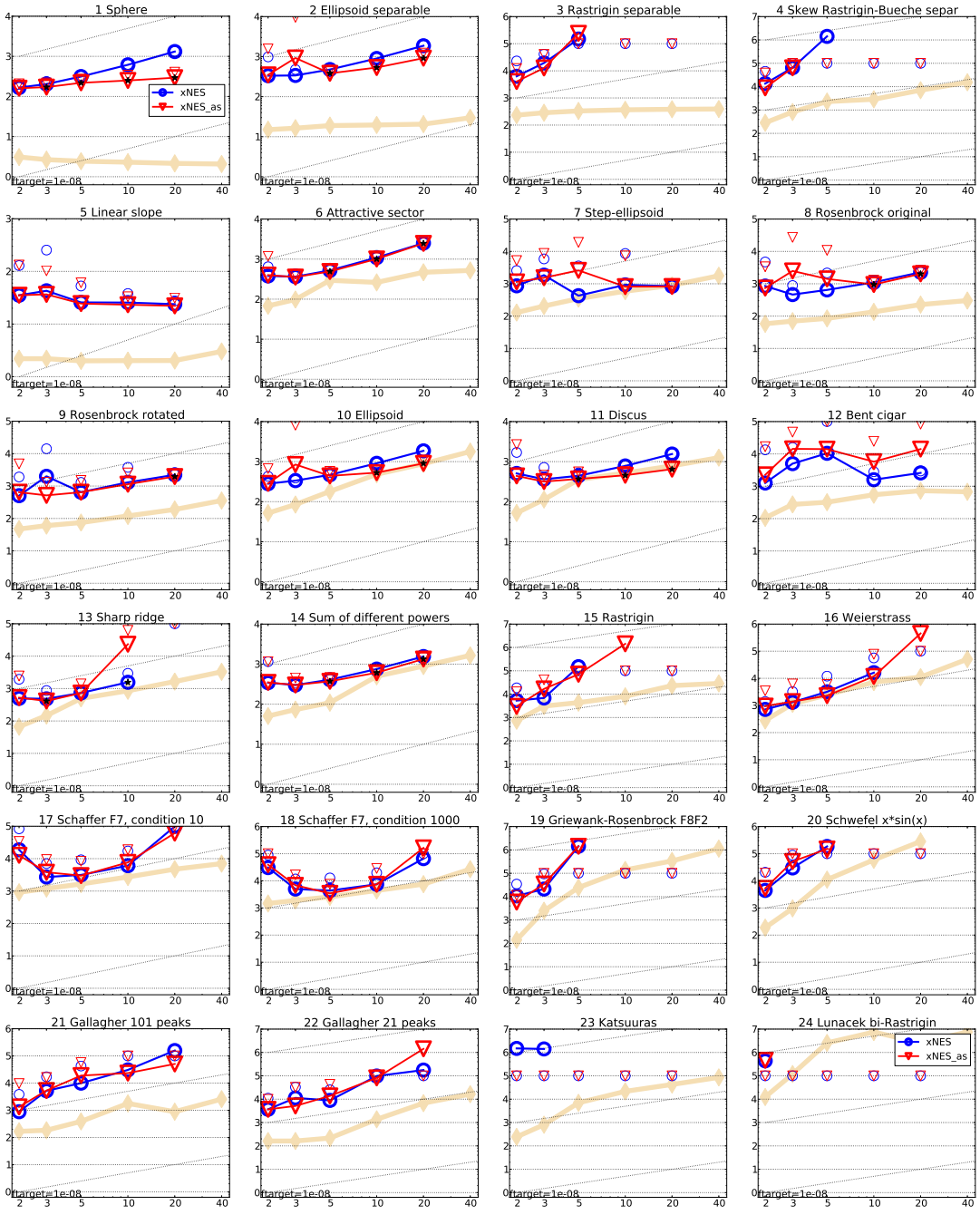


Figure 5: Expected running time (ERT, as $\log_{10}$ value of the number of f-evaluations divided by dimension) on all noise-free functions (f_1 and f_{24}) for target precision 10^{-8}, versus dimension on the horizontal axis. Blue circles $\circ$ refer to xNES, red triangles $\triangledown$ refer to xNES_as, and the light beige lines show the performance of the best-performing algorithm from the BBOB-2009 entrants (the best, individually for each dimension-function pair). Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Black stars indicate statistically better result compared to all other algorithms with $p < 0.01$.

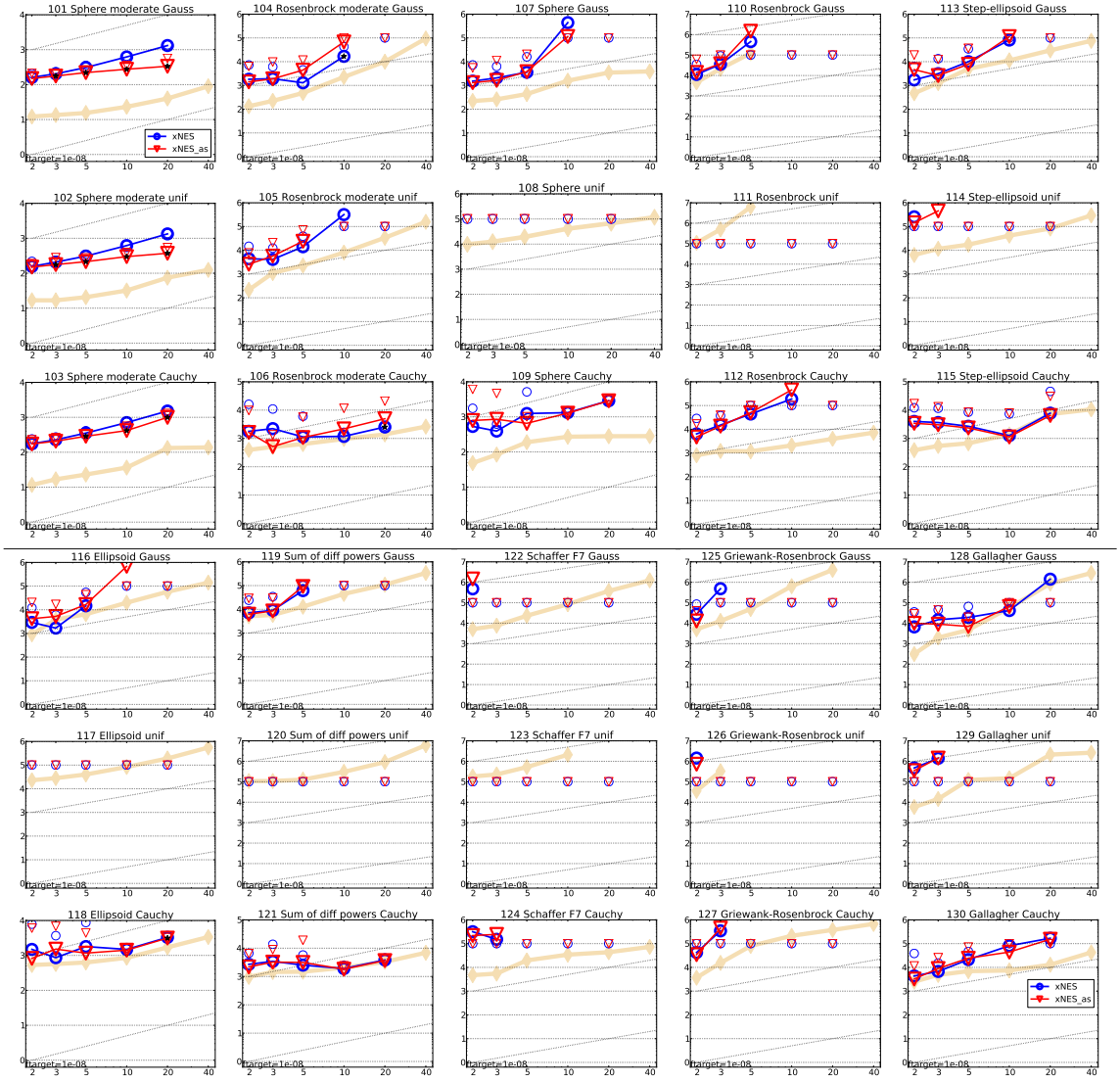


Figure 6: Expected running time (ERT in number of f-evaluations, divided by dimension) on all noisy functions (f_{101} and f_{130}) for target precision 10^{-8}, versus dimension (see Figure 5 for details). We observe that, despite using small population sizes and the same parameter settings than on the noise-free benchmarks, xNES achieves state-of-the art performance on a subset of the functions.

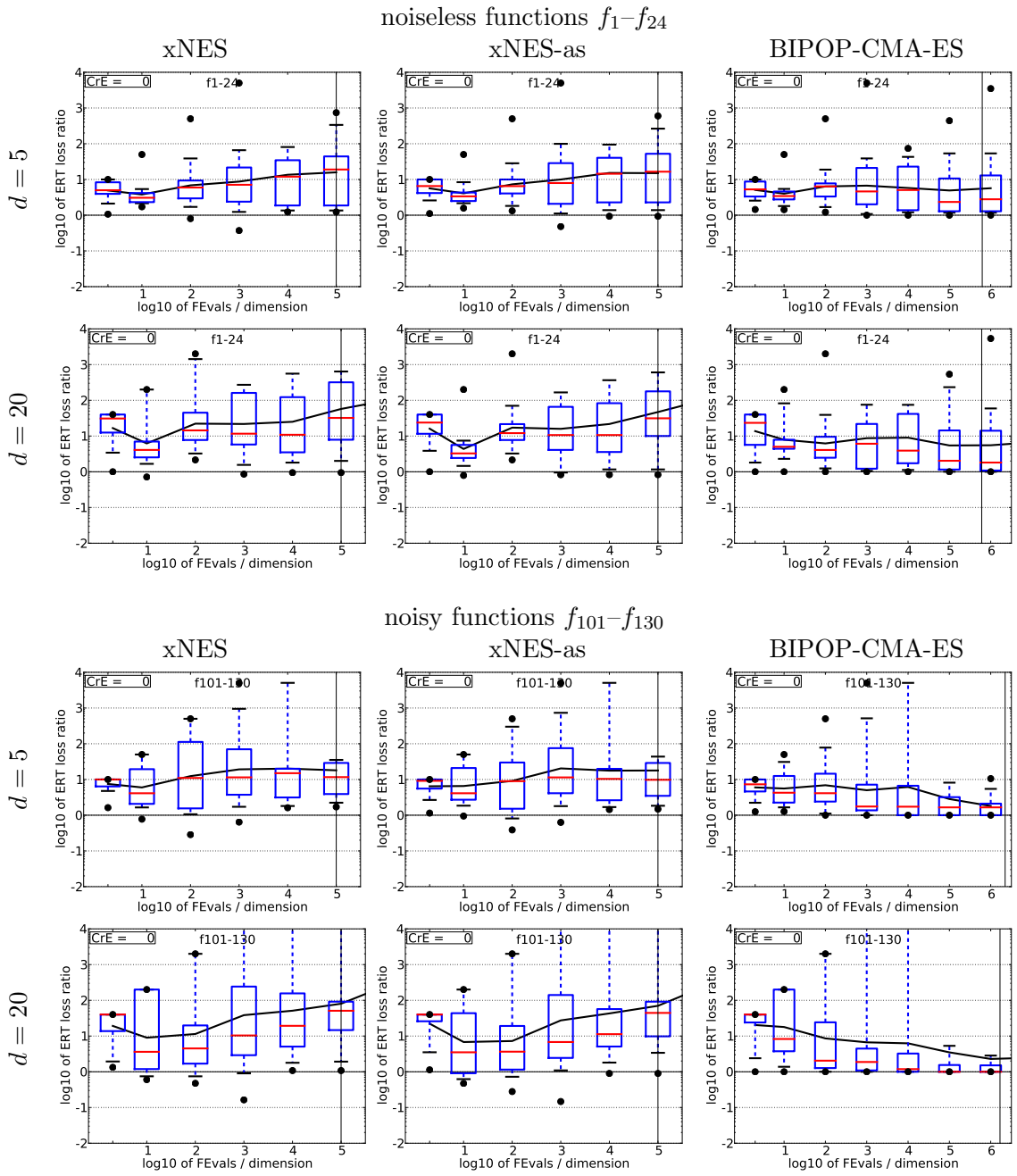


Figure 7: Loss ratio of expected running time (ERT), given a budget of function evaluations (here, smaller is better), for xNES (left column), xNES-as (middle column) and BIPOP-CMA-ES (right column). The target value f_t used for a given budget is the smallest (best) recorded function value such that $\mathrm{ERT}(f_t) \leq \mathrm{FEvals}$ for the presented algorithm. Shown is FEvals divided by the respective best $\mathrm{ERT}(f_t)$ from BBOB-2009 for all functions (noiseless f_1–f_{24}, top rows, and noisy f_{101}–f_{130}, bottom rows) in 5-D and 20-D. Black line: geometric mean. Box-Whisker error bar: 25-75%-ile (box) with median (red), 10-90%-ile (caps), and minimum and maximum ERT loss ratio (black points).

marked on the entire BBOB suite. In contrast to the BBOB setups, which are based on tricky fitness functions in *small* problem dimensions, this section illustrates how SNES scales with dimension, on a classical neuro-evolutionary controller design problem.

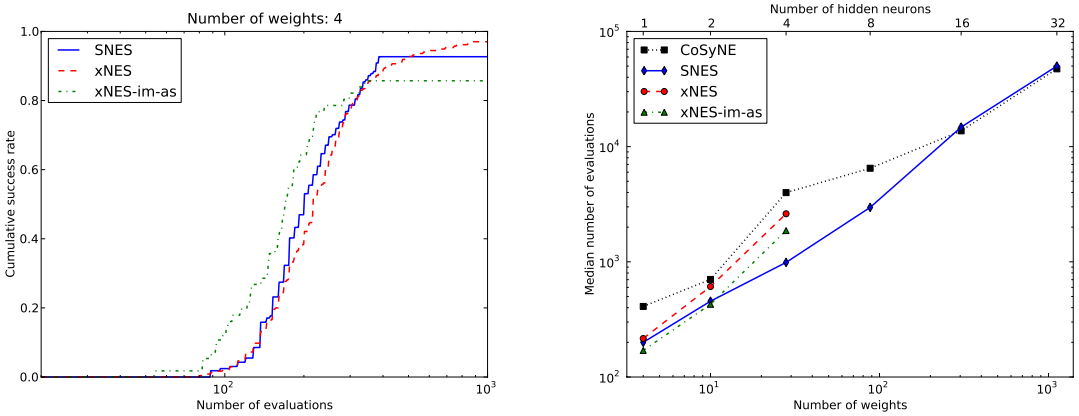


Figure 8: **Left:** Plotted are the cumulative success rates on the non-Markovian double-pole balancing task after a certain number of evaluations, empirically determined over 100 runs for each algorithm, using a single tanh-unit ($n = 1$) (i.e., optimizing 4 weights). We find that all three algorithm variants give state-of-the-art results, with a slightly faster but less robust performance for xNES-as. **Right:** Median number of evaluations required to solve the same task, but with increasing number of neurons (and corresponding number of weights). We limited the runtime to one hour per run, which explains why no results are available for xNES on higher dimensions (cubic time complexity). The fact that SNES quickly outperforms xNES, also in number of function evaluations, indicates that the benchmark is (sufficiently close to) separable, and it is unnecessary to use the full covariance matrix. For reference we also plot the corresponding results of the previously best performing algorithm CoSyNE (Gomez et al., 2008).

SNES is well-suited for neuroevolution problems because they tend to be high-dimensional, multi-modal, but with highly redundant global optima (there is not a unique set of weights that defines the optimal behavior). We use it to find a controller for non-Markovian double pole balancing, a task which involves balancing two differently sized poles hinged on a cart that moves on a finite track. The single control consists of the force F applied to the cart, and observations include the cart's position and the poles' angles, but no velocity information, which makes this task partially observable. It provides a perfect testbed for algorithms focusing on learning fine control with memory in continuous state and action spaces (Wieland, 1991). The controller is represented by a simple recurrent neural network, with three inputs, (position x and the two poles' angles β_1 and β_2), and a variable number n of tanh units in the output layer, which are fully connected (recurrently), resulting in a total of $n(n+3)$ weights to be optimized. The activation of the first of these recurrent

neurons directly determines the force to be applied. We use the implementation found in PyBrain (Schaul et al., 2010).

An evaluation is considered a success if the poles do not fall over for $100,000$ time steps. We experimented with recurrent layers of sizes $n = 1$ to $n = 32$ (corresponding to between 4 and 1120 weights). It turns out that a single recurrent neuron is sufficient to solve the task (Figure 8, left). In fact, both the xNES and SNES results are state-of-the-art, outperforming the previously best algorithm (CoSyNE; Gomez et al., 2008, with a median of 410 evaluations) by a factor two.

In practical scenarios however, we cannot know the best network size a priori, and thus the prudent choice consists in overestimating the required size. An algorithm that graciously scales with problem dimension is therefore highly desirable, and we find (Figure 8, right) that SNES is exhibiting precisely that behavior. The fact that SNES outperforms xNES with increasing dimension, also in number of function evaluations, indicates that the benchmark is not ill-conditioned, and it is unnecessary to use the full covariance matrix. We conjecture that this is a property shared with the majority of neuroevolution problems that have enough weights to exhibit redundant global optima (some of which can be found without considering all parameter covariances).

Additional SNES results, for example, on the Lennard-Jones benchmark for atom clusters (with dimension $d > 200$) can be found in Schaul et al. (2011).

6. Discussion and Conclusion

Our results on the BBOB benchmarks show that NES algorithms perform well across a wide variety of black-box optimization problems. We have demonstrated advantages and limitations of specific variants, and as such established the generality and flexibility of the NES framework. Experiments with heavy-tailed and separable distributions demonstrate the viability of the approach on high-dimensional domains. We obtained best reported results on the difficult task of training a neural controller for double pole-balancing.

Technique	Issue addressed	Applicability limited to	Relevant section
Natural gradient	Scale-invariance, many more	-	2.3
Fitness shaping	Robustness	-	3.1
Adaptation sampling	Performance, sensitivity	-	3.2
Exponential parameterization	Covariance constraints	Multivariate	3.4
Natural coordinate system	Computational efficiency	Multivariate	3.4

Table 2: Summary of enhancing techniques

Table 2 summarizes the various techniques we introduced. The plain search gradient suffers from premature convergence and lack of scale invariance (see Section 2.2). Therefore, we use the natural gradient instead, which turns NES into a viable optimization method. To improve performance and robustness, we introduced several novel techniques. Fitness shaping makes the NES algorithm invariant to order-preserving transformations of the fitness function, thus increasing robustness. Adaptation sampling adjusts the learning rates online, which yields highly performant results on standard benchmarks. Finally, the ex-

ponential parameterization is crucial for maintaining positive-definite covariance matrices, and the use of the natural coordinate system guarantees computational feasibility.

NES applies to general parameterizable distributions. In this paper, we have experimentally investigated two variants, adjusted to the particular properties of different problem classes. We demonstrated the power of the xNES variant using a full multinormal distribution, which is invariant under arbitrary translations and rotations, on the canonical suite of standard benchmarks. Additionally, we showed that the restriction of the covariance matrix to a diagonal parameterization (SNES) allows for scaling to very high dimensions, on the difficult non-Markovian double pole balancing task.

Acknowledgments

This research was funded through the 7th framework program of the European Union, under grant number 231576 (STIFF project) and ICT-270327 CompLACS, SNF grants 200020-116674/1, 200021-111968/1 and 200020-122124/1, and SSN grant Sinergia CRSIK0-122697. We thank Faustino Gomez for helpful suggestions and his CoSyNE data, as well as Andreas Krause for insightful discussions. We particularly thank the anonymous reviewers for their many constructive remarks.

References

Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi. Bidirectional relation between CMA evolution strategies and natural evolution strategies. In *Parallel Problem Solving from Nature (PPSN)*, 2010.

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

S. Amari and S. C. Douglas. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '98)*, volume 2, pages 1213–1216, 1998.

A. Auger. Convergence results for the $(1,\lambda)$-SA-ES using the theory of ϕ-irreducible Markov chains. *Theoretical Computer Science*, 334(1-3):35 – 69, 2005.

A. Berny. Selection and reinforcement learning for combinatorial optimization. In *Parallel Problem Solving from Nature PPSN VI*, volume 1917, pages 601–610. Springer Berlin / Heidelberg, 2000.

A. Berny. *Statistical machine learning and combinatorial optimization*, pages 287–306. Springer-Verlag, London, UK, 2001.

H.-G. Beyer. *The Theory of Evolution Strategies*. Springer-Verlag, New York, USA, 2001.

H.-G. Beyer and H.-P. Schwefel. Evolution strategies: a comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

P. A. N. Bosman and D. Thierens. Expanding from discrete to continuous estimation of distribution algorithms: the IDEA. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 767–776, London, UK, 2000. Springer-Verlag.

P. A. N. Bosman, J. Grahl, and D. Thierens. Adapted maximum-likelihood Gaussian models for numerical optimization with continuous EDAs. Technical report, 2007.

F. Friedrichs and C. Igel. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 64:107–117, 2005.

T. Glasmachers, T. Schaul, and J. Schmidhuber. A natural evolution strategy for multi-objective optimization. In *Parallel Problem Solving from Nature (PPSN)*, 2010a.

T. Glasmachers, T. Schaul, Y. Sun, D. Wierstra, and J. Schmidhuber. Exponential natural evolution strategies. In *Genetic and Evolutionary Computation Conference (GECCO)*, Portland, USA, 2010b.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989. ISBN 0201157675.

F. Gomez, J. Schmidhuber, and R. Miikkulainen. Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, 2008.

N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *GECCO (Companion)*, pages 2389–2396, 2009a.

N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 noisy testbed. In *GECCO (Companion)*, pages 2397–2402, 2009b.

N. Hansen and A. Auger. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical report, INRIA, 2010.

N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

N. Hansen, A. S. P. Niederberger, L. Guzzella, and P. Koumoutsakos. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *Transactions on Evolutionary Computation*, 13:180–197, 2009.

N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Noiseless function definitions. Technical report, INRIA, 2010a.

N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Noisy functions definitions. Technical report, INRIA, 2010b.

M. Hasenjäger, B. Sendhoff, T. Sonoda, and T. Arima. Three dimensional evolutionary aerodynamic design optimization with CMA-ES. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pages 2173–2180, New York, NY, USA, 2005. ACM.

J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, 1975.

C. Igel and M. Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50:2003, 2003.

J. Jägersküpper. Analysis of a simple evolutionary algorithm for minimization in Euclidean spaces. *Theoretical Computer Science*, 379(3):329–347, 2007.

G. A. Jastrebski and D. V. Arnold. Improving evolution strategies through active covariance matrix adaptation. In *IEEE Congress on Evolutionary Computation*, 2006.

M. Jebalia, A. Auger, M. Schoenauer, F. James, and M. Postel. Identification of the isotherm function in chromatography using CMA-ES. In *IEEE Congress on Evolutionary Computation*, pages 4289–4296, 2007.

M. Jebalia, A. Auger, and N. Hansen. Log-linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments. *Algorithmica*, pages 1–36, 2010.

J. Kennedy and R. Eberhart. *Swarm Intelligence.* Morgan Kaufmann, San Francisco, CA, 2001.

S. Kirkpatrick, C. D. Gelatt, Jr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

J. Klockgether and H. P. Schwefel. Two-phase nozzle and hollow core jet experiments. In *Proc. 11th Symp. Engineering Aspects of Magnetohydrodynamics*, pages 141–148, 1970.

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. ISSN 00034851.

P. Larrañaga. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.* Kluwer Academic Publishers, 2002.

H. Mühlenbein and G. Paass. From recombination of genes to the estimation of distributions I. binary parameters. In *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, PPSN IV, pages 178–187, London, UK, 1996. Springer-Verlag.

S. D. Muller, J. Marchetto, S. Airaghi, and P. Koumoutsakos. Optimization based on bacterial chemotaxis. *IEEE Transactions on Evolutionary Computation*, 6:6–16, 2002.

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

A. Ostermeier, A. Gawelczyk, and N. Hansen. Step-size adaption based on non-local use of selection information. In *The Third Conference on Parallel Problem Solving from Nature*, pages 189–198, London, UK, 1994. Springer-Verlag.

M. Pelikan, D. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. In *American Control Conference*, volume 5, pages 3289 –3293 vol.5, 2000.

M. Pelikan, K. Sastry, and E. C. Paz. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications (Studies in Computational Intelligence)*. Springer-Verlag New York, Inc., 2006.

J. Peters. *Machine Learning of Motor Skills for Robotics*. PhD thesis, Department of Computer Science, University of Southern California, 2007.

I. Rechenberg and M. Eigen. *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzboog Stuttgart, 1973.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591. IEEE Press, 1993.

R. Ros and N. Hansen. A simple modification in CMA-ES achieving linear time and space complexity. In R. et al., editor, *Parallel Problem Solving from Nature, PPSN X*, pages 296–305. Springer, 2008.

R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning (Information Science and Statistics)*. Springer, 2004.

T. Schaul. Benchmarking separable natural evolution strategies on the noiseless and noisy black-box optimization testbeds. In *Black-box Optimization Benchmarking Workshop, Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012a.

T. Schaul. Benchmarking exponential natural evolution strategies on the noiseless and noisy black-box optimization testbeds. In *Black-box Optimization Benchmarking Workshop, Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012b.

T. Schaul. Benchmarking natural evolution strategies with adaptation sampling on the noiseless and noisy black-box optimization testbeds. In *Black-box Optimization Benchmarking Workshop, Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012c.

T. Schaul. Comparing natural evolution strategies to BIPOP-CMA-ES on noiseless and noisy black-box optimization testbeds. In *Black-box Optimization Benchmarking Workshop, Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012d.

T. Schaul. Investigating the impact of adaptation sampling in natural evolution strategies on black-box optimization testbeds. In *Black-box Optimization Benchmarking Workshop, Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012e.

T. Schaul. Natural evolution strategies converge on sphere functions. In *Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012f.

T. Schaul and J. Schmidhuber. Metalearning. *Scholarpedia*, 5(6):4650, 2010.

T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 11:743–746, 2010.

T. Schaul, T. Glasmachers, and J. Schmidhuber. High dimensions and heavy tails for natural evolution strategies. In *Genetic and Evolutionary Computation Conference*, 2011.

H.-P. Schwefel. Numerische optimierung von computer-modellen mittels der evolutionsstrategie, 1977.

J. Shepherd, D. McDowell, and K. Jacob. Modeling morphology evolution and mechanical behavior during thermo-mechanical processing of semi-crystalline polymers. *Journal of the Mechanics and Physics of Solids*, 54(3):467 – 489, 2006.

O. M. Shir and T. Bäck. The second harmonic generation case-study as a gateway for ES to quantum control problems. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 713–721, New York, NY, USA, 2007. ACM.

R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11:341–359, December 1997. ISSN 0925-5001.

Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Stochastic search using the natural gradient. In *International Conference on Machine Learning (ICML)*, 2009a.

Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Efficient natural evolution strategies. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2009b.

A. Wieland. Evolving neural network controllers for unstable systems. In *Proceedings of the International Joint Conference on Neural Networks (Seattle, WA)*, pages 667–673, 1991.

D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber. Natural evolution strategies. In *Proceedings of the Congress on Evolutionary Computation (CEC08), Hongkong*. IEEE Press, 2008.

S. Winter, B. Brendel, and C. Igel. Registration of bone structures in 3D ultrasound and CT data: Comparison of different optimization strategies. *International Congress Series*, 1281:242 – 247, 2005.

Journal of Machine Learning Research 15 (2014) 981-1009 Submitted 10/12; Revised 9/13; Published 3/14

Conditional Random Field with High-order Dependencies for Sequence Labeling and Segmentation

Nguyen Viet Cuong NVCUONG@COMP.NUS.EDU.SG
Nan Ye YENAN@COMP.NUS.EDU.SG
Wee Sun Lee LEEWS@COMP.NUS.EDU.SG
Department of Computer Science
National University of Singapore
13 Computing Drive
Singapore 117417

Hai Leong Chieu CHAILEON@DSO.ORG.SG
DSO National Laboratories
20 Science Park Drive
Singapore 118230

Editor: Kevin Murphy

Abstract

Dependencies among neighboring labels in a sequence are important sources of information for sequence labeling and segmentation. However, only first-order dependencies, which are dependencies between adjacent labels or segments, are commonly exploited in practice because of the high computational complexity of typical inference algorithms when longer distance dependencies are taken into account. In this paper, we give efficient inference algorithms to handle high-order dependencies between labels or segments in conditional random fields, under the assumption that the number of distinct label patterns used in the features is small. This leads to efficient learning algorithms for these conditional random fields. We show experimentally that exploiting high-order dependencies can lead to substantial performance improvements for some problems, and we discuss conditions under which high-order features can be effective.

Keywords: conditional random field, semi-Markov conditional random field, high-order feature, sequence labeling, segmentation, label sparsity

1. Introduction

Many problems can be cast as the problem of labeling or segmenting a sequence of observations. Examples include natural language processing tasks, such as part-of-speech tagging (Lafferty et al., 2001), phrase chunking (Sha and Pereira, 2003), named entity recognition (McCallum and Li, 2003), and tasks in bioinformatics such as gene prediction (Culotta et al., 2005) and RNA secondary structure prediction (Durbin, 1998).

Conditional random field (CRF) (Lafferty et al., 2001) is a discriminative, undirected Markov model which represents a conditional probability distribution of a structured output variable $\mathbf{y}$ given an observation $\mathbf{x}$. Conditional random fields have been successfully applied in sequence labeling and segmentation. Compared to generative models such as hidden Markov models (Rabiner, 1989), CRFs model only the conditional distribution of $\mathbf{y}$

Type of CRF	Feature example
First-order (Lafferty et al., 2001)	*author year*
Semi-CRF (Sarawagi and Cohen, 2004)	*author+ year+*
High-order (Ye et al., 2009, this paper)	*author year title title*
High-order semi-CRF (this paper)	*author+ year+ title+*

Table 1: Examples of the information that can be captured by different types of CRFs for the bibliography extraction task. The $x+$ symbol represents a segment of "1 or more" labels of class x.

given $\mathbf{x}$, and do not model the observations $\mathbf{x}$. Hence, CRFs can be used to encode complex dependencies of $\mathbf{y}$ on $\mathbf{x}$ without significantly increasing the inference and learning costs. However, inference for CRFs is NP-hard in general (Istrail, 2000), and most CRFs have been restricted to consider very local dependencies. Examples include the linear-chain CRF which considers dependencies between at most two adjacent labels (Lafferty et al., 2001) and the first-order semi-Markov CRF (semi-CRF) which considers dependencies between at most two adjacent segments (Sarawagi and Cohen, 2004), where a segment is a contiguous sequence of identical labels. In linear-chain CRF and semi-CRF, a k^{th}-order feature is a feature that encodes the dependency between $\mathbf{x}$ and $(k+1)$ consecutive labels or segments. Existing inference algorithms for CRFs such as the Viterbi and the forward-backward algorithms can only handle up to first-order features, and inference algorithms for semi-CRFs (Sarawagi and Cohen, 2004) can only handle up to first-order features between segments. These algorithms can be easily generalized to handle high-order features, but will require time exponential in k. In addition, a general inference algorithm such as the clique tree algorithm (Huang and Darwiche, 1996) also requires time exponential in k to handle k^{th}-order features ($k > 1$).

In this paper, we exploit a form of sparsity that is often observed in real data to design efficient algorithms for inference and learning with high-order label or segment dependencies. Our algorithms are presented for high-order semi-CRFs in its most general form. Algorithms for high-order CRFs are obtained by restricting the segment lengths to 1, and algorithms for linear-chain CRFs and first-order semi-CRFs are obtained by restricting the maximum order to 1.

We use a bibliography extraction task in Table 1 to show examples of features that can be used with different classes of CRFs. In this task, different fields are often arranged in a fixed order, hence using high-order features can be advantageous. The sparsity property that we exploit is the following *label pattern sparsity*: the number of observed sequences of k consecutive segment labels (e.g., "*author+ year+ title+*" is one such sequence where $k = 3$) is much smaller than n^k, where n is the number of distinct labels. This assumption often holds in real problems. Under this assumption, we give algorithms for computing marginals, partition functions, and Viterbi parses for high-order semi-CRFs. The partition function and the marginals can be used to efficiently compute the log-likelihood and its gradient. In turn, the log-likelihood and its gradient can be used with quasi-Newton methods to efficiently find the maximum likelihood parameters (Sha and Pereira, 2003). The algo-

rithm for Viterbi parsing can also be used with cutting plane methods to train max-margin solutions for sequence labeling problems in polynomial time (Tsochantaridis et al., 2004). Our inference and learning algorithms run in time polynomial in the maximum segment length as well as the number and length of the label patterns that the features depend on.

We demonstrate that modeling high-order dependencies can lead to significant performance improvements in various problems. In our first set of experiments, we focus on high-order CRFs and demonstrate that using high-order features can improve performance in sequence labeling problems. We show that in handwriting recognition, using even simple high-order indicator features improves performance over using linear-chain CRFs, and significant performance improvement is observed when the maximum order of the indicator features is increased. We also use a synthetic data set to discuss the conditions under which high-order features can be helpful. In our second set of experiments, we demonstrate that using high-order semi-Markov features can be helpful in some applications. More specifically, we show that high-order semi-CRFs outperform high-order CRFs and first-order semi-CRFs on three segmentation tasks: relation argument detection, punctuation prediction, and bibliography extraction.[1]

2. Algorithms for High-order Dependencies

Our algorithms are presented for high-order semi-CRFs in its most general form. These algorithms generalize the algorithms for linear-chain CRFs and first-order semi-CRFs, which are special cases of our algorithms when the maximum order is set to 1. They also generalize the algorithms for high-order CRFs (Ye et al., 2009), which are special cases of our algorithms when the segment lengths are set to 1. Thus, only the general algorithms described in this section need to be implemented to handle all these different cases.[2]

2.1 High-order Semi-CRFs

Let $\mathcal{Y} = \{1, 2, \ldots, n\}$ denote the set of distinct labels, $\mathbf{x} = (x_1, \ldots, x_{|\mathbf{x}|})$ denote an input sequence of length $|\mathbf{x}|$, and $\mathbf{x}_{a:b}$ denote the sub-sequence $(x_a, \ldots, x_b)$. A *segment* of $\mathbf{x}$ is defined as a triplet (u, v, y), where y is the common label of the segment $\mathbf{x}_{u:v}$. A *segmentation* for $\mathbf{x}_{a:b}$ is a segment sequence $\mathbf{s} = (s_1, \ldots, s_p)$, with $s_j = (u_j, v_j, y_j)$ such that $u_{j+1} = v_j + 1$ for all j, $u_1 = a$ and $v_p = b$. A segmentation for $\mathbf{x}_{a:b}$ is a partial segmentation for $\mathbf{x}$.

A semi-CRF defines a conditional distribution over all possible segmentations $\mathbf{s}$ of an input sequence $\mathbf{x}$ such that

$$P(\mathbf{s}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \exp\Big(\sum_{i=1}^{m} \sum_{t=1}^{|\mathbf{s}|} \lambda_i f_i(\mathbf{x}, \mathbf{s}, t)\Big)$$

1. This paper is an extended version of a previous paper (Ye et al., 2009) published in NIPS 2009. Some of the additional material presented here has also been presented as an abstract (Nguyen et al., 2011) at the ICML Workshop on Structured Sparsity: Learning and Inference, 2011. The source code for our algorithms is available at `https://github.com/nvcuong/HOSemiCRF`.
2. In an earlier paper (Ye et al., 2009), we give algorithms for high-order CRFs which are similar to those presented here. The main difference lies in the backward algorithm. The version presented here is a *conditional* version which uses properties of labels before the suffix labels being considered, making extension to the high-order semi-Markov features simpler.

where $Z_{\mathbf{x}} = \sum_{\mathbf{s}} \exp(\sum_i \sum_t \lambda_i f_i(\mathbf{x}, \mathbf{s}, t))$ is the partition function with the summation over all segmentations of $\mathbf{x}$, and $\{f_i(\mathbf{x}, \mathbf{s}, t)\}_{1 \leq i \leq m}$ is the set of semi-Markov features, each of which has a corresponding weight λ_i.

We shall work with features of the following form

$$f_i(\mathbf{x}, \mathbf{s}, t) = \begin{cases} g_i(\mathbf{x}, u_t, v_t) & \text{if } y_{t-|\mathbf{z}^i|+1} \dots y_t = \mathbf{z}^i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $\mathbf{z}^i \in \mathcal{Y}^{|\mathbf{z}_i|}$ is a segment label pattern associated with f_i, and $\mathbf{s}$ is a segmentation or a partial segmentation for $\mathbf{x}$. The function $f_i(\mathbf{x}, \mathbf{s}, t)$ depends on the t-th segment as well as the label pattern $\mathbf{z}^i$ and is said to be of order $|\mathbf{z}^i| - 1$. The order of the resulting semi-CRF is the maximal order of the features.

We will give exact inference algorithms for high-order semi-CRFs in the following sections. As in exact inference algorithms for linear-chain CRFs and semi-CRFs, our algorithms perform forward and backward passes to obtain the necessary information for inference.

2.2 Notations

Without loss of generality, let $\mathcal{Z} = \{\mathbf{z}^1, \dots, \mathbf{z}^M\}$ be the *segment label pattern set*, that is, the set of distinct segment label patterns of the m features ($M \leq m$). For our forward algorithm, the *forward-state set* $\mathcal{P} = \{\mathbf{p}^1, \dots, \mathbf{p}^{|\mathcal{P}|}\}$ consists of distinct elements in the set of all the labels and proper prefixes (including the empty sequence ϵ) of the segment label patterns. Thus, $\mathcal{P} = \mathcal{Y} \cup \{\mathbf{z}^j_{1:k}\}_{0 \leq k < |\mathbf{z}^j|, 1 \leq j \leq M}$. For the backward algorithm, the *backward-state set* $\mathcal{S} = \{\mathbf{s}^1, \dots, \mathbf{s}^{|\mathcal{S}|}\}$ consists of distinct elements in $\mathcal{P}\mathcal{Y}$, that is, the set consisting of elements in $\mathcal{P}$ concatenated with a label in $\mathcal{Y}$.

Transitions between states in our algorithm are defined using the suffix relationships between them. We use $\mathbf{z}_1 \leq^s \mathbf{z}_2$ to denote that $\mathbf{z}_1$ is a suffix of $\mathbf{z}_2$. The longest suffix relation on a set $\mathcal{A}$ is denoted by $\mathbf{z}_1 \leq^s_{\mathcal{A}} \mathbf{z}_2$. This relation holds true if and only if $\mathbf{z}_1$, among all the elements of $\mathcal{A}$, is the longest suffix of $\mathbf{z}_2$. More formally, $\mathbf{z}_1 \leq^s_{\mathcal{A}} \mathbf{z}_2$ if and only if $\mathbf{z}_1 \in \mathcal{A}$ and $\mathbf{z}_1 \leq^s \mathbf{z}_2$ and $\forall \mathbf{z} \in \mathcal{A}, \mathbf{z} \leq^s \mathbf{z}_2 \Rightarrow \mathbf{z} \leq^s \mathbf{z}_1$.

2.3 Training

Given a training set $\mathcal{T}$, we estimate the model parameters $\vec{\lambda} = (\lambda_1, \dots, \lambda_m)$ by maximizing the regularized log-likelihood function

$$\mathcal{L}_{\mathcal{T}}(\vec{\lambda}) \quad = \quad \sum_{(\mathbf{x},\mathbf{s}) \in \mathcal{T}} \log P(\mathbf{s}|\mathbf{x}) - \sum_{i=1}^{m} \frac{\lambda_i^2}{2\sigma^2_{\text{reg}}}$$

where σ_{reg} is a regularization parameter. This function is convex, and thus can be maximized using any convex optimization algorithm. In our implementation, we use the L-BFGS method (Liu and Nocedal, 1989). The method requires computation of the value of $\mathcal{L}_{\mathcal{T}}(\vec{\lambda})$ and its partial derivatives

$$\frac{\partial \mathcal{L}_{\mathcal{T}}}{\partial \lambda_i} = \tilde{E}(f_i) - E(f_i) - \frac{\lambda_i}{\sigma^2_{\text{reg}}}$$

where $\tilde{E}(f_i) = \sum_{(\mathbf{x},\mathbf{s}) \in \mathcal{T}} \sum_t f_i(\mathbf{x}, \mathbf{s}, t)$ is the empirical feature sum of the feature f_i, and $E(f_i) = \sum_{(\mathbf{x},\mathbf{s}) \in \mathcal{T}} \sum_{\mathbf{s}'} P(\mathbf{s}'|\mathbf{x}) \sum_t f_i(\mathbf{x}, \mathbf{s}', t)$ is the expected feature sum of f_i. To compute

$\mathcal{L}_{\mathcal{T}}(\vec{\lambda})$ and its partial derivatives, we need to efficiently compute the partition function $Z_{\mathbf{x}}$ and the expected feature sum of f_i's.

2.3.1 Partition Function

For any $\mathbf{p}^i \in \mathcal{P}$, let $\mathbf{p}_{j,\mathbf{p}^i}$ be the set of all segmentations for $\mathbf{x}_{1:j}$ whose segment label sequences contain $\mathbf{p}^i$ as the longest suffix among all elements in $\mathcal{P}$. We define the forward variables $\alpha_{\mathbf{x}}(j, \mathbf{p}^i)$ as follows

$$\alpha_{\mathbf{x}}(j, \mathbf{p}^i) = \sum_{\mathbf{s} \in \mathbf{p}_{j,\mathbf{p}^i}} \exp(\sum_k \sum_t \lambda_k f_k(\mathbf{x}, \mathbf{s}, t)).$$

The above definition of the forward variable $\alpha_{\mathbf{x}}$ is the same as the usual definition of forward variable for first-order semi-CRFs when only zeroth-order and first-order semi-Markov features are used. The forward variables can be computed by dynamic programming:

$$\alpha_{\mathbf{x}}(j, \mathbf{p}^i) = \sum_{d=0}^{L-1} \sum_{(\mathbf{p}^k, y): \mathbf{p}^i \leq_{\mathcal{P}}^{s} \mathbf{p}^k y} \Psi_{\mathbf{x}}(j-d, j, \mathbf{p}^k y) \alpha_{\mathbf{x}}(j-d-1, \mathbf{p}^k)$$

where L is the longest possible length of a segment, $\sum_{i:\mathrm{Pred}(i)}$ denotes summation over all i's satisfying the predicate $\mathrm{Pred}(i)$, and $\Psi_{\mathbf{x}}(u, v, \mathbf{p})$ counts the contribution of features activated when there is a segment label sequence $\mathbf{p}$ with its last segment having boundary (u, v). The factor $\Psi_{\mathbf{x}}(u, v, \mathbf{p})$ is defined as

$$\Psi_{\mathbf{x}}(u, v, \mathbf{p}) = \exp(\sum_{i: \mathbf{z}^i \leq^s \mathbf{p}} \lambda_i g_i(\mathbf{x}, u, v)).$$

The correctness of the above recurrence is shown in Appendix A. The partition function can be computed from the forward variables by

$$Z_{\mathbf{x}} = \sum_i \alpha_{\mathbf{x}}(|\mathbf{x}|, \mathbf{p}^i).$$

2.3.2 Expected Feature Sum

Let $\mathbf{s}_j$ be the set of all partial segmentations for $\mathbf{x}_{j:|\mathbf{x}|}$. For $\mathbf{s} \in \mathbf{s}_j$ and $\mathbf{s}^k \in \mathcal{S}$, we define for each feature f_i a conditional feature function $f_i(\mathbf{x}, \mathbf{s}, t | \mathbf{s}^k)$, which takes the value of $f_i(\mathbf{x}, \mathbf{s}, t)$ when $\mathbf{s}^k$ is the longest suffix (in $\mathcal{S}$) of the segment label sequence for $\mathbf{x}_{1:j-1}$. Otherwise, its value is 0. For example, if $\mathbf{s} = (s_1, \ldots, s_p) \in \mathbf{s}_j$ and $s_1 = (u_1, v_1, y_1)$, then

$$f_i(\mathbf{x}, \mathbf{s}, 1 | \mathbf{s}^k) = \begin{cases} g_i(\mathbf{x}, u_1, v_1) & \text{if } \mathbf{z}^i \leq^s \mathbf{s}^k y_1 \\ 0 & \text{otherwise} \end{cases}.$$

For each $\mathbf{s}^i \in \mathcal{S}$, we define the backward variables $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$ as follows

$$\beta_{\mathbf{x}}(j, \mathbf{s}^i) = \sum_{\mathbf{s} \in \mathbf{s}_j} \exp(\sum_k \sum_t \lambda_k f_k(\mathbf{x}, \mathbf{s}, t | \mathbf{s}^i)).$$

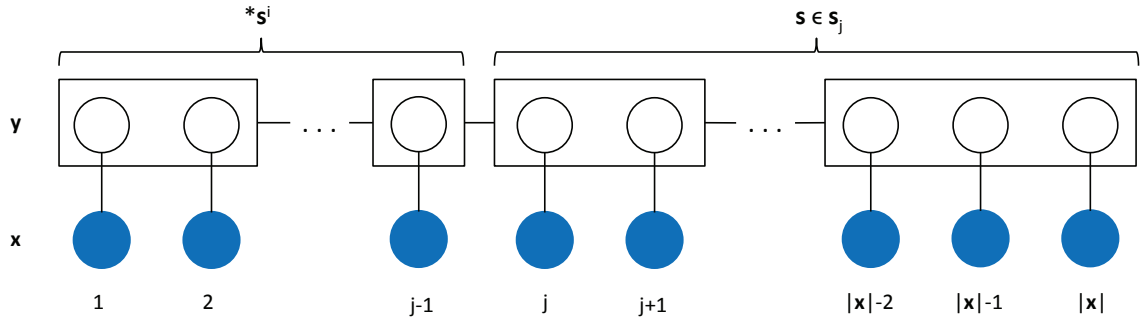


Figure 1: An illustration of the backward variable $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$. Each rectangular box corresponds to a segment. The regular expression $*\mathbf{s}^i$ means that $\mathbf{s}^i$ is the suffix of the segment label sequence for $\mathbf{x}_{1:j-1}$. In fact, $\mathbf{s}^i$ is the longest suffix of the segment label sequence for $\mathbf{x}_{1:j-1}$. The summation in the definition of $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$ is over all the partial segmentations $\mathbf{s}$ of $\mathbf{x}_{j:|\mathbf{x}|}$.

Figure 1 gives an illustration of the backward variable $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$. Note that our definition of $\beta_{\mathbf{x}}$ uses the conditional feature function and does not generalize the usual definitions of the backward variables in first-order semi-CRFs (Sarawagi and Cohen, 2004) or high-order CRFs (Ye et al., 2009).

Similar to the case of forward variables, we can compute $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$ by dynamic programming:

$$\beta_{\mathbf{x}}(j, \mathbf{s}^i) = \sum_{d=0}^{L-1} \sum_{(\mathbf{s}^k, y): \mathbf{s}^k \leq_S^s \mathbf{s}^i y} \Psi_{\mathbf{x}}(j, j+d, \mathbf{s}^i y) \beta_{\mathbf{x}}(j+d+1, \mathbf{s}^k).$$

In Appendix A, we show the correctness proof for the recurrence. We can now compute the marginals $P(u, v, \mathbf{z}|\mathbf{x})$ for each $\mathbf{z} \in \mathcal{Z}$ and $u \leq v$, where $P(u, v, \mathbf{z}|\mathbf{x})$ denotes the probability that a segmentation of $\mathbf{x}$ contains label pattern $\mathbf{z}$ and has (u, v) as $\mathbf{z}$'s last segment boundaries. These marginals can be computed by

$$P(u, v, \mathbf{z}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \sum_{(\mathbf{p}^i, y): \mathbf{z} \leq^s \mathbf{p}^i y} \alpha_{\mathbf{x}}(u-1, \mathbf{p}^i) \Psi_{\mathbf{x}}(u, v, \mathbf{p}^i y) \beta_{\mathbf{x}}(v+1, \mathbf{p}^i y).$$

We compute the expected feature sum for f_i by

$$E(f_i) = \sum_{(\mathbf{x}, \mathbf{s}) \in \mathcal{T}} \sum_{u \leq v} P(u, v, \mathbf{z}^i|\mathbf{x}) g_i(\mathbf{x}, u, v).$$

In Appendix B, we give an example to illustrate our algorithms for the second-order CRF model.

Using the conditional feature function to define the backward variables $\beta_{\mathbf{x}}$ can help to simplify the computation of the marginals for high-order semi-CRF models. If we directly generalized the usual definition of the backward variables (Ye et al., 2009) to high-order semi-CRFs (which can be done easily), computing the marginals using these backward variables would be complicated. The main reason is that the semi-Markov features in Equation

(1) only know the correct position (u_t, v_t) of the last segment. In other words, although they know the label sequence of the previous segments, the features do not know the actual boundaries of these segments. So, to compute the marginal $P(u, v, \mathbf{z}|\mathbf{x})$ using the usual extension of the backward variables, we need to sum over all possible segmentations near (u, v) that contain (u, v) as a segment. This may result in an algorithm that is exponential in the order of the semi-CRFs. Note that this problem does not occur for high-order CRFs (Ye et al., 2009) since in these models, the segment length is 1 and thus we can always determine the boundaries of the segments.

2.4 Decoding

We compute the most likely segmentation for high-order semi-CRF by a Viterbi-like decoding algorithm. It is the same as the forward algorithm with the sum operator replaced by the max operator. Define

$$\delta_{\mathbf{x}}(j, \mathbf{p}^i) = \max_{\mathbf{s} \in \mathbf{P}_{j,\mathbf{p}^i}} \exp(\sum_k \sum_t \lambda_k f_k(\mathbf{x}, \mathbf{s}, t)).$$

These variables can be computed by

$$\delta_{\mathbf{x}}(j, \mathbf{p}^i) = \max_{(d,\mathbf{p}^k,y):\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{p}^k y} \Psi_{\mathbf{x}}(j-d, j, \mathbf{p}^k y)\delta_{\mathbf{x}}(j-d-1, \mathbf{p}^k).$$

Note that the value of d is inclusively between 0 and $L-1$ in the above equation. The most likely segmentation can be obtained using backtracking from $\max_i \delta_{\mathbf{x}}(|\mathbf{x}|, \mathbf{p}^i)$.

2.5 Time Complexity

We now give rough time bounds for the above algorithm. It is important to note that the bounds given in this part are pessimistic, and the computation can be done more quickly in practice. For simplicity, we assume that the features $g_i(\cdot, \cdot, \cdot)$ can be computed in $O(1)$ time for all $i \in \{1, 2, \ldots, m\}$ and the algorithm would pre-compute all the values of $\Psi_{\mathbf{x}}$ before doing the forward and backward passes. This assumption often holds for features used in practice, although one can define g_i's which are arbitrarily difficult to compute.

Since the total number of different patterns of the last argument of $\Psi_{\mathbf{x}}$ is $O(|\mathcal{S}||\mathcal{Y}|) = O(|\mathcal{P}||\mathcal{Y}|^2)$, the time complexity to pre-compute all the values of $\Psi_{\mathbf{x}}$ in the worst case is $O(mT^2|\mathcal{P}||\mathcal{Y}|^2) = O(mn^2T^2|\mathcal{P}|)$, where T is the maximum length of an input sequence. After pre-computing the values of $\Psi_{\mathbf{x}}$, we can compute all the values of $\alpha_{\mathbf{x}}$ in $O(T^2|\mathcal{Y}||\mathcal{P}|)$ time. Similarly, the time complexity to compute all the values of $\beta_{\mathbf{x}}$ is $O(T^2|\mathcal{Y}||\mathcal{S}|)$. Then, with these values, we can compute all the marginal probabilities in $O(T^2|\mathcal{Z}||\mathcal{P}|)$. Finally, the time complexity for decoding is $O(T^2|\mathcal{Y}||\mathcal{P}|)$.

3. Experiments

In this section, we describe experiments comparing CRFs, semi-CRFs, high-order CRFs, and high-order semi-CRFs. The experiments in Section 3.1 show the advantages of the high-order CRFs, while those in Section 3.2 show the advantages of the high-order semi-CRFs.

3.1 Experiments with High-order CRFs

The practical feasibility of making use of high-order features based on our algorithm lies in the observation that the label pattern sparsity assumption often holds. Our algorithm can be applied to take those high-order features into consideration: high-order features now form a component that one can play with in feature engineering.

Now, the question is whether high-order features are *practically significant*. We first use a synthetic data set to explore conditions under which high-order features can be expected to help. We then use a handwritten character recognition problem to demonstrate that even incorporating simple high-order features can lead to impressive performance improvement on a naturally occurring data set.[3]

3.1.1 Synthetic Data Generated Using k^{th}-order Markov Model

We randomly generate an order k Markov model with n states $s_1, \ldots, s_n$ as follows. To increase pattern sparsity, we allow at most r randomly chosen possible next states given the previous k states. This limits the number of possible label sequences in each length $(k+1)$ segment from n^{k+1} to $n^k r$. The conditional probabilities of these r next states are generated by randomly selecting a vector from the uniform distribution over $[0,1]^r$ and normalizing them. Each state s_i generates an observation $(a_1, \ldots, a_m)$ such that a_j follows a Gaussian distribution with mean μ_{ij} and standard deviation σ. Each $\mu_{i,j}$ is independently randomly generated from the uniform distribution over $[0,1]$. In the experiments, we use values of $n = 5$, $r = 2$ and $m = 3$.

The standard deviation σ controls how much information the observations reveal about the states. If σ is very small as compared to most μ_{ij}'s, then using the observations alone as features is likely to be good enough to obtain a good classifier of the states; the label correlations become less important for classification. However, if σ is large, then it is difficult to distinguish the states based on the observations alone and the label correlations, particularly those captured by higher order features are likely to be helpful.

We use the current, previous, and next observations, rather than just the current observation as features, exploiting the conditional probability modeling strength of CRFs. For higher order features, we simply use all indicator features that appeared in the training data up to a maximum order. We considered the case $k = 2$ and $k = 3$, and varied σ and the maximum order. We run the experiment with training sets that contain 300, 400, and 500 sequences, and evaluate the models on a test set that contains 500 sequences. All the sequences are of length 20; each sequence was initialized with a random sequence of length k and generated using the randomly generated order k Markov model. Training was done by maximizing the regularized log-likelihood with regularization parameter $\sigma_{\text{reg}} = 1$ in all experiments in this paper. The experimental results are shown in Figures 2.

Figure 2 shows that the high-order indicator features are useful in all cases. In particular, we can see that it is beneficial to increase the order of the high-order features when the underlying model has longer distance correlations. As expected, increasing the order of the features beyond the order of the underlying model is not helpful. The results also suggest

3. The results given in the earlier version of this work (Ye et al., 2009) are significantly lower than the results presented here due to a bug in the decoding algorithm. We have fixed the bug and reported the corrected results in this paper.

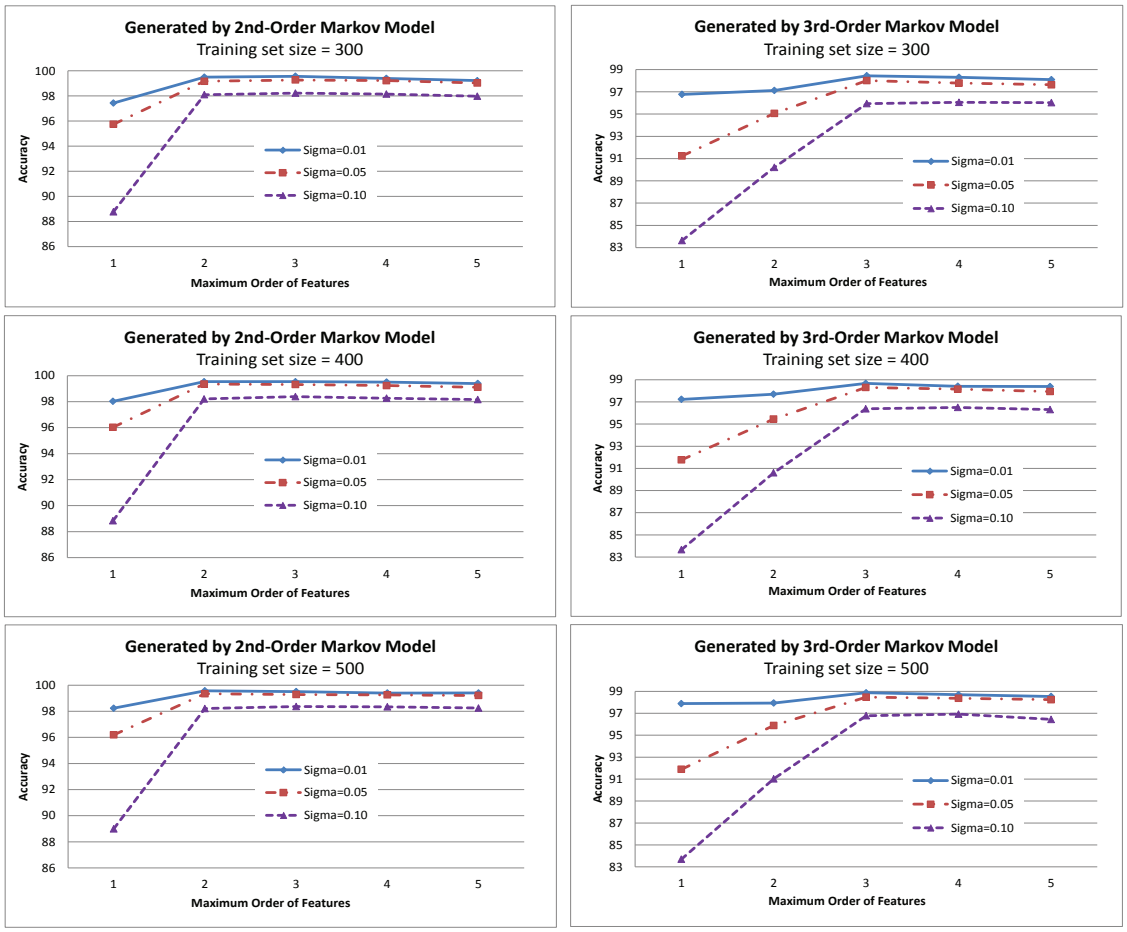


Figure 2: Accuracy of high-order CRFs as a function of maximum order on synthetic data sets.

that in general, if the observations are closely coupled with the states (in the sense that different states correspond to very different observations), then feature engineering on the observations is generally enough to perform well, and it is less important to use high-order features to capture label correlations. On the other hand, when such coupling is not clear, it becomes important to capture the label correlations, and high-order features can be useful.

We also study the effects of spurious, rare high-order patterns, and show that such patterns in the training or test set do not significantly impair performance of high-order CRFs in our experiments. For this purpose, we tabulate the proportion of fourth-order patterns (i.e., length 5 patterns) exclusive to the training or test sets in Table 2. The statistics show that around 10% of the patterns are exclusive to the training or test data. On the other hand, the results in Figure 2 show that when these patterns are used in the fourth-order model, the performance only drops slightly. Even if we increase the number of spurious, rare high-order patterns (by reducing the training data size), there is no significant drop in accuracies for high-order CRFs.

Size	300		400		500	
Order	Train	Test	Train	Test	Train	Test
2	16/173	13/170	17/175	12/170	17/177	10/170
3	34/393	58/417	37/406	48/417	42/424	35/417

Table 2: Proportions of length 5 patterns exclusive to training and test data where the data sets are generated by 2^{nd}-order and 3^{rd}-order Markov models. For each proportion, the denominator shows the number of patterns in the data set, and the numerator shows the number of patterns exclusive to it. Nearly all of these patterns occur for less than 5 times (mostly once or twice). Note that the labels are first generated independently of σ in our data sets, thus the statistics are the same for all σ values.

In practical problems, regularization may work well as a means for avoiding overfitting spurious high-order features. But this depends on how heavily the training process is regularized, and some tuning may be needed. For example, for a regularizer like Gaussian regularizer $\sum_i \frac{\lambda_i^2}{2\sigma_{reg}^2}$, the parameter σ_{reg} is often determined using a validation data set or cross-validation on the training data.

3.1.2 Handwriting Recognition

We used the handwriting recognition data set (Taskar et al., 2004), consisting of around 6100 handwritten words with an average length of around 8 characters. The data was originally collected by Kassel (1995) from around 150 human subjects. The words were segmented into characters, and each character was converted into an image of 16 by 8 binary pixels. In this labeling problem, each x_i is the image of a character, and each y_i is a lower-case letter. The experimental setup is the same as that used by Taskar et al. (2004): the data set was divided into 10 folds with each fold having approximately 600 training and 5500 test examples and the zero-th order features for a character are the pixel values.

For high-order features, we again used all indicator features that appeared in the training data up to a maximum order. The average accuracies over the 10 folds are shown in Figure 3, where strong improvements are observed as the maximum order increases. Figure 3 also shows the number of label patterns, the total training time, and the running time per iteration of the L-BFGS algorithm (which requires computation of the gradient and value of the function at each iteration). Both the number of patterns and the running time appear to grow no more than linearly with the maximum order of the features for this data set.

3.2 Experiments with High-order Semi-CRFs

We now show that high-order semi-CRFs are also practically useful by evaluating their performance on three different sequence labeling tasks: relation argument detection, punctuation prediction in movie transcripts, and bibliography extraction. We compare high-order semi-CRFs with CRFs of different orders on the same tasks. In our tables, C^k and SC^k

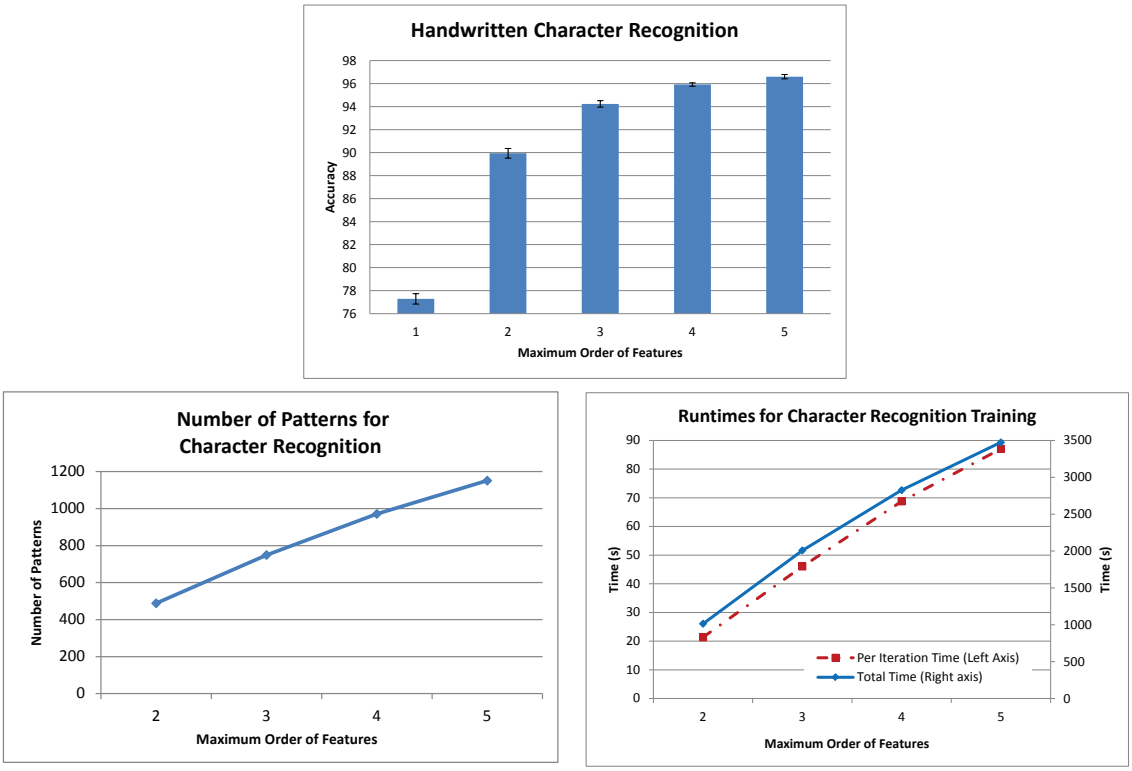


Figure 3: Accuracy (top), number of label patterns (bottom left), and running time (bottom right) as a function of maximum order for the handwriting recognition data set.

refer to k^{th}-order CRF and semi-CRF respectively. We also give the number of segment label patterns and the running time of high-order semi-CRFs on the tasks.

To test if the results obtained by high-order semi-CRFs are significantly better than lower order ones in terms of F1-measure, we perform the randomization tests described by Noreen (1989) and Yeh (2000). In such tests, we shuffle the responses by randomly reassigning the outputs of two systems we are comparing, and see how likely such a shuffle produces a difference in the metric of interest (in our case, the F1-measure). An exact randomization test will iterate through all possible shuffles, but due to the large data sizes, we use an approximate randomization test where for each comparison, we perform 10000 random shuffles, and we repeat this for 999 times. It can be shown (Noreen, 1989; Yeh, 2000) that the significance level p is at most $p' = (n_c+1)/(n_t+1)$, where n_c is the number of trials in which the difference between the F1-measures is greater than the original difference, and n_t is the total number of iterations (in our case, 999). In Table 4, 7, and 9, we summarize the p' obtained in the significance tests. We will comment on these results for each of the three data sets in the following sections.

3.2.1 Relation Argument Detection

In this experiment, we consider the problem of relation argument detection, which identifies and labels arguments of relations in English sentences. More specifically, we construct the label sequence for each sentence as follows: If a word in a sentence is the first argument of a relation, we label it as *Arg1*. If it is the second argument, we label it as *Arg2*. If the word is the first argument of a relation and it is also the second argument of another relation of the same type, we label it as *Arg1Arg2*. Otherwise, we label it as *O*, which means the word is not part of any relation. For example, in the labeled sentence "Peter/Arg1 is/O working/O for/O IBM/Arg2 ./O", *Peter* and *IBM* are arguments of a relation.

It is important to note that if a sentence contains many *Arg1*'s and *Arg2*'s, we do not know which pairs of *Arg1* and *Arg2* would be the actual arguments of a relation. Furthermore, the matching of *Arg1*'s and *Arg2*'s is not one-to-one either, since a word may participate in many different relations of the same type. Thus, to actually extract the relations in a sentence, we would need a separate classifier to determine which pairs of *Arg1* and *Arg2* are the true mentions of a relation. In this experiment, however, we only focus and report on the sentence labeling task.

The relation argument detection problem can be thought of as part of the relation extraction task, which requires extracting some prespecified relationships between named entity mentions. For example, if a person works for an organization, then the person and the organization form an organization-affiliation relation. Previous works on the relation extraction problem usually involve building a classifier to decide whether two named entity mentions are the actual arguments of the relation (GuoDong et al., 2005; Zhang et al., 2006). It may also be beneficial for the classifiers if they can make use of the information obtained from relation argument detection.

We compared the models on the English portion of the Automatic Content Extraction (ACE) 2005 corpus (Walker et al., 2006). The corpus contains articles from six source domains and we group the labeled relations into six types. For the experiment, we trained a separate tagger for each type of relations. The training set contains 70% of the sentences from each source domain. The remaining 30% of the sentences are used for testing. Most sentences do not contain a relation and they make the trained tagger less likely to predict an argument. Hence, we randomly sampled from these negative examples so that the numbers of positive and negative examples are the same. We also assumed the manually annotated named entity mentions are known.

For linear-chain CRF, the zeroth-order features are: surrounding words before and after the current word and their capitalization patterns; letter n-grams in words; surrounding named entity mentions, part-of-speeches before and after the current word and their combinations. The first-order features are: transitions without any observation, transitions with the current or previous words or combinations of their capitalization patterns. The high-order CRFs and semi-CRFs include additional high-order Markov and high-order semi-Markov transition features.

From the results in Table 3, SC^2 gives an improvement of 5.52% on F1 score when compared to SC^1 on average. SC^3 further improves the performance of SC^2 by 0.75% F1 score. High-order CRFs show significant improvement on all except for *PHYS*, which has arguments located further apart compared to other relations. In Table 4, we see that

Type	C^1	C^2	C^3	SC^1	SC^2	SC^3
Part-Whole	38.68	41.41	**46.52**	38.57	42.56	44.30
Phys	33.24	33.60	35.20	33.35	42.04	**42.46**
Org-Aff	60.56	63.28	**64.93**	60.77	63.72	64.86
Gen-Aff	31.00	35.84	**40.16**	31.19	35.85	38.09
Per-Soc	53.67	**58.62**	58.31	53.46	57.66	57.07
Art	40.30	43.80	46.35	40.61	**49.21**	48.78
Average	42.91	46.09	48.58	42.99	48.51	**49.26**

Table 3: F1 scores of different CRF taggers for relation argument detection on six types of relations.

	C^2	C^3	SC^1	SC^2	SC^3
C^1	**0.001**<	**0.001**<	0.226<	**0.001**<	**0.001**<
C^2	–	**0.001**<	**0.001**>	**0.001**<	**0.001**<
C^3	–	–	**0.001**>	0.441>	0.074<
SC^1	–	–	–	**0.001**<	**0.001**<
SC^2	–	–	–	–	0.017<

Table 4: The values of p' obtained in the statistical significance tests comparing CRFs and semi-CRFs of different orders in the relation argument detection task, where the p-value of the significance test is smaller than p'. Figures in bold are where the difference is statistically significant at the 1% confidence level. The symbol $<$ (respectively $>$) at position (i, j) means that the system on row i performs worse (respectively better) than the system on column j.

for this task, first-order semi-CRF does not perform significantly better than simple linear-chain CRF. We also observe that SC^3 outperforms C^1, C^2, and SC^1 significantly, while it outperforms C^3 and SC^2 with p-values at most 7.4% and 1.7% respectively. Figure 4 shows the average number of segment label patterns and the average running time of high-order semi-CRFs as a function of the maximum order.

The CRFs in Table 3 do not use begin-inside-outside (BIO) encoding of the labels. In the labeling protocol described above for this problem, although the label O indicates the outside of any argument, we do not differentiate between the beginning and the insides of an argument. In Table 5, we report the F1 scores of C^1, C^2, and C^3 using BIO encoding (C^1-BIO, C^2-BIO, and C^3-BIO respectively). We use *Arg1-B*, *Arg2-B*, and *Arg1Arg2-B* to indicate the beginning of an argument and use *Arg1-I*, *Arg2-I*, and *Arg1Arg2-I* to indicate the insides of an argument. The scores are computed after removing the B and I suffixes in the labels. From the results in Table 5, BIO encoding does not help C^1-BIO and C^2-BIO much, but it helps to improve C^3-BIO substantially. Overall, C^3-BIO achieves the best average F1 score (51.11%) for the relation argument detection problem. Comparing C^3-BIO and SC^3 on each individual relation, we note that SC^3 is useful for *PHYS* where

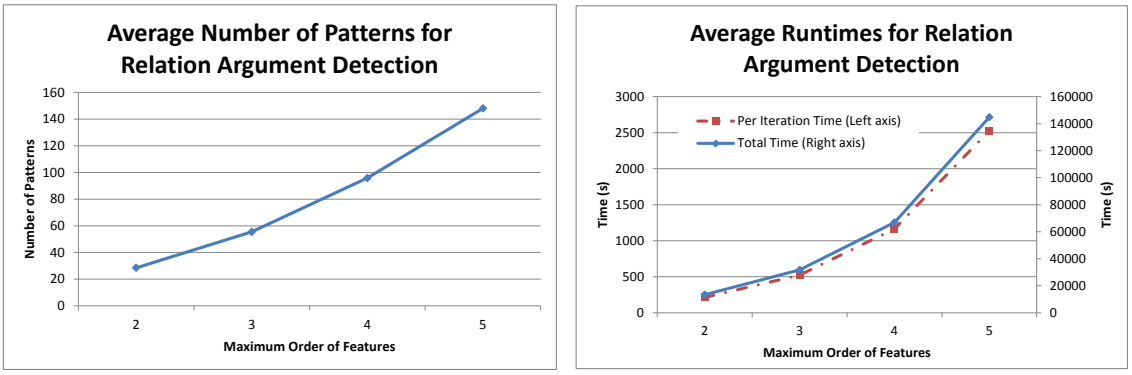


Figure 4: Average number of segment label patterns (left) and average running time (right) of high-order semi-CRFs as a function of maximum order for relation argument detection.

Type	C^1	C^2	C^3	SC^3	C^1-BIO	C^2-BIO	C^3-BIO
Part-Whole	38.68	41.41	46.52	44.30	38.66	41.23	**50.30**
Phys	33.24	33.60	35.20	**42.46**	33.81	34.77	36.88
Org-Aff	60.56	63.28	64.93	64.86	61.33	64.33	**67.50**
Gen-Aff	31.00	35.84	40.16	38.09	30.38	35.03	**43.37**
Per-Soc	53.67	58.62	58.31	57.07	55.07	58.50	**59.37**
Art	40.30	43.80	46.35	48.78	40.62	43.01	**49.25**
Average	42.91	46.09	48.58	49.26	43.31	46.15	**51.11**

Table 5: F1 scores of different (non-semi) CRF taggers for relation argument detection using BIO encoding of the labels (C^1-BIO, C^2-BIO, and C^3-BIO). The scores of C^1, C^2, C^3, and SC^3 are copied from Table 3 for comparison.

the arguments are located further apart. C^3-BIO, on the other hand, is useful for other relations where the arguments are located near to each other.

3.2.2 Punctuation Prediction

In this experiment, we evaluated the performance of high-order semi-CRFs on the punctuation prediction task. This task is usually used as a post-processing step for automatic speech recognition systems to add punctuations to the transcribed conversational speech texts (Liu et al., 2005; Lu and Ng, 2010). Previous evaluations on the IWSLT corpus (Paul, 2009) have shown that capturing long-range dependencies is useful for the task (Lu and Ng, 2010). In the experiment, we used high-order CRFs and high-order semi-CRFs to capture long-range dependencies in the labels and showed that they outperform linear-chain CRF and first-order semi-CRF on movie transcripts data, which contains 5450 conversational speech texts with annotated punctuations from various movie transcripts online. We used

Tag	C^1	C^2	C^3	SC^1	SC^2	SC^3
Comma	59.29	59.70	59.90	**61.13**	60.89	60.35
Period	75.37	75.37	75.46	75.03	**78.97**	78.82
QMark	58.18	59.54	60.57	57.61	**74.05**	73.56
All	66.21	66.53	66.85	66.73	**70.85**	70.47

Table 6: F1 scores for punctuation prediction task. The last row contains the micro-averaged scores.

	C^2	C^3	SC^1	SC^2	SC^3
C^1	0.155<	0.048<	0.043<	**0.001**<	**0.001**<
C^2	–	0.153<	0.289<	**0.001**<	**0.001**<
C^3	–	–	0.378>	**0.001**<	**0.001**<
SC^1	–	–	–	**0.001**<	**0.001**<
SC^2	–	–	–	–	0.044>

Table 7: The values of p' obtained in the statistical significance tests comparing CRFs and semi-CRFs of different orders in the punctuation prediction task, where the p-value of the significance test is smaller than p'. Figures in bold are where the difference is statistically significant at the 1% confidence level. The symbol $<$ (respectively $>$) at position (i, j) means that the system on row i performs worse (respectively better) than the system on column j.

60% of the texts for training and the remaining 40% for testing. The punctuation and case information are removed, and the words are tagged with different labels.

Originally, there are 4 labels: None, Comma, Period, and QMark, which respectively indicate that no punctuation, a comma, a period, or a question mark comes immediately after the current word. To help capture the long-range dependencies, we added 6 more labels: None-Comma, None-Period, None-QMark, Comma-Comma, QMark-QMark, and Period-Period. The left parts of these labels serve the same purpose as the original four labels. The right parts of the labels indicate that the current word is the beginning of a text segment which ends in comma, period, or question mark. This part is used to capture useful information at the beginning of the segment. For example, the sentence "no, she is working." would be labeled as "no/Comma-Comma she/None-Period is/None working/Period". In this case, *she is working* is a text segment (with length 3) that ends with a period. This information is marked in the label of the word *working* and the right part of the label of the word *she*. The text segment *no* (with length 1) is also labeled in a similar way.

We reported the F1 scores of the models in Table 6. We used the combinations of words and their positions relatively to the current position as zeroth-order features. For first-order features, we used transitions without any observation, and transitions with the current or previous words, as well as their combinations. C^k uses k^{th}-order Markov features, while

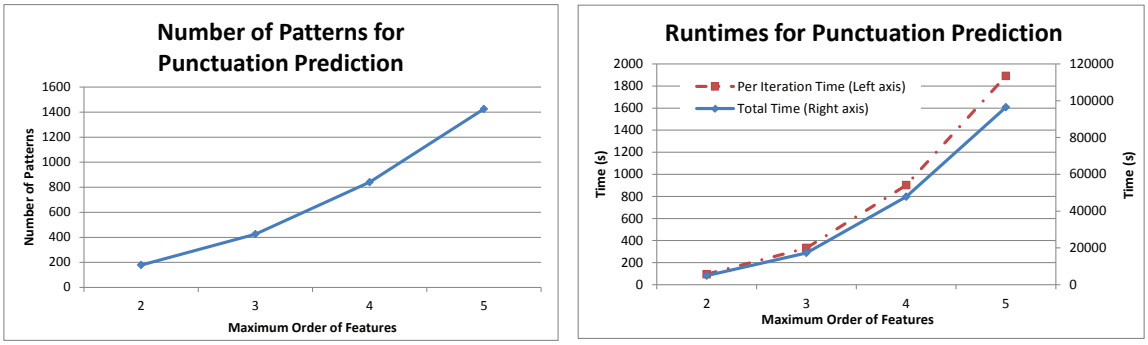


Figure 5: Number of segment label patterns (left) and running time (right) of high-order semi-CRFs as a function of maximum order for the punctuation prediction data set.

SC^k uses k^{th}-order semi-Markov transition features with the observed words in the last segment.

The scores reported in Table 6 are lower than those of the IWSLT corpus (Lu and Ng, 2010) because online movie transcripts are usually annotated by different people, and they tend to put the punctuations slightly differently. Besides, in movies, people sometimes use declarative sentences as questions. Hence, the punctuations are harder to predict. Nevertheless, the results have clearly shown that high-order semi-CRFs can capture long-range dependencies with the help of additional labels and can achieve more than 3.6% improvement in F1 score compared to the CRFs and first-order semi-CRF. SC^k also outperforms C^k for all k. For this task, using third-order semi-Markov features decrease the performance of SC^3 slightly compared to SC^2. From Table 7, we see that the p-value of the statistical significance test comparing SC^2 and SC^3 is at most 4.4%, while both SC^2 and SC^3 significantly outperform the other models. Figure 5 shows the number of segment label patterns and the running time of high-order semi-CRFs as a function of the maximum order.

3.2.3 Bibliography Extraction

In this experiment, we consider the problem of bibliography extraction in scientific papers. For this problem, we need to divide a reference, such as those appearing in the *References* section of this paper, into the following 13 types of segments: Author, Booktitle, Date, Editor, Institution, Journal, Location, Note, Pages, Publisher, Tech, Title, or Volume. The problem can be naturally considered as a sequence labeling problem with the above labels. We evaluated the performance of high-order semi-CRFs and CRFs on the bibliography extraction problem with the Cora Information Extraction data set.[4] In the data set, there are 500 instances of references. We used 300 instances for training and the remaining 200 instances for testing.

We reported in Table 8 the F1 scores of the models. In C^1, zeroth-order features include the surrounding words at each position and letter n-grams, and first-order features include

4. The data set is available at `http://people.cs.umass.edu/~mccallum/data.html`.

Tag	C^1	C^2	C^3	SC^1	SC^2	SC^3
Author	94.21	91.65	93.67	93.97	**94.74**	94.00
Booktitle	73.05	75.00	72.39	75.74	**78.11**	76.47
Date	95.67	**96.68**	94.36	95.19	95.43	95.70
Editor	68.57	**72.73**	66.67	57.14	58.82	54.55
Institution	68.57	64.71	64.71	**70.27**	**70.27**	64.86
Journal	78.08	**78.32**	**78.32**	77.55	77.55	75.68
Location	**70.33**	69.66	68.13	68.13	67.39	65.22
Note	**66.67**	57.14	57.14	57.14	**66.67**	**66.67**
Pages	84.82	**87.83**	85.34	85.96	86.96	87.18
Publisher	84.62	84.62	83.54	84.62	**86.08**	**86.08**
Tech	77.78	**80.00**	**80.00**	77.78	77.78	77.78
Title	89.62	85.42	86.73	90.18	**92.23**	90.95
Volume	66.23	**75.68**	72.60	71.90	72.37	75.00
All	85.34	85.47	84.77	85.67	**86.67**	86.07

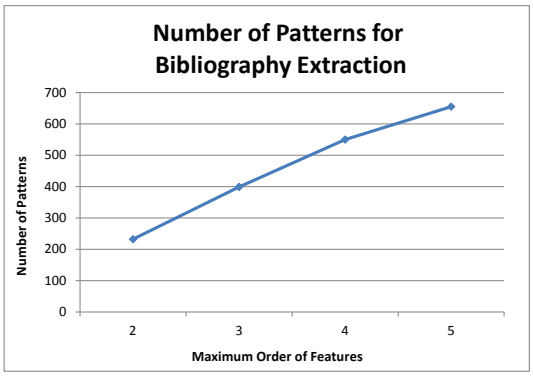
Table 8: F1 scores for bibliography extraction task. The last row contains the micro-averaged scores.

	C^2	C^3	SC^1	SC^2	SC^3
C^1	0.393<	0.174>	0.198<	**0.004**<	0.095<
C^2	–	0.073>	0.351<	0.019<	0.161<
C^3	–	–	0.095<	**0.002**<	0.030<
SC^1	–	–	–	**0.003**<	0.200<
SC^2	–	–	–	–	0.025>

Table 9: The values of p' obtained in the statistical significance tests comparing CRFs and semi-CRFs of different orders in the bibliography extraction task, where the p-value of the significance test is smaller than p'. Figures in bold are where the difference is statistically significant at the 1% confidence level. The symbol $<$ (respectively $>$) at position (i, j) means that the system on row i performs worse (respectively better) than the system on column j.

transitions with words at the current or previous positions. C^k and SC^k $(1 \leq k \leq 3)$ use additional k^{th}-order Markov and semi-Markov transition features.

From Table 8, high-order semi-CRFs perform generally better than high-order CRFs and first-order semi-CRF. SC^2 achieves the best overall performance with 86.67% F1 score. From Table 9, SC^2 outperforms C^2 and SC^3 with a p-value at most 1.9% and 2.5% respectively, while it outperforms other models significantly. Figure 6 shows the number of segment label patterns and the running time of high-order semi-CRFs as a function of the maximum order.



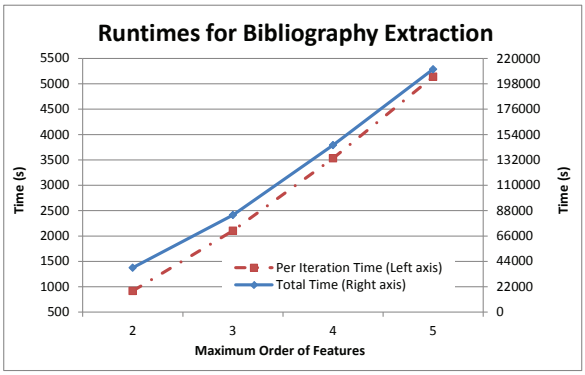


Figure 6: Number of segment label patterns (left) and running time (right) of high-order semi-CRFs as a function of maximum order for the bibliography extraction data set.

3.3 Discussions

From Figures 4, 5, and 6, the number of segment label patterns of high-order features grows about linearly in the maximum order of features. The running time of high-order semi-CRFs on the bibliography extraction task is also nearly linear in the maximum order of the features, while the running times on the relation argument detection task and the punctuation prediction task grow more than linearly in the maximum order of features. We also note that from the time complexity discussions in Section 2.5 and the setup for these experiments, the time complexity of our algorithm is $O(|\mathcal{Z}|^2)$, where $|\mathcal{Z}|$ is the number of segment label patterns.

From Tables 6 and 8, there is a drop in F1 scores for the punctuation prediction task and the bibliography extraction task when we increase the order of the semi-CRFs from 2 to 3. For the punctuation task, the drop is not very significant and the third-order semi-CRF still performs significantly better than the CRFs or the first-order semi-CRFs. For the bibliography extraction task, there is a big drop in the F1 scores for some of the labels and the third-order semi-CRF does not significantly outperform the other models. However, it does not indicate that the third-order semi-CRF is not useful for this task since we fixed the regularization parameter $\sigma_{\text{reg}} = 1$ for all the models in this experiment. If we set $\sigma_{\text{reg}} = 10$ for the third-order semi-CRF, it can achieve 87.45% F1 score and outperform all the other models. In practice, if we have enough data, we can choose a suitable σ_{reg} for each individual model using a validation data set or cross-validation on the training data. We can also allow different regularizers for features of different orders[5] and use a validation set to determine the most suitable combination of regularizers.

An important question in practice is which features (or equivalently, label patterns) should be included in the model. In our experiments, we used all the label patterns that appear in the training data. This simple approach is usually reasonable with a suitable value of the regularization parameter σ_{reg}. For applications where the pattern sparsity assumption is not satisfied, but certain patterns do not appear frequently enough and are

5. This would require a slight change to our regularized log-likelihood function.

not really important, then it is useful to see how we can select a subset of features with few distinct label patterns automatically. One possible approach would be to use boosting type methods (Dietterich et al., 2004) to sequentially select useful features.

For high-order CRFs, it should be possible to use kernels within the approach here. On the handwritten character problem, Taskar et al. (2004) reported substantial improvement in performance with the use of kernels. Use of kernels together with high-order features may lead to further improvements. However, we note that the advantage of the higher order features may become less substantial as the observations become more powerful in distinguishing the classes. Whether the use of higher order features together with kernels brings substantial improvement in performance is likely to be problem dependent.

4. Related Work

A commonly used inference algorithm for CRFs is the clique tree algorithm (Huang and Darwiche, 1996). Defining a feature depending on k (not necessarily consecutive) labels will require forming a clique of size k, resulting in a clique-tree with tree-width greater or equal to k. Inference on such a clique tree will be exponential in k. For sequence models, a feature of order k can be incorporated into a k^{th}-order Markov chain, but the complexity of inference is again exponential in k. Under the label pattern sparsity assumption, our algorithm achieves efficiency by maintaining only information related to a few occurred patterns, while previous algorithms maintain information about all (exponentially many) possible patterns.

Long distance dependencies can also be captured using hierarchical models such as Hierarchical Hidden Markov Model (HHMM) (Fine et al., 1998) or Probabilistic Context Free Grammar (PCFG) (Heemskerk, 1993). The time complexity of inference in an HHMM is $O(\min\{nl^3, n^2l\})$ (Fine et al., 1998; Murphy and Paskin, 2002), where n is the number of states and l is the length of the sequence. Discriminative versions such as hierarchical semi-CRF have also been studied (Truyen et al., 2008). Inference in PCFG and its discriminative version can also be efficiently done in $O(ml^3)$ where m is the number of productions in the grammar (Jelinek et al., 1992). These methods are able to capture dependencies of arbitrary lengths, unlike k^{th}-order Markov chains. However, to do efficient learning with these methods, the hierarchical structure of the examples needs to be provided. For example, if we use PCFG to do character sequence labeling, we need to provide the parse trees for efficient learning; providing the labels for each character is not sufficient. Hence, a training set that has not been labeled with hierarchical labels will need to be relabeled before it can be trained efficiently. Alternatively, methods that employ hidden variables can be used (e.g., to infer the hidden parse tree) but the optimization problem is no longer convex and local optima can sometimes be a problem. The high-order semi-CRF presented in this paper allows us to capture a different class of dependencies that does not depend on hierarchical structures in the data, while keeping the high-order semi-CRF objective a convex optimization problem.

Another work on using high-order features for CRFs was independently done by Qian et al. (2009). Their work applies to a larger class of CRFs, including those requiring exponential time for inference, and they did not identify subclasses for which inference is guaranteed to be efficient. For sequence labeling with high-order features, Qian and Liu

(2012) developed an efficient decoding algorithm under the assumption that all the high-order features have non-negative weights. Their decoding algorithm requires quadratic running time on the number of high-order features in the worst case.

There are other models similar to the high-order CRF with pattern sparsity assumption (Ye et al., 2009), a special case of the high-order semi-CRF presented in this paper. They include the CRFs that use the sparse higher-order potentials (Rother et al., 2009) or the pattern-based potentials (Komodakis and Paragios, 2009). Rother et al. (2009) proposed a method for minimization of sparse higher order energy functions by first transforming them into a quadratic functions and then employing efficient inference algorithms to minimize these resulting functions. For the pattern-based potentials, Komodakis and Paragios (2009) derived an efficient message-passing algorithm for inference. The algorithm is based on the master-slave framework where the original high-order optimization problem is decomposed into smaller subproblems that can be solved easily. Other tractable inference algorithms with high-order potentials include the α-expansion and $\alpha\beta$-swap algorithms for the $\mathcal{P}^n$ Potts model (Kohli et al., 2007) and the MAP message passing algorithm for cardinality and order potentials (Tarlow et al., 2010). A special case of the order potentials, the before-after potential (Tarlow et al., 2010), can also be used to capture some semi-Markov structures in the data labelings.

5. Conclusion

The label pattern sparsity assumption often holds in real applications, and we give efficient inference algorithms for CRFs using high-order dependencies between labels or segments when the pattern sparsity assumption is satisfied. This allows high-order features to be explored in feature engineering for real applications. We studied the conditions that are favorable for using high-order features in CRFs with a synthetic data set, and demonstrated that using simple high-order features can lead to performance improvement on a handwriting recognition problem. We also demonstrated that high-order semi-CRFs outperform high-order CRFs and first-order semi-CRF in segmentation problems like relation argument detection, punctuation prediction, and bibliography extraction.

Acknowledgments

This material is based on research sponsored by DSO under grant DSOCL11102 and by the Air Force Research Laboratory, under agreement number FA2386-09-1-4123. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government. The authors also would like to thank Sumit Bhagwani for his help with the HOSemiCRF package and the anonymous reviewers for their constructive comments.

Appendix A. Correctness of the Forward and Backward Algorithms

In this appendix, we will prove the correctness of the forward and backward algorithms described in Section 2. We shall prove two lemmas and then provide the proofs for the correctness of the forward and backward algorithms as well as the marginal computation.

Lemma 1 below gives the key properties that can be used in an inductive proof. Lemma 1(a) shows that we can partition the segmentations using the forward states. Lemma 1(b-c) show that considering all $(\mathbf{p}^k, y) : \mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{p}^k y$ is sufficient for obtaining the sum over all sequences $\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{z}y$, while Lemma 1(d) is used to show that the features are counted correctly.

Lemma 1 *Let* $\mathbf{s}$ *be a segmentation for a prefix of* $\mathbf{x}$. *Let* $\omega(\mathbf{s}, t) = \exp(\sum_{k=1}^m \lambda_k f_k(\mathbf{x}, \mathbf{s}, t))$ *and* $\omega(\mathbf{s}) = \exp(\sum_{k=1}^m \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t)) = \prod_{t=1}^{|\mathbf{s}|} \omega(\mathbf{s}, t)$.
(a) For any segment label sequence $\mathbf{z}$, *there exists a unique* $\mathbf{p}^i \in \mathcal{P}$ *such that* $\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{z}$.
(b) For any segment label sequence $\mathbf{z}$ *and* $y \in \mathcal{Y}$, *if* $\mathbf{p}^k \leq^s_{\mathcal{P}} \mathbf{z}$ *and* $\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{p}^k y$, *then* $\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{z}y$.
(c) For any $\mathbf{z}^a \in \mathcal{Z}$, $y \in \mathcal{Y}$, *and any segment label sequence* $\mathbf{z}$, *if* $\mathbf{z}^a \leq^s \mathbf{z}y$, *and* $\mathbf{p}^k \leq^s_{\mathcal{P}} \mathbf{z}$, *then* $\mathbf{z}^a \leq^s \mathbf{p}^k y$.
(d) Let $\mathbf{s} = ((u_1, v_1, y_1), \ldots, (u_{|\mathbf{s}|}, v_{|\mathbf{s}|}, y_{|\mathbf{s}|}))$ *and let* $\mathbf{p}^{k_t} \leq^s_{\mathcal{P}} y_1 y_2 \ldots y_t$ *for* $t = 1, \ldots, |\mathbf{s}|$. *Then* $\omega(\mathbf{s}) = \prod_{t=1}^{|\mathbf{s}|} \Psi_{\mathbf{x}}(u_t, v_t, \mathbf{p}^{k_{t-1}} y_t) = \omega(\mathbf{s}_{1:|\mathbf{s}|-1}) \Psi_{\mathbf{x}}(u_{|\mathbf{s}|}, v_{|\mathbf{s}|}, \mathbf{p}^{k_{|\mathbf{s}|-1}} y_{|\mathbf{s}|})$.

A.1 Proof of Lemma 1

(a) The intersection of $\mathcal{P}$ and the set of prefixes of $\mathbf{z}$ contains at least one element ϵ, and is finite.

(b) We have $\mathbf{p}^i \leq^s \mathbf{p}^k y \leq^s \mathbf{z}y$. Furthermore, if $\mathbf{p}^j \leq^s \mathbf{z}y$, then we have $\mathbf{p}^j_{1:|\mathbf{p}^j|-1} \leq^s \mathbf{z}$. Thus, $\mathbf{p}^j_{1:|\mathbf{p}^j|-1} \leq^s \mathbf{p}^k$ since $\mathbf{p}^k \leq^s_{\mathcal{P}} \mathbf{z}$. Hence, $\mathbf{p}^j = \mathbf{p}^j_{1:|\mathbf{p}^j|-1} y \leq^s \mathbf{p}^k y$. Since $\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{p}^k y$, we have $\mathbf{p}^j \leq^s \mathbf{p}^i$. Therefore, $\mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{z}y$.

(c) Since $\mathbf{z}^a_{1:|\mathbf{z}^a|-1} \leq^s \mathbf{z}$ and $\mathbf{p}^k \leq^s_{\mathcal{P}} \mathbf{z}$, we have $\mathbf{z}^a_{1:|\mathbf{z}^a|-1} \leq^s \mathbf{p}^k$. Thus, $\mathbf{z}^a \leq^s \mathbf{p}^k y$.

(d) Straightforward from part (c) and definition of $\Psi_{\mathbf{x}}$.

Lemma 2 below serves the same purpose as Lemma 1 for showing correctness.

Lemma 2 *Let* $\mathbf{s}$ *be a segmentation for a suffix of* $\mathbf{x}$. *Let* $\omega(\mathbf{s}, t|\mathbf{s}^i) = \exp(\sum_{k=1}^m \lambda_k f_k(\mathbf{x}, \mathbf{s}, t|\mathbf{s}^i))$ *and* $\omega(\mathbf{s}|\mathbf{s}^i) = \exp(\sum_{k=1}^m \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t|\mathbf{s}^i)) = \prod_{t=1}^{|\mathbf{s}|} \omega(\mathbf{s}, t|\mathbf{s}^i)$.
(a) For all $\mathbf{s}^i \in \mathcal{S}$ *and* $y \in \mathcal{Y}$, *there exists a unique* $\mathbf{s}^k \in \mathcal{S}$ *such that* $\mathbf{s}^k \leq^s_{\mathcal{S}} \mathbf{s}^i y$.
(b) For any $\mathbf{z}^a \in \mathcal{Z}$ *and any segment label sequences* $\mathbf{z}_1, \mathbf{z}_2$, *if* $\mathbf{z}^a \leq^s \mathbf{z}_1\mathbf{z}_2$, *and* $\mathbf{s}^i \leq^s_{\mathcal{S}} \mathbf{z}_1$, *then* $\mathbf{z}^a \leq^s \mathbf{s}^i \mathbf{z}_2$.
(c) If $\mathbf{s}^k \leq^s_{\mathcal{S}} \mathbf{s}^i y$, *and* $(u, v, y) \cdot \mathbf{s}$ *is a segmentation for* $\mathbf{x}_{u:|\mathbf{x}|}$, *then* $\omega((u, v, y) \cdot \mathbf{s}|\mathbf{s}^i) = \Psi_{\mathbf{x}}(u, v, \mathbf{s}^i y)\omega(\mathbf{s}|\mathbf{s}^k)$.

A.2 Proof of Lemma 2

(a) Note that $y \in \mathcal{S}$ and $y \leq^s \mathbf{s}^i y$ and the number of suffixes of $\mathbf{s}^i y$ is finite.

(b) This is clearly true if $\mathbf{z}^a$ is not longer than $\mathbf{z}_2$. If $\mathbf{z}^a$ is longer than $\mathbf{z}_2$, let $\mathbf{p}$ be the prefix of $\mathbf{z}^a$ obtained by stripping off the suffix $\mathbf{z}_2$. Then $\mathbf{p}$ is a suffix of $\mathbf{z}_1$ and $\mathbf{p} \in \mathcal{S}$. Since $\mathbf{s}^i$ is the longest suffix of $\mathbf{z}_1$ in $\mathcal{S}$, $\mathbf{p}$ is a suffix of $\mathbf{s}^i$, thus $\mathbf{z}^a = \mathbf{p}\mathbf{z}_2$ is a suffix of $\mathbf{s}^i\mathbf{z}_2$.

(c) From part (b), we have $\omega(\mathbf{s}|\mathbf{s}^i y) = \omega(\mathbf{s}|\mathbf{s}^k)$. Thus, $\omega((u,v,y)\cdot\mathbf{s}|\mathbf{s}^i) = \Psi_{\mathbf{x}}(u,v,\mathbf{s}^i y)\omega(\mathbf{s}|\mathbf{s}^i y) = \Psi_{\mathbf{x}}(u,v,\mathbf{s}^i y)\omega(\mathbf{s}|\mathbf{s}^k)$.

A.3 Correctness of the Forward Algorithm

Given the forward variables $\alpha_{\mathbf{x}}(j,\mathbf{p}^i)$ as defined in Section 2

$$\alpha_{\mathbf{x}}(j,\mathbf{p}^i) = \sum_{\mathbf{s}\in\mathbf{p}_{j,\mathbf{p}^i}} \exp(\sum_{k=1}^{m}\sum_{t=1}^{|\mathbf{s}|}\lambda_k f_k(\mathbf{x},\mathbf{s},t)) = \sum_{\mathbf{s}\in\mathbf{p}_{j,\mathbf{p}^i}} \omega(\mathbf{s}),$$

we prove that the following recurrence can be used to compute $\alpha_{\mathbf{x}}(j,\mathbf{p}^i)$'s by induction on j,

$$\alpha_{\mathbf{x}}(j,\mathbf{p}^i) = \sum_{d=0}^{L-1}\sum_{(\mathbf{p}^k,y):\mathbf{p}^i\le_{\mathcal{P}}^{s}\mathbf{p}^k y} \Psi_{\mathbf{x}}(j-d,j,\mathbf{p}^k y)\alpha_{\mathbf{x}}(j-d-1,\mathbf{p}^k). \quad (2)$$

Base case: If $j=1$, for any $\mathbf{p}^i \in \mathcal{P}$, we can initialize the values of $\alpha_{\mathbf{x}}(1,\mathbf{p}^i)$ such that

$$\alpha_{\mathbf{x}}(1,\mathbf{p}^i) = \sum_{\mathbf{s}\in\mathbf{p}_{1,\mathbf{p}^i}} \exp(\sum_{k=1}^{m}\sum_{t=1}^{|\mathbf{s}|}\lambda_k f_k(\mathbf{x},\mathbf{s},t)) = \sum_{\mathbf{s}\in\mathbf{p}_{1,\mathbf{p}^i}} \omega(\mathbf{s}).$$

Inductive step: Assume that for all $j' < j$ and $\mathbf{p}^i \in \mathcal{P}$, we have

$$\alpha_{\mathbf{x}}(j',\mathbf{p}^i) = \sum_{\mathbf{s}\in\mathbf{p}_{j',\mathbf{p}^i}} \exp(\sum_{k=1}^{m}\sum_{t=1}^{|\mathbf{s}|}\lambda_k f_k(\mathbf{x},\mathbf{s},t)) = \sum_{\mathbf{s}\in\mathbf{p}_{j',\mathbf{p}^i}} \omega(\mathbf{s}).$$

Then, using Lemma 1,

$$\begin{aligned}
\alpha_{\mathbf{x}}(j,\mathbf{p}^i) &= \textstyle\sum_{\mathbf{s}\in\mathbf{p}_{j,\mathbf{p}^i}} \omega(\mathbf{s}) \\
&= \textstyle\sum_{d=0}^{L-1}\sum_{(\mathbf{p}^k,y):\mathbf{p}^i\le_{\mathcal{P}}^{s}\mathbf{p}^k y}\sum_{\mathbf{s}\in\mathbf{p}_{j-d-1,\mathbf{p}^k}} \omega(\mathbf{s}\cdot(j-d,j,y)) \\
&= \textstyle\sum_{d=0}^{L-1}\sum_{(\mathbf{p}^k,y):\mathbf{p}^i\le_{\mathcal{P}}^{s}\mathbf{p}^k y}\sum_{\mathbf{s}\in\mathbf{p}_{j-d-1,\mathbf{p}^k}} [\Psi_{\mathbf{x}}(j-d,j,\mathbf{p}^k y)\textstyle\prod_{t=1}^{|\mathbf{s}|}\omega(\mathbf{s},t)] \\
&= \textstyle\sum_{d=0}^{L-1}\sum_{(\mathbf{p}^k,y):\mathbf{p}^i\le_{\mathcal{P}}^{s}\mathbf{p}^k y}\Psi_{\mathbf{x}}(j-d,j,\mathbf{p}^k y)\sum_{\mathbf{s}\in\mathbf{p}_{j-d-1,\mathbf{p}^k}}\textstyle\prod_{t=1}^{|\mathbf{s}|}\omega(\mathbf{s},t) \\
&= \textstyle\sum_{d=0}^{L-1}\sum_{(\mathbf{p}^k,y):\mathbf{p}^i\le_{\mathcal{P}}^{s}\mathbf{p}^k y}\Psi_{\mathbf{x}}(j-d,j,\mathbf{p}^k y)\alpha_{\mathbf{x}}(j-d-1,\mathbf{p}^k).
\end{aligned}$$

Hence, by induction, Recurrence (2) correctly computes the forward variables $\alpha_{\mathbf{x}}(j,\mathbf{p}^i)$'s.

A.4 Correctness of the Backward Algorithm

Given the backward variables $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$ as defined in Section 2

$$\beta_{\mathbf{x}}(j, \mathbf{s}^i) = \sum_{\mathbf{s} \in \mathbf{s}_j} \exp(\sum_{k=1}^{m} \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t | \mathbf{s}^i)) = \sum_{\mathbf{s} \in \mathbf{s}_j} \omega(\mathbf{s}|\mathbf{s}^i),$$

we prove that the following recurrence can be used to compute $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$'s by induction on j,

$$\beta_{\mathbf{x}}(j, \mathbf{s}^i) = \sum_{d=0}^{L-1} \sum_{(\mathbf{s}^k, y): \mathbf{s}^k \leq_{\mathcal{S}}^{s} \mathbf{s}^i y} \Psi_{\mathbf{x}}(j, j+d, \mathbf{s}^i y) \beta_{\mathbf{x}}(j+d+1, \mathbf{s}^k). \tag{3}$$

Base case: If $j = |\mathbf{x}|$, for any $\mathbf{s}^i \in \mathcal{S}$, we can initialize the values of $\beta_{\mathbf{x}}(|\mathbf{x}|, \mathbf{s}^i)$ such that

$$\beta_{\mathbf{x}}(|\mathbf{x}|, \mathbf{s}^i) = \sum_{\mathbf{s} \in \mathbf{s}_{|\mathbf{x}|}} \exp(\sum_{k=1}^{m} \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t | \mathbf{s}^i)) = \sum_{\mathbf{s} \in \mathbf{s}_{|\mathbf{x}|}} \omega(\mathbf{s}|\mathbf{s}^i).$$

Inductive step: Assume that for all $j' > j$ and $\mathbf{s}^i \in \mathcal{S}$, we have

$$\beta_{\mathbf{x}}(j', \mathbf{s}^i) = \sum_{\mathbf{s} \in \mathbf{s}_{j'}} \exp(\sum_{k=1}^{m} \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t | \mathbf{s}^i)) = \sum_{\mathbf{s} \in \mathbf{s}_{j'}} \omega(\mathbf{s}|\mathbf{s}^i).$$

Then, using Lemma 2,

$$\begin{aligned}
\beta_{\mathbf{x}}(j, \mathbf{s}^i) &= \textstyle\sum_{\mathbf{s} \in \mathbf{s}_j} \omega(\mathbf{s}|\mathbf{s}^i) \\
&= \textstyle\sum_{d=0}^{L-1} \sum_{(\mathbf{s}^k, y): \mathbf{s}^k \leq_{\mathcal{S}}^{s} \mathbf{s}^i y} \sum_{\mathbf{s} \in \mathbf{s}_{j+d+1}} \omega((j, j+d, y) \cdot \mathbf{s}|\mathbf{s}^i) \\
&= \textstyle\sum_{d=0}^{L-1} \sum_{(\mathbf{s}^k, y): \mathbf{s}^k \leq_{\mathcal{S}}^{s} \mathbf{s}^i y} \sum_{\mathbf{s} \in \mathbf{s}_{j+d+1}} \Psi_{\mathbf{x}}(j, j+d, \mathbf{s}^i y) \omega(\mathbf{s}|\mathbf{s}^k) \\
&= \textstyle\sum_{d=0}^{L-1} \sum_{(\mathbf{s}^k, y): \mathbf{s}^k \leq_{\mathcal{S}}^{s} \mathbf{s}^i y} \Psi_{\mathbf{x}}(j, j+d, \mathbf{s}^i y) \beta_{\mathbf{x}}(j+d+1, \mathbf{s}^k).
\end{aligned}$$

Hence, by induction, Recurrence (3) correctly computes the backward variables $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$'s.

A.5 Correctness of the Marginal Computation

Consider a segmentation $\mathbf{s}$ such that the segment label sequence of $\mathbf{s}$ contains $\mathbf{z}$ as a subsequence with the last segment of $\mathbf{z}$ having boundaries (u, v). Suppose $\mathbf{s} = \mathbf{s}_1 \cdot (u, v, y) \cdot \mathbf{s}_2$ and let $\mathbf{y}_1$ be the segment label sequence of $\mathbf{s}_1$. If $\mathbf{p}^i \leq_{\mathcal{P}}^{s} \mathbf{y}_1$, then we have $\mathbf{p}^i y \leq_{\mathcal{S}}^{s} \mathbf{y}_1 y$. In this case, it can be verified that $\omega(\mathbf{s}) = \omega(\mathbf{s}_1) \Psi(u, v, \mathbf{p}^i y) \omega(\mathbf{s}_2 | \mathbf{p}^i y)$. The marginal formula thus follows easily.

Appendix B. An Example for the Algorithms

In this appendix, we give an example to illustrate our algorithms. For simplicity, we use the second-order CRF as our model. Extensions to higher-order CRFs or semi-CRFs should be straightforward by respectively expanding the set of segment label patterns or summing over all the possible lengths d of the segments.

i	$f_i(\mathbf{x}, \mathbf{s}, t)$
1	$x_t = Peter \wedge s_t = P$
2	$x_t = goes \wedge s_t = O$
3	$x_t = to \wedge s_t = O$
4	$x_t = Britain \wedge s_t = L$
5	$x_t = and \wedge s_t = O$
6	$x_t = France \wedge s_t = L$
7	$x_t = annually \wedge s_t = O$
8	$x_t = . \wedge s_t = O$
9	$s_{t-2}s_{t-1}s_t = LOL$

Table 10: List of features for the example in Appendix B.

$t\backslash\mathbf{z}$	P	O	L	LOL
1	1	0	0	0
2	0	1	0	0
3	0	1	0	0
4	0	0	1	0
5	0	1	0	0
6	0	0	1	1
7	0	1	0	0
8	0	1	0	0

Table 11: The values of $\sum_{i:\mathbf{z}^i=\mathbf{z}} \lambda_i g_i(\mathbf{x}, u_t, v_t) = \sum_{i:\mathbf{z}^i=\mathbf{z}} \lambda_i g_i(\mathbf{x}, t, t)$.

In this example, let $\mathbf{x}$ be the sentence "Peter goes to Britain and France annually.". Assume there are 9 binary features defined by Boolean predicates as in Table 10, and each $\lambda_i = 1$. The label set is $\{P, O, L\}$ where P represents *Person*, L represents *Location* and O represents *Others*. Note that for second-order CRFs, the length of all the segments is 1 and thus $s_t = y_t$ for all t.

The segment label pattern set is $\mathcal{Z} = \{P, O, L, LOL\}$. Table 11 shows the sum of the weights for features with the same segment label pattern at each position. We have $\mathcal{P} = \{\epsilon, P, O, L, LO\}$ and $\mathcal{S} = \{P, O, L, PP, PO, PL, OP, OO, OL, LP, LO, LL, LOP, LOO, LOL\}$. The tables for $\ln \alpha_{\mathbf{x}}$ and $\ln \beta_{\mathbf{x}}$ are shown in Table 12 and Table 13 respectively.

In Figure 7, we give a diagram to show the messages passed from step $j-1$ to step j to compute the forward variables $\alpha_{\mathbf{x}}$. We also give a diagram in Figure 8 to show some messages passed from step $j+1$ to step j to compute the backward variables $\beta_{\mathbf{x}}$.

We illustrate the computation of $\alpha_{\mathbf{x}}$ with $\alpha_{\mathbf{x}}(6, L)$. The condition $(\mathbf{p}^k, y) : \mathbf{p}^i \leq^s_{\mathcal{P}} \mathbf{p}^k y$ with $\mathbf{p}^i = L$ gives us the following 5 pairs as $(\mathbf{p}^k, y)$: $\{(\epsilon, L), (P, L), (O, L), (L, L), (LO, L)\}$. Thus,

$$\begin{aligned}
\alpha_{\mathbf{x}}(6, L) &= \alpha_{\mathbf{x}}(5, \epsilon)\Psi_{\mathbf{x}}(6, 6, L) + \alpha_{\mathbf{x}}(5, P)\Psi_{\mathbf{x}}(6, 6, PL) + \alpha_{\mathbf{x}}(5, O)\Psi_{\mathbf{x}}(6, 6, OL) + \\
&\quad \alpha_{\mathbf{x}}(5, L)\Psi_{\mathbf{x}}(6, 6, LL) + \alpha_{\mathbf{x}}(5, LO)\Psi_{\mathbf{x}}(6, 6, LOL) \\
&= 0\Psi_{\mathbf{x}}(6, 6, L) + \alpha_{\mathbf{x}}(5, P)e + \alpha_{\mathbf{x}}(5, O)e + \alpha_{\mathbf{x}}(5, L)e + \alpha_{\mathbf{x}}(5, LO)e^2.
\end{aligned}$$

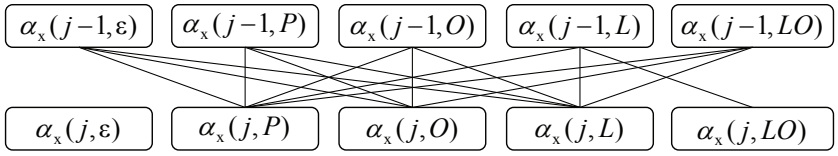


Figure 7: Messages passed from step $j-1$ to step j in order to compute the forward variables. For example, $\alpha_{\mathbf{x}}(j,O)$ is computed from $\alpha_{\mathbf{x}}(j-1,\epsilon)$, $\alpha_{\mathbf{x}}(j-1,P)$, $\alpha_{\mathbf{x}}(j-1,O)$, and $\alpha_{\mathbf{x}}(j-1,LO)$.

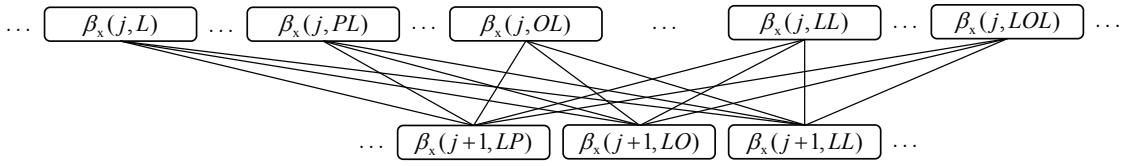


Figure 8: Some messages passed from step $j+1$ to step j in order to compute the backward variables. In this example, all the variables $\beta_{\mathbf{x}}(j,L)$, $\beta_{\mathbf{x}}(j,PL)$, $\beta_{\mathbf{x}}(j,OL)$, $\beta_{\mathbf{x}}(j,LL)$, and $\beta_{\mathbf{x}}(j,LOL)$ are computed from $\beta_{\mathbf{x}}(j+1,LP)$, $\beta_{\mathbf{x}}(j+1,LO)$, and $\beta_{\mathbf{x}}(j+1,LL)$.

$j\backslash\mathbf{p}^i$	ϵ	P	O	L	LO
1	-∞	1.00	0.00	0.00	-∞
2	-∞	1.55	2.31	1.55	1.00
3	-∞	3.10	3.87	3.12	2.55
4	-∞	4.65	4.42	5.65	3.10
5	-∞	6.21	6.35	6.21	6.65
6	-∞	7.76	7.52	9.21	6.21
7	-∞	9.60	9.45	9.59	10.21
8	-∞	11.14	11.91	11.14	10.59

Table 12: The values of $\ln \alpha_{\mathbf{x}}(j,\mathbf{p}^i)$.

We also have $Z_{\mathbf{x}} = \alpha_{\mathbf{x}}(8,\epsilon) + \alpha_{\mathbf{x}}(8,P) + \alpha_{\mathbf{x}}(8,O) + \alpha_{\mathbf{x}}(8,L) + \alpha_{\mathbf{x}}(8,LO) = e^{12.696}$.

We now illustrate the computation of $\beta_{\mathbf{x}}$ with $\beta_{\mathbf{x}}(5,OL)$. The condition $(\mathbf{s}^k,y) : \mathbf{s}^k \leq^s_{\mathcal{S}} \mathbf{s}^i y$ with $\mathbf{s}^i = OL$ gives us the following 3 pairs as $(\mathbf{s}^k,y)$: $\{(LP,P),(LO,O),(LL,L)\}$. Thus,

$$\begin{aligned}\beta_{\mathbf{x}}(5,OL) &= \beta_{\mathbf{x}}(6,LP)\Psi_{\mathbf{x}}(5,5,OLP) + \beta_{\mathbf{x}}(6,LO)\Psi_{\mathbf{x}}(5,5,OLO) + \\ &\quad \beta_{\mathbf{x}}(6,LL)\Psi_{\mathbf{x}}(5,5,OLL) \\ &= \beta_{\mathbf{x}}(6,LP)e^0 + \beta_{\mathbf{x}}(6,LO)e + \beta_{\mathbf{x}}(6,LL)e^0.\end{aligned}$$

The values of the marginals $P(j,j,\mathbf{z}|\mathbf{x})$ are shown in Table 14. We illustrate the computation of $P(6,6,LOL|\mathbf{x})$ from the forward and backward variables. The condition

$j\backslash\mathbf{s}^i$	P	O	L	PP	PO	PL	OP	OO	OL	LP	LO	LL	LOP	LOO	LOL
1	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70	12.70
2	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14	11.14
3	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59	9.59
4	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04
5	6.21	6.21	6.66	6.21	6.21	6.66	6.21	6.21	6.66	6.21	6.21	6.66	6.21	6.21	6.66
6	4.65	4.65	4.65	4.65	4.65	4.65	4.65	4.65	4.65	4.65	5.34	4.65	4.65	4.65	4.65
7	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10	3.10
8	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55	1.55

Table 13: The values of $\ln \beta_{\mathbf{x}}(j, \mathbf{s}^i)$.

$(\mathbf{p}^i, y) : \mathbf{z} \leq^s \mathbf{p}^i y$ with $\mathbf{z} = LOL$ gives us the only pair (LO, L) as $(\mathbf{p}^i, y)$. Hence,

$$\begin{aligned} P(6, 6, LOL|\mathbf{x}) &= \frac{\alpha_{\mathbf{x}}(5, LO)\beta_{\mathbf{x}}(7, LOL)\Psi_{\mathbf{x}}(6, 6, LOL)}{Z_{\mathbf{x}}} \\ &= \frac{\alpha_{\mathbf{x}}(5, LO)\beta_{\mathbf{x}}(7, LOL)e^2}{Z_{\mathbf{x}}}. \end{aligned}$$

$j\backslash\mathbf{z}$	P	O	L	LOL
1	0.58	0.21	0.21	0.00
2	0.21	0.58	0.21	0.00
3	0.21	0.58	0.21	0.03
4	0.16	0.16	0.68	0.08
5	0.16	0.68	0.16	0.01
6	0.16	0.16	0.68	0.39
7	0.21	0.58	0.21	0.01
8	0.21	0.58	0.21	0.08

Table 14: The marginals $P(j, j, \mathbf{z}|\mathbf{x})$.

References

Aron Culotta, David Kulp, and Andrew McCallum. Gene prediction with conditional random fields. Technical Report UM-CS-2005-028, University of Massachusetts, Amherst, 2005.

Thomas G. Dietterich, Adam Ashenfelter, and Yaroslav Bulatov. Training conditional random fields via gradient tree boosting. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Richard Durbin. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical hidden Markov model: analysis and applications. *Machine Learning*, 32(1):41–62, 1998.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 427–434, 2005.

Josée S. Heemskerk. A probabilistic context-free grammar for disambiguation in morphological parsing. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*, pages 183–192, 1993.

Cecil Huang and Adnan Darwiche. Inference in belief networks: a procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

Sorin Istrail. Statistical mechanics, three-dimensionality and NP-completeness: I. Universality of intractability for the partition function of the Ising model across non-planar lattices. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 87–96, 2000.

Frederick Jelinek, John D. Lafferty, and Robert L. Mercer. Basic methods of probabilistic context free grammars. In *Speech Recognition and Understanding. Recent Advances, Trends, and Applications*. Springer, 1992.

Robert H. Kassel. *A Comparison of Approaches to On-line Handwritten Character Recognition*. PhD thesis, Massachusetts Institute of Technology, 1995.

Pushmeet Kohli, M. Pawan Kumar, and Philip H. S. Torr. P3 & beyond: solving energies with higher order cliques. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

Nikos Komodakis and Nikos Paragios. Beyond pairwise energies: efficient optimization for higher-order MRFs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2985–2992, 2009.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.

Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. Using conditional random fields for sentence boundary detection in speech. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 451–458, 2005.

Wei Lu and Hwee Tou Ng. Better punctuation prediction with dynamic conditional random fields. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 177–186, 2010.

Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the 7th Conference on Computational Natural Language Learning*, pages 188–191, 2003.

Kevin P. Murphy and Mark A. Paskin. Linear-time inference in hierarchical HMMs. In *Advances in Neural Information Processing Systems*, pages 833–840, 2002.

Viet Cuong Nguyen, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. Semi-Markov conditional random field with high-order features. In *ICML Workshop on Structured Sparsity: Learning and Inference*, 2011.

Eric W. Noreen. *Computer Intensive Methods for Testing Hypotheses: An Introduction.* Wiley, 1989.

Michael Paul. Overview of the IWSLT 2009 evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 3–27, 2009.

Xian Qian and Yang Liu. Sequence labeling with non-negative weighted higher order features. In *Proceedings of the 26th Conference on Artificial Intelligence*, 2012.

Xian Qian, Xiaoqian Jiang, Qi Zhang, Xuanjing Huang, and Lide Wu. Sparse higher order conditional random fields for improved sequence labeling. In *Proceedings of the 26th International Conference on Machine Learning*, pages 849–856, 2009.

Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Carsten Rother, Pushmeet Kohli, Wei Feng, and Jiaya Jia. Minimizing sparse higher order energy functions of discrete variables. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1382–1389, 2009.

Sunita Sarawagi and William W. Cohen. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, pages 1185–1192, 2004.

Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 134–141, 2003.

Daniel Tarlow, Inmar E. Givoni, and Richard S. Zemel. HOP-MAP: efficient message passing with high order potentials. In *International Conference on Artificial Intelligence and Statistics*, pages 812–819, 2010.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*, 2004.

Tran T. Truyen, Dinh Q. Phung, Hung H. Bui, and Svetha Venkatesh. Hierarchical semi-Markov conditional random fields for recursive sequential data. In *Advances in Neural Information Processing Systems*, pages 1657–1664, 2008.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. ACE 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 2006.

Nan Ye, Wee Sun Lee, Hai Leong Chieu, and Dan Wu. Conditional random fields with high-order features for sequence labeling. In *Advances in Neural Information Processing Systems*, pages 2196–2204, 2009.

Alexander Yeh. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 947–953, 2000.

Min Zhang, Jie Zhang, and Jian Su. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 288–295, 2006.

Journal of Machine Learning Research 15 (2014) 1011-1039 Submitted 9/13; Published 3/14

Ellipsoidal Rounding for Nonnegative Matrix Factorization Under Noisy Separability

Tomohiko Mizutani MIZUTANI@KANAGAWA-U.AC.JP
Department of Information Systems Creation
Kanagawa University
Yokohama, 221-8686, Japan

Editor: Nathan Srebro

Abstract

We present a numerical algorithm for nonnegative matrix factorization (NMF) problems under noisy separability. An NMF problem under separability can be stated as one of finding all vertices of the convex hull of data points. The research interest of this paper is to find the vectors as close to the vertices as possible in a situation in which noise is added to the data points. Our algorithm is designed to capture the shape of the convex hull of data points by using its enclosing ellipsoid. We show that the algorithm has correctness and robustness properties from theoretical and practical perspectives; correctness here means that if the data points do not contain any noise, the algorithm can find the vertices of their convex hull; robustness means that if the data points contain noise, the algorithm can find the near-vertices. Finally, we apply the algorithm to document clustering, and report the experimental results.

Keywords: nonnegative matrix factorization, separability, robustness to noise, enclosing ellipsoid, document clustering

1. Introduction

This paper presents a numerical algorithm for nonnegative matrix factorization (NMF) problems under noisy separability. The problem can be regarded as a special case of an NMF problem. Let $\mathbb{R}_+^{d\times m}$ be the set of d-by-m nonnegative matrices, and $\mathbb{N}$ be the set of nonnegative integer numbers. A nonnegative matrix is a real matrix whose elements are all nonnegative. For a given $\boldsymbol{A} \in \mathbb{R}_+^{d\times m}$ and $r \in \mathbb{N}$, the nonnegative matrix factorization (NMF) problem is to find $\boldsymbol{F} \in \mathbb{R}_+^{d\times r}$ and $\boldsymbol{W} \in \mathbb{R}_+^{r\times m}$ such that the product $\boldsymbol{FW}$ is as close to $\boldsymbol{A}$ as possible. The nonnegative matrices $\boldsymbol{F}$ and $\boldsymbol{W}$ give a factorization of $\boldsymbol{A}$ of the form,

$$\boldsymbol{A} = \boldsymbol{FW} + \boldsymbol{N},$$

where $\boldsymbol{N}$ is a d-by-m matrix. This factorization is referred to as the NMF of $\boldsymbol{A}$.

Recent studies have shown that NMFs are useful for tackling various problems such as facial image analysis (Lee and Seung, 1999), topic modeling (Arora et al., 2012b, 2013; Ding et al., 2013), document clustering (Xu et al., 2003; Shahnaz et al., 2006), hyperspectral unmixing (Nascimento and Dias, 2005; Miao and Qi, 2007; Gillis and Vavasis, 2014), and blind source separation (Cichocki et al., 2009). Many algorithms have been developed in the context of solving such practical applications. However, there are some drawbacks in

the use of NMFs for such applications. One of them is in the hardness of solving an NMF problem. In fact, the problem has been shown to be NP-hard by Vavasis (2009).

As a remedy for the hardness of the problem, Arora et al. (2012a) proposed to exploit the notion of separability, which was originally introduced by Donoho and Stodden (2003) for the uniqueness of NMF. An NMF problem under separability becomes a tractable one. *Separability* assumes that $\boldsymbol{A} \in \mathbb{R}_+^{d\times m}$ can be represented as

$$\boldsymbol{A} = \boldsymbol{F}\boldsymbol{W} \text{ for } \boldsymbol{F} \in \mathbb{R}_+^{d\times r} \text{ and } \boldsymbol{W} = (\boldsymbol{I}, \boldsymbol{K})\boldsymbol{\Pi} \in \mathbb{R}_+^{r\times m}, \tag{1}$$

where $\boldsymbol{I}$ is an r-by-r identity matrix, $\boldsymbol{K}$ is an r-by-$(m-r)$ nonnegative matrix, and $\boldsymbol{\Pi}$ is an m-by-m permutation matrix. This means that each column of $\boldsymbol{F}$ corresponds to that of $\boldsymbol{A}$ up to a scaling factor. A matrix $\boldsymbol{A}$ is said to be a *separable matrix* if it can be represented in the form (1). In this paper, we call $\boldsymbol{F}$ the *basis matrix* of a separable matrix, and $\boldsymbol{W}$, as well as its submatrix $\boldsymbol{K}$, the *weight matrix*. *Noisy separability* assumes that a separable matrix $\boldsymbol{A}$ contains a noise matrix $\boldsymbol{N}$ such that $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{N}$, where $\boldsymbol{N}$ is a d-by-m matrix. Arora et al. showed that there exists an algorithm for finding the near-basis matrix of a noisy separable one if the noise is small in magnitude. Although a separability assumption restricts the fields of application for NMFs, it is known to be reasonable at least, in the contexts of document clustering (Kumar et al., 2013), topic modeling (Arora et al., 2012a,b, 2013), and hyperspectral unmixing (Gillis and Vavasis, 2014). In particular, this assumption is widely used as a pure-pixel assumption in hyperspectral unmixing (see, for instance, Nascimento and Dias, 2005; Miao and Qi, 2007; Gillis and Vavasis, 2014).

An NMF problem under noisy separability is to seek for the basis matrix of a noisy separable one. The problem is formally described as follows:

Problem 1 *Let a data matrix $\boldsymbol{M}$ be a noisy separable matrix of size d-by-m. Find an index set $\mathcal{I}$ with cardinality r on $\{1,\dots,m\}$ such that $\boldsymbol{M}(\mathcal{I})$ is as close to the basis matrix $\boldsymbol{F}$ as possible.*

Here, $\boldsymbol{M}(\mathcal{I})$ denotes a submatrix of $\boldsymbol{M}$ that consists of every column vector with an index in $\mathcal{I}$. We call the column vector of $\boldsymbol{M}$ a *data point* and that of the basis matrix $\boldsymbol{F}$ a *basis vector*. An ideal algorithm for the problem should have correctness and robustness properties; correctness here means that, if the data matrix $\boldsymbol{M}$ is just a separable one, the algorithm can find the basis matrix; robustness means that, if the data matrix $\boldsymbol{M}$ is a noisy separable one, the algorithm can find the near-basis matrix. A formal description of the properties is given in Section 2.1

We present a novel algorithm for Problem 1. The main contribution of this paper is to show that the algorithm has correctness and robustness properties from theoretical and practical perspectives. It is designed on the basis of the geometry of a separable matrix. Under reasonable assumptions, the convex hull of the column vectors of a separable matrix forms a simplex, and in particular, the basis vectors correspond to the vertices. Therefore, if all vertices of a simplex can be found, we can obtain the basis matrix of the separable matrix. Our algorithm uses the fact that the vertices of simplex can be found by an ellipsoid. That is, if we draw the minimum-volume enclosing ellipsoid (MVEE) for a simplex, the ellipsoid only touches its vertices. More precisely, we give plus and minus signs to the vertices of a simplex, and take the convex hull; it becomes a crosspolytope having the simplex as one of

the facets. Then, the MVEE for the crosspolytope only touches the vertices of the simplex with plus and minus signs.

Consider Problem 1 without noise. In this case, the data matrix is just a separable one. Our algorithm computes the MVEE for the data points and outputs the points on the boundary of the ellipsoid. Then, the obtained points correspond to the basis vectors for a separable matrix. We show in Theorem 5 that the correctness property holds. Moreover, the algorithm works well even when the problem contains noise. We show in Theorem 9 that, if the noise is lower than a certain level, the algorithm correctly identifies the near-basis vectors for a noisy separable matrix, and hence, the robustness property holds. The existing algorithms (Arora et al., 2012a; Bittorf et al., 2012; Gillis, 2013; Gillis and Luce, 2013; Gillis and Vavasis, 2014; Kumar et al., 2013) are formally shown to have these correctness and robustness properties. In Section 2.4, our correctness and robustness properties are compared with those of the existing algorithms.

It is possible that noise will exceed the level that Theorem 9 guarantees. In such a situation, the MVEE for the data points may touch many points. Hence, r points need to be selected from the points on the boundary of the ellipsoid. We make the selection by using existing algorithms such as SPA (Gillis and Vavasis, 2014) and XRAY (Kumar et al., 2013). Our algorithm thus works as a preprocessor which filters out basis vector candidates from the data points and enhance the performance of existing algorithms.

We demonstrated the robustness of the algorithms to noise through experiments with synthetic data sets. In particular, we experimentally compared our algorithm with SPA and XRAY. We synthetically generated data sets with various noise levels, and measured the robustness of an algorithm by its recovery rate. The experimental results indicated that our algorithm can improve the recovery rates of SPA and XRAY.

Finally, we applied our algorithm to document clustering. Separability for a document-term matrix means that each topic has an anchor word. An anchor word is a word which is contained in one topic but not contained in the other topics. If an anchor word is found, it suggests the existence of its associated topic. We conducted experiments with document corpora and compared the clustering performances of our algorithm and SPA. The experimental results indicated that our algorithm would usually outperform SPA and can extract more recognizable topics.

The rest of this paper is organized as follows. Section 2 gives an outline of our algorithm and reviews related work. Then, the correctness and robustness properties of our algorithm are given, and a comparison with existing algorithms is described. Section 3 reviews the formulation and algorithm of computing the MVEE for a set of points. Sections 4 and 5 are the main part of this paper. We show the correctness and robustness properties of our algorithm in Section 4, and discuss its practical implementation in Section 5. Section 6 reports the numerical experiments for the robustness of algorithms and document clustering. Section 7 gives concluding remarks.

1.1 Notation and Symbols

We use $\mathbb{R}^{d\times m}$ to denote a set of real matrices of size d-by-m, and $\mathbb{R}_+^{d\times m}$ to denote a set of nonnegative matrices of d-by-m. Let $\boldsymbol{A} \in \mathbb{R}^{d\times m}$. The symbols $\boldsymbol{A}^\top$ and $\mathrm{rank}(\boldsymbol{A})$ respectively denote the transposition and the rank. The symbols $||\boldsymbol{A}||_p$ and $||\boldsymbol{A}||_F$ are the matrix p-norm

and the Frobenius norm. The symbol $\sigma_i(\boldsymbol{A})$ is the ith largest singular value. Let $\boldsymbol{a}_i$ be the ith column vector of $\boldsymbol{A}$, and $\mathcal{I}$ be a subset of $\{1,\ldots,m\}$. The symbol $\boldsymbol{A}(\mathcal{I})$ denotes a d-by-$|\mathcal{I}|$ submatrix of $\boldsymbol{A}$ such that $(\boldsymbol{a}_i : i \in \mathcal{I})$. The convex hull of all the column vectors of $\boldsymbol{A}$ is denoted by $\mathrm{conv}(\boldsymbol{A})$, and referred to as the convex hull of $\boldsymbol{A}$ for short. We denote an identity matrix and a vector of all ones by $\boldsymbol{I}$ and $\boldsymbol{e}$, respectively.

We use $\mathbb{S}^d$ to denote a set of real symmetric matrices of size d. Let $\boldsymbol{A} \in \mathbb{S}^d$. If the matrix is positive definite, we represent it as $\boldsymbol{A} \succ \boldsymbol{0}$. Let $\boldsymbol{A}_1 \in \mathbb{S}^d$ and $\boldsymbol{A}_2 \in \mathbb{S}^d$. We denote by $\langle \boldsymbol{A}_1, \boldsymbol{A}_2 \rangle$ the Frobenius inner product of the two matrices which is given as the trace of matrix $\boldsymbol{A}_1\boldsymbol{A}_2$.

We use a MATLAB-like notation. Let $\boldsymbol{A}_1 \in \mathbb{R}^{d\times m_1}$ and $\boldsymbol{A}_2 \in \mathbb{R}^{d\times m_2}$. We denote by $(\boldsymbol{A}_1, \boldsymbol{A}_2)$ the horizontal concatenation of the two matrices, which is a d-by-(m_1+m_2) matrix. Let $\boldsymbol{A}_1 \in \mathbb{R}^{d_1\times m}$ and $\boldsymbol{A}_2 \in \mathbb{R}^{d_2\times m}$. We denote by $(\boldsymbol{A}_1; \boldsymbol{A}_2)$ the vertical concatenation of the two matrices, and it is a matrix of the form,

$$\begin{pmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \end{pmatrix} \in \mathbb{R}^{(d_1+d_2)\times m}.$$

Let $\boldsymbol{A}$ be a d-by-m rectangular diagonal matrix having diagonal elements $a_1,\ldots,a_t$ where $t = \min\{d, m\}$. We use $\mathrm{diag}(a_1,\ldots,a_t)$ to denote the matrix.

2. Outline of Proposed Algorithm and Comparison with Existing Algorithms

Here, we formally describe the properties mentioned in Section 1 that an algorithm is expected to have, and also describe the assumptions we place on Problem 1. Next, we give a geometric interpretation of a separable matrix under these assumptions, and then, outline the proposed algorithm. After reviewing the related work, we describe the correctness and robustness properties of our algorithm and compare with those of the existing algorithms.

2.1 Preliminaries

Consider Problem 1 whose data matrix $\boldsymbol{M}$ is a noisy separable one of the form $\boldsymbol{A} + \boldsymbol{N}$. Here, $\boldsymbol{A}$ is a separable matrix of (1) and $\boldsymbol{N}$ is a noise matrix. We can rewrite it as

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{A} + \boldsymbol{N} \\ &= \boldsymbol{F}(\boldsymbol{I}, \boldsymbol{K})\boldsymbol{\Pi} + \boldsymbol{N} \\ &= (\boldsymbol{F} + \boldsymbol{N}^{(1)}, \boldsymbol{F}\boldsymbol{K} + \boldsymbol{N}^{(2)})\boldsymbol{\Pi} \end{aligned} \tag{2}$$

where $\boldsymbol{N}^{(1)}$ and $\boldsymbol{N}^{(2)}$ are d-by-r and d-by-ℓ submatrices of $\boldsymbol{N}$ such that $\boldsymbol{N}\boldsymbol{\Pi}^{-1} = (\boldsymbol{N}^{(1)}, \boldsymbol{N}^{(2)})$. Hereinafter, we use the notation ℓ to denote $m - r$. The goal of Problem 1 is to identify an index set $\mathcal{I}$ such that $\boldsymbol{M}(\mathcal{I}) = \boldsymbol{F} + \boldsymbol{N}^{(1)}$.

As mentioned in Section 1, it is ideal that an algorithm for Problem 1 has correctness and robustness properties. These properties are formally described as follows:

- **Correctness.** If the data matrix $\boldsymbol{M}$ does not contain a noise matrix $\boldsymbol{N}$ and is just a separable matrix, the algorithm returns an index set $\mathcal{I}$ such that $\boldsymbol{M}(\mathcal{I}) = \boldsymbol{F}$.

- **Robustness.** If the data matrix $\boldsymbol{M}$ contains a noise matrix $\boldsymbol{N}$ and is a noisy separable matrix such that $||\boldsymbol{N}||_p < \epsilon$, the algorithm returns an index set $\mathcal{I}$ such that $||\boldsymbol{M}(\mathcal{I}) - \boldsymbol{F}||_p < \tau\epsilon$ for some constant real number τ.

In particular, the robustness property has $\tau = 1$, if an algorithm can identify an index set $\mathcal{I}$ such that $\boldsymbol{M}(\mathcal{I}) = \boldsymbol{F}+\boldsymbol{N}^{(1)}$ where $\boldsymbol{F}$ and $\boldsymbol{N}^{(1)}$ are of (2) since $||\boldsymbol{M}(\mathcal{I})-\boldsymbol{F}||_p = ||\boldsymbol{N}^{(1)}||_p < \epsilon$.

In the design of the algorithm, some assumptions are usually placed on a separable matrix. Our algorithm uses Assumption 1.

Assumption 1 *A separable matrix $\boldsymbol{A}$ of (1) consists of an basis matrix $\boldsymbol{F}$ and a weight matrix $\boldsymbol{W}$ satisfying the following conditions.*

1-a) Every column vector of weight matrix $\boldsymbol{W}$ has unit 1-norm.

1-b) The basis matrix $\boldsymbol{F}$ has full column rank.

Assumption 1-a can be invoked without loss of generality. If the ith column of $\boldsymbol{W}$ is zero, so is the ith column of $\boldsymbol{A}$. Therefore, we can construct a smaller separable matrix having $\boldsymbol{W}$ with no zero column. Also, since we have

$$\boldsymbol{A} = \boldsymbol{F}\boldsymbol{W} \Leftrightarrow \boldsymbol{A}\boldsymbol{D} = \boldsymbol{F}\boldsymbol{W}\boldsymbol{D},$$

every column of $\boldsymbol{W}$ can have unit 1-norm. Here, $\boldsymbol{D}$ denotes a diagonal matrix having the (i,i)th diagonal element $d_{ii} = 1/||\boldsymbol{w}_i||_1$.

The same assumption is used by the algorithm in Gillis and Vavasis (2014). We may get the feeling that 1-b is strong. The algorithms (Arora et al., 2012a; Bittorf et al., 2012; Gillis, 2013; Gillis and Luce, 2013; Kumar et al., 2013) instead assume *simpliciality*, wherein no column vector of $\boldsymbol{F}$ can be represented as a convex hull of the remaining vectors of $\boldsymbol{F}$. Although 1-b is a stronger assumption, it still seems reasonable for Problem 1 from the standpoint of practical application. This is because, in such cases, it is less common for the column vectors of the basis matrix $\boldsymbol{F}$ to be linearly dependent.

2.2 Outline of Proposed Algorithm

Let us take a look at Problem 1 from a geometric point of view. For simplicity, consider the noiseless case first. Here, a data matrix is just a separable matrix $\boldsymbol{A}$. Separability implies that $\boldsymbol{A}$ has a factorization of the form (1). Under Assumption 1, $\mathrm{conv}(\boldsymbol{A})$ becomes an $(r-1)$-dimensional simplex in $\mathbb{R}^d$. The left part of Figure 1 visualizes a separable data matrix. The white points are data points, and the black ones are basis vectors. The key observation is that the basis vectors $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_r$ of $\boldsymbol{A}$ correspond to the vertices of $\mathrm{conv}(\boldsymbol{A})$. This is due to separability. Therefore, if all vertices of $\mathrm{conv}(\boldsymbol{A})$ can be found, we can obtain the basis matrix $\boldsymbol{F}$ of $\boldsymbol{A}$. This is not hard task, and we can design an efficient algorithm for doing it. But, if noise is added to a separable matrix, the task becomes hard. Let us suppose that the data matrix of Problem 1 is a noisy separable matrix $\widetilde{\boldsymbol{A}}$ of the form $\boldsymbol{A}+\boldsymbol{N}$. The vertices of $\mathrm{conv}(\widetilde{\boldsymbol{A}})$ do not necessarily match the basis vectors $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_r$ of $\boldsymbol{A}$. The right part of Figure 1 visualizes a noisy separable data matrix. This is the main reason why it is hard to identify the basis matrix from noisy separable one.

Our algorithm is designed on the basis of Proposition 3 in Section 4.1; it states that all vertices of a simplex can be found by using an ellipsoid. We here describe the proposition

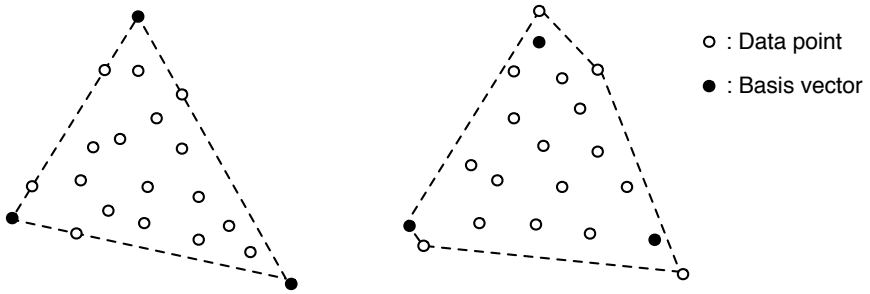


Figure 1: Convex hull of a separable data matrix with $r = 3$ under Assumption 1. (Left) Noiseless case. (Right) Noisy case.

from a geometric point of view. Consider an $(r-1)$-dimensional simplex Δ in $\mathbb{R}^r$. Let $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_r \in \mathbb{R}^r$ be the vertices of Δ, and $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_\ell \in \mathbb{R}^r$ be the points in Δ. We draw the MVEE centered at the origin for a set $\mathcal{S} = \{\pm\boldsymbol{g}_1, \ldots, \pm\boldsymbol{g}_r, \pm\boldsymbol{b}_1, \ldots, \pm\boldsymbol{b}_\ell\}$. Then, the proposition says that the ellipsoid only touches the points $\pm\boldsymbol{g}_1, \ldots, \pm\boldsymbol{g}_r$ among all the points in $\mathcal{S}$. Therefore, the vertices of Δ can be found by checking whether the points in $\mathcal{S}$ lie on the boundary of ellipsoid. We should mention that the convex hull of the points in $\mathcal{S}$ becomes a full-dimensional crosspolytope in $\mathbb{R}^r$. Figure 2 illustrates the MVEE for a crosspolytope in $\mathbb{R}^3$.

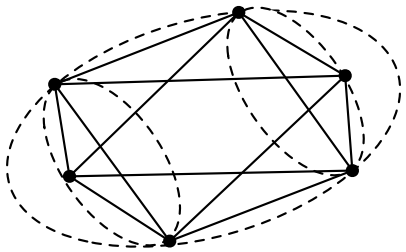


Figure 2: Minimum-volume enclosing ellipsoid for a full-dimensional crosspolytope in $\mathbb{R}^3$.

Under Assumption 1, the convex hull of a separable matrix $\boldsymbol{A}$ becomes an $(r-1)$-dimensional simplex in $\mathbb{R}^d$. Therefore, we rotate and embed the simplex in $\mathbb{R}^r$ by using an orthogonal transformation. Such a transformation can be obtained by singular value decomposition (SVD) of $\boldsymbol{A}$.

Now let us outline our algorithm for Problem 1. In this description, we assume for simplicity that the data matrix is a separable one $\boldsymbol{A} \in \mathbb{R}_+^{d\times m}$. First, the algorithm constructs an orthogonal transformation through the SVD of $\boldsymbol{A}$. By applying the transformation, it transforms $\boldsymbol{A}$ into a matrix $\boldsymbol{P} \in \mathbb{R}^{r\times m}$ such that the $\mathrm{conv}(\boldsymbol{P})$ is an $(r-1)$-dimensional sim-

plex in $\mathbb{R}^r$. Next, it draws the MVEE centered at the origin for a set $\mathcal{S} = \{\pm \boldsymbol{p}_1, \ldots, \pm \boldsymbol{p}_m\}$, where $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m$ are the column vectors of $\boldsymbol{P}$, and outputs r points lying on the ellipsoid.

We call the algorithm *ellipsoidal rounding*, abbreviated as ER. The main computational costs of ER are in computing the SVD of $\boldsymbol{A}$ and the MVEE for $\mathcal{S}$. The MVEE computation can be formulated as a tractable convex optimization problem with m variables. A polynomial-time algorithm exists, and it is also known that a hybrid of the interior-point algorithm and cutting plane algorithm works efficiently in practice.

In later sections, we will see that ER algorithm works well even if noise is added. In particular, we show that ER correctly identifies the near-basis vectors of a noisy separable matrix if the noise is smaller than some level. We consider a situation in which the noise exceeds that level. In such a situation, the shape of crosspolytope formed by the data points is considerably perturbed by the noise, and it is possible that the MVEE touches many points. We thus need to select r points from the points on the boundary of the ellipsoid. In this paper, we perform existing algorithms such as SPA (Gillis and Vavasis, 2014) and XRAY (Kumar et al., 2013) to make the selection. Hence, ER works as a preprocessor which filters out basis vector candidates from data points and enhances the performance of existing algorithms.

2.3 Related Work

First, we will review the algorithms for NMF of general nonnegative matrices. There are an enormous number of studies. A commonly used approach is to formulate it as a nonconvex optimization problem and compute the local solution. Let $\boldsymbol{A}$ be a d-by-m nonnegative matrix, and consider an optimization problem with matrix variables $\boldsymbol{F} \in \mathbb{R}^{d \times r}$ and $\boldsymbol{W} \in \mathbb{R}^{r \times m}$,

$$\text{minimize } ||\boldsymbol{F}\boldsymbol{W} - \boldsymbol{A}||_F^2 \text{ subject to } \boldsymbol{F} \geq \boldsymbol{0} \text{ and } \boldsymbol{W} \geq \boldsymbol{0}.$$

This is an intractable nonconvex optimization problem, and in fact, it was shown to be NP-hard by Vavasis (2009). Therefore, the research target is in how to compute the local solution efficiently. It is popular to use the block coordinate descent (BCD) algorithm for this purpose. The algorithm solves the problem by alternately fixing the variables $\boldsymbol{F}$ and $\boldsymbol{W}$. The problem obtained by fixing either of $\boldsymbol{F}$ and $\boldsymbol{W}$ becomes a convex optimization problem. The existing studies propose to use, for instance, the projected gradient algorithm (Lin, 2007) and its variant (Lee and Seung, 2001), active set algorithm (Kim and Park, 2008, 2011), and projected quasi-Newton algorithm (Gong and Zhang, 2012). It is reported that the BCD algorithm shows good performance on average in computing NMFs. However, its performance depends on how we choose the initial point for starting the algorithm. We refer the reader to Kim et al. (2014) for a survey on the algorithms for NMF.

Next, we will survey the algorithms that work on noisy separable matrices. Four types of algorithm can be found:

- **AGKM (Arora et al., 2012a).** The algorithm constructs r sets of data points such that all of the basis vectors are contained in the union of the sets and each set has one basis vector. The construction entails solving m linear programming (LP) problems with $m - 1$ variables. Then, it chooses one element from each set, and outputs them.

- **Hottopixx (Bittorf et al., 2012; Gillis, 2013; Gillis and Luce, 2013).** Let $\boldsymbol{A}$ be a separable matrix of the form $\boldsymbol{F}(\boldsymbol{I}, \boldsymbol{K})\boldsymbol{\Pi}$. Consider a matrix $\boldsymbol{C}$ such that

$$\boldsymbol{C} = \boldsymbol{\Pi}^{-1} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{K} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \boldsymbol{\Pi} \in \mathbb{R}^{m \times m}.$$

  It satisfies $\boldsymbol{A} = \boldsymbol{AC}$, and also, if the diagonal element is one, the position of its diagonal element indicates the index of basis vector in $\boldsymbol{A}$. The algorithm models $\boldsymbol{C}$ as the variable of an LP problem. It entails solving a single LP problem with m^2 variables.

- **SPA (Gillis and Vavasis, 2014).** Let $\boldsymbol{A}$ be a separable matrix of size d-by-m, and $\mathcal{S}$ be the set of the column vectors of $\boldsymbol{A}$. The algorithm is based on the following observation. Under Assumption 1, the maximum of a convex function over the elements in $\mathcal{S}$ is attained at the vertex of $\mathrm{conv}(\boldsymbol{A})$. The algorithm finds one element $\boldsymbol{a}$ in $\mathcal{S}$ that maximizes a convex function, and then, projects all elements in $\mathcal{S}$ into the orthogonal space to $\boldsymbol{a}$. This procedure is repeated until r elements are found.

- **XRAY (Kumar et al., 2013).** The algorithm has a similar spirit as SPA, but it uses a linear function instead of a convex one. Let $\boldsymbol{A}$ be a separable matrix of size d-by-m and $\mathcal{S}$ be the set of the column vectors of $\boldsymbol{A}$. Let $\mathcal{I}_k$ be the index set obtained after the kth iteration. This is a subset of $\{1, \ldots, m\}$ with cardinality k. In the $(k+1)$th iteration, it computes a residual matrix $\boldsymbol{R} = \boldsymbol{A}(\mathcal{I}_k)\boldsymbol{X}^* - \boldsymbol{A}$, where

$$\boldsymbol{X}^* = \arg\min_{\boldsymbol{X} \geq \boldsymbol{0}} ||\boldsymbol{A}(\mathcal{I}_k)\boldsymbol{X} - \boldsymbol{A}||_2^2,$$

  and picks up one of the column vectors $\boldsymbol{r}_i$ of $\boldsymbol{R}$. Then, it finds one element from $\mathcal{S}$ which maximizes a linear function having $\boldsymbol{r}_i$ as the normal vector. Finally, $\mathcal{I}_k$ is updated by adding the index of the obtained element. This procedure is repeated until r indices are found. The performance of XRAY depends on how we select the column vector of the residual matrix $\boldsymbol{R}$ for making the linear function. Several ways of selection, called "rand", "max", "dist" and "greedy", have been proposed by the authors.

The next section describes the properties of these algorithms.

2.4 Comparison with Existing Algorithm

We compare the correctness and robustness properties of ER with those of AGKM, Hottopixx, SPA, and XRAY. ER is shown to have the two properties in Theorems 5 and 9. In particular, our robustness property in Theorem 9 states that ER correctly identifies the near-basis matrix of a noisy separable one $\widetilde{\boldsymbol{A}}$, and a robustness property with $\tau = 1$ holds if we set

$$\epsilon = \frac{\sigma(1-\mu)}{4} \tag{3}$$

and $p = 2$ under Assumption 1. Here, σ is the minimum singular value of the basis matrix $\boldsymbol{F}$ of a separable one $\boldsymbol{A}$ in the $\widetilde{\boldsymbol{A}}$, that is, $\sigma = \sigma_r(\boldsymbol{F})$, and μ is $\mu(\boldsymbol{K})$:

$$\mu(\boldsymbol{K}) = \max_{i=1,\ldots,\ell} ||\boldsymbol{k}_i||_2$$

for a weight matrix $\boldsymbol{K}$ of $\boldsymbol{A}$. Under Assumption 1-a, we have $\mu \leq 1$, and in particular, equality holds if and only if $\boldsymbol{k}_i$ has only one nonzero element.

All four of the existing algorithms have been shown to have a correctness property, whereas every one except XRAY has a robustness property. Hottopixx is the most similar to ER. Bittorf et al. (2012) showed that it has the correctness and robustness with $\tau = 1$ properties if one sets

$$\epsilon = \frac{\alpha \min\{d_0, \alpha\}}{9(r+1)} \tag{4}$$

and $p = 1$ under simpliciality and other assumptions. Here, α and d_0 are as follows. α is the minimum value of $\delta_{\boldsymbol{F}}(j)$ for $j = 1, \ldots, r$, where $\delta_{\boldsymbol{F}}(j)$ denotes an ℓ_1-distance between the jth column vector $\boldsymbol{f}_j$ of $\boldsymbol{F}$ and the convex hull of the remaining column vectors in $\boldsymbol{F}$. d_0 is the minimum value of $||\boldsymbol{a}_i - \boldsymbol{f}_j||_1$ for every i such that $\boldsymbol{a}_i$ is not a basis vector, and every $j = 1, \ldots, r$. The robustness of Hottopixx is further analyzed (Gillis, 2013; Gillis and Luce, 2013).

It can be interpreted that the ϵ of ER (3) is given by the multiplication of two parameters representing flatness and closeness of a given data matrix since σ measures the flatness of the convex hull of data points, and $1 - \mu$ measures the closeness between basis vectors and data points. Intuitively, we may say that an algorithm becomes sensitive to noise when a data matrix has the following features; one is that the convex hull of data points is close to a flat shape, and another is that there are data points close to basis vectors. The ϵ of (3) well matches the intuition. We see a similar structure in the ϵ of Hottopixx (4) since α and d_0 respectively measure the flatness and closeness of a given data.

Compared with Hottopixx, the ϵ of ER (3) does not contain $1/r$, and hence, it does not decrease as r increases. However, Assumption 1-b of ER is stronger than the simpliciality of Hottopixx. In a practical implementation, ER can handle a large matrix, while Hottopixx may have limitations on the size of the matrix it can handle. Hottopixx entails solving an LP problem with m^2 variables. In the NMFs arising in applications, m tends to be a large number. Although an LP is tractable, it becomes harder to solve as the size increases. Through experiments, we assessed the performance of Hottopixx with the CPLEX LP solver. The experiments showed that the algorithm had out of memory issues when m exceeded 2,000 with $d = 100$. Bittorf et al. (2012) proposed a parallel implementation to resolve these computational issues.

AGKM and SPA were shown to have a robustness property with $\tau \geq 1$ for some ϵ in Arora et al. (2012a) and Gillis and Vavasis (2014), respectively. In practical implementations, SPA and XRAY are scalable to the problem size and experimentally show good robustness. Section 6 reports a numerical comparison of ER with SPA and XRAY.

3. Review of Formulation and Algorithm for MVEE Computation

We review the formulation for computing the MVEE for a set of points, and survey the existing algorithms for the computation.

First of all, let us recall the terminology related to an ellipsoid. An ellipsoid in $\mathbb{R}^d$ is defined as a set $\mathcal{E}(\boldsymbol{L}, \boldsymbol{z}) = \{\boldsymbol{x} \in \mathbb{R}^d : (\boldsymbol{x} - \boldsymbol{z})^\top \boldsymbol{L}(\boldsymbol{x} - \boldsymbol{z}) \leq 1\}$ for a positive definite matrix $\boldsymbol{L}$ of size d and a vector $\boldsymbol{z} \in \mathbb{R}^d$. Here, $\boldsymbol{L}$ determines the shape of the ellipsoid and $\boldsymbol{z}$ is the center. Let $\boldsymbol{x}$ be a point in an ellipsoid $\mathcal{E}(\boldsymbol{L}, \boldsymbol{z})$. If the point $\boldsymbol{x}$ satisfies the equality

$(\boldsymbol{x}-\boldsymbol{z})^\top \boldsymbol{L}(\boldsymbol{x}-\boldsymbol{z})=1$, we call it an *active point* of the ellipsoid. In other words, an active point is one lying on the boundary of the ellipsoid.

The volume of the ellipsoid is given as $c(d)/\sqrt{\det \boldsymbol{L}}$, where $c(d)$ represents the volume of a unit ball in $\mathbb{R}^d$ and it is a real number depending on the dimension d. ER algorithm considers d-dimensional ellipsoids containing a set $\mathcal{S}$ of points in $\mathbb{R}^d$, and in particular, finds the minimum volume ellipsoid centered at the origin. In this paper, such an ellipsoid is referred to as an origin-centered MVEE for short.

Now, we are ready to describe a formulation for computing the origin-centered MVEE for a set of points. For m points $\boldsymbol{p}_1,\ldots,\boldsymbol{p}_m\in\mathbb{R}^d$, let $\mathcal{S}=\{\pm\boldsymbol{p}_1,\ldots,\pm\boldsymbol{p}_m\}$. The computation of the origin-centered MVEE for $\mathcal{S}$ is formulated as

$$\begin{array}{rlll}\mathbb{Q}(\mathcal{S}): & \text{minimize} & -\log\det \boldsymbol{L}, & \\ & \text{subject to} & \langle \boldsymbol{p}_i\boldsymbol{p}_i^\top, \boldsymbol{L}\rangle \le 1, & i=1,\ldots,m,\\ & & \boldsymbol{L}\succ \boldsymbol{0}, & \end{array}$$

where the matrix $\boldsymbol{L}$ of size d is the decision variable. The optimal solution $\boldsymbol{L}^*$ of $\mathbb{Q}$ gives the origin-centered MVEE for $\mathcal{S}$ as $\mathcal{E}(\boldsymbol{L}^*)=\{\boldsymbol{x}:\boldsymbol{x}^\top\boldsymbol{L}^*\boldsymbol{x}\le 1\}$. We here introduce some terminology. An active point of $\mathcal{E}(\boldsymbol{L}^*)$ is a vector $\boldsymbol{p}_i\in\mathbb{R}^d$ satisfying $\boldsymbol{p}_i^\top\boldsymbol{L}^*\boldsymbol{p}_i=1$. We call $\boldsymbol{p}_i$ an *active point of* $\mathbb{Q}(\mathcal{S})$, and the index i of $\boldsymbol{p}_i$ an *active index of* $\mathbb{Q}(\mathcal{S})$. The ellipsoid $\mathcal{E}(\boldsymbol{L}^*)$ is centrally symmetric, and if a vector $\boldsymbol{p}_i$ is an active point, so is $-\boldsymbol{p}_i$. The dual of $\mathbb{Q}$ reads

$$\begin{array}{rlll}\mathbb{Q}_*(\mathcal{S}): & \text{maximize} & \log\det \Omega(\boldsymbol{u}), & \\ & \text{subject to} & \boldsymbol{e}^\top\boldsymbol{u}=1, & \\ & & \boldsymbol{u}\ge \boldsymbol{0}, & \end{array}$$

where the vector $\boldsymbol{u}$ is the decision variable. Here, $\Omega:\mathbb{R}^m\to\mathbb{S}^d$ is a linear function given as $\Omega(\boldsymbol{u})=\sum_{i=1}^m \boldsymbol{p}_i\boldsymbol{p}_i^\top u_i$; equivalently, $\Omega(\boldsymbol{u})=\boldsymbol{P}\mathrm{diag}(\boldsymbol{u})\boldsymbol{P}^\top$ for $\boldsymbol{P}=(\boldsymbol{p}_1,\ldots,\boldsymbol{p}_m)\in\mathbb{R}^{d\times m}$. It follows from the Karush-Kuhn-Tucker (KKT) conditions for these problems that the optimal solution $\boldsymbol{L}^*$ of $\mathbb{Q}$ is represented by $\frac{1}{d}\Omega(\boldsymbol{u}^*)^{-1}$ for the optimal solution $\boldsymbol{u}^*$ of $\mathbb{Q}_*$. We make the following assumption to ensure the existence of an optimal solution of $\mathbb{Q}$.

Assumption 2 *rank*$(\boldsymbol{P})=d$ *for* $\boldsymbol{P}=(\boldsymbol{p}_1,\ldots,\boldsymbol{p}_m)\in\mathbb{R}^{d\times m}$.

Later, the KKT conditions will play an important role in our discussion of the active points of $\mathbb{Q}$. Here though, we will describe the conditions: $\boldsymbol{L}^*\in\mathbb{S}^d$ is an optimal solution for $\mathbb{Q}$ and $\boldsymbol{z}^*\in\mathbb{R}^m$ is the associated Lagrange multiplier vector if and only if there exist $\boldsymbol{L}^*\in\mathbb{S}^d$ and $\boldsymbol{z}^*\in\mathbb{R}^m$ such that

$$-(\boldsymbol{L}^*)^{-1}+\Omega(\boldsymbol{z}^*)=\boldsymbol{0}, \tag{5}$$
$$z_i^*(\langle \boldsymbol{p}_i\boldsymbol{p}_i^\top,\boldsymbol{L}^*\rangle-1)=0,\quad i=1,\ldots,m, \tag{6}$$
$$\langle \boldsymbol{p}_i\boldsymbol{p}_i^\top,\boldsymbol{L}^*\rangle\le 1,\quad i=1,\ldots,m, \tag{7}$$
$$\boldsymbol{L}^*\succ\boldsymbol{0}, \tag{8}$$
$$z_i^*\ge 0,\quad i=1,\ldots,m. \tag{9}$$

Many algorithms have been proposed for solving problems $\mathbb{Q}$ and $\mathbb{Q}_*$. These can be categorized into mainly two types: conditional gradient algorithms (also referred to as

Frank-Wolfe algorithms) and interior-point algorithms. Below, we survey the studies on these two algorithms.

Khachiyan (1996) proposed a barycentric coordinate descent algorithm, which can be interpreted as a conditional gradient algorithm. He showed that the algorithm has a polynomial-time iteration complexity. Several researchers investigated and revised Khachiyan's algorithm. Kumar and Yildirim (2005) showed that the iteration complexity of Khachiyan's algorithm can be slightly reduced if it starts from a well-selected initial point. Todd and Yildirim (2007) and Ahipasaoglu et al. (2008) incorporated a step called as a Wolfe's away-step. The revised algorithm was shown to have a polynomial-time iteration complexity and a linear convergence rate.

A dual interior-point algorithm was given by Vandenberghe et al. (1998). A primal-dual interior-point algorithm was given by Toh (1999), and numerical experiments showed that this algorithm is efficient and can provide accurate solutions. A practical algorithm was designed by Sun and Freund (2004) for solving large-scale problems. In particular, a hybrid of the interior-point algorithm and cutting plane algorithm was shown to be efficient in numerical experiments. For instance, the paper reported that the hybrid algorithm can solve problems with $d = 30$ and $m = 30,000$ in under 30 seconds on a personal computer. Tsuchiya and Xia (2007) considered generalized forms of $\mathbb{Q}$ and $\mathbb{Q}_*$ and showed that a primal-dual interior-point algorithm for the generalized forms has a polynomial-time iteration complexity.

Next, let us discuss the complexity of these two sorts of algorithms for $\mathbb{Q}$ and $\mathbb{Q}_*$. In each iteration, the arithmetic operations of the conditional gradient algorithms are less than those of the interior-point algorithms. Each iteration of a conditional gradient algorithm (Khachiyan, 1996; Kumar and Yildirim, 2005; Todd and Yildirim, 2007; Ahipasaoglu et al., 2008) requires $O(md)$ arithmetic operations. On the other hand, assuming that the number of data points m is sufficiently larger than the dimension of data points d, the main complexity of interior-point algorithms (Vandenberghe et al., 1998; Toh, 1999) comes from solving an m-by-m system of linear equations in each iteration. The solution serves as the search direction for the next iteration. Solving these linear equations requires $O(m^3)$ arithmetic operations. In practice, the number of iterations of conditional gradient algorithms is much larger than that of interior-point algorithms. Ahipasaoglu et al. (2008) reports that conditional gradient algorithms take several thousands iterations to solve problems such that d runs from 10 to 30 and m from $10,000$ to $30,000$. On the other hand, Sun and Freund (2004) reports that interior-point algorithms usually terminate after several dozen iterations and provide accurate solutions.

One of the concerns about interior-point algorithms is the computational cost of each iteration. It is possible to reduce the cost considerably by using a cutting plane strategy. A hybrid of interior-point algorithm and cutting plane algorithm has an advantage over conditional gradient algorithms. In fact, Ahipasaoglu et al. (2008) reports that the hybrid algorithm is faster than the conditional gradient algorithms and works well even on large problems. Therefore, we use the hybrid algorithm to solve $\mathbb{Q}$ in our practical implementation of ER. The details are in Section 5.1.

Here, it should be mentioned that this paper uses a terminology "cutting plane strategy" for what other papers (Sun and Freund, 2004; Ahipasaoglu et al., 2008) have called the

"active set strategy", since it might be confused with "active set algorithm" for solving a nonnegative least square problem.

4. Description and Analysis of the Algorithm

The ER algorithm is presented below. Throughout of this paper, we use the notation $\mathbb{N}$ to denote a set of nonnegative integer numbers.

Algorithm 1 Ellipsoidal Rounding (ER) for Problem 1

Input: $\boldsymbol{M} \in \mathbb{R}_+^{d\times m}$ and $r \in \mathbb{N}$.
Output: $\mathcal{I}$.

1: Compute the SVD of $\boldsymbol{M}$, and construct the reduced matrix $\boldsymbol{P} \in \mathbb{R}^{r\times m}$ associated with r.

2: Let $\mathcal{S} = \{\pm\boldsymbol{p}_1, \ldots, \pm\boldsymbol{p}_m\}$ for the column vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m$ of $\boldsymbol{P}$. Solve $\mathbb{Q}(\mathcal{S})$, and construct the active index set $\mathcal{I}$.

Step 1 needs to be explained in detail. Let $\boldsymbol{M}$ be a noisy separable matrix of size d-by-m. In general, the $\boldsymbol{M}$ is a full-rank due to the existence of a noise matrix. However, the rank is close to r when the amount of noise is small, and in particular, it is r in the noiseless case. Accordingly, we construct a low-rank approximation matrix to $\boldsymbol{M}$ and reduce the redundancy in the space spanned by the column vectors of $\boldsymbol{M}$.

We use an SVD for the construction of the low-rank approximation matrix. The SVD of $\boldsymbol{M}$ gives a decomposition of the form,

$$\boldsymbol{M} = \boldsymbol{U}\Sigma\boldsymbol{V}^\top.$$

Here, $\boldsymbol{U}$ and $\boldsymbol{V}$ are d-by-d and m-by-m orthogonal matrices, respectively. In this paper, we call the $\boldsymbol{U}$ a *left orthogonal matrix* of the SVD of $\boldsymbol{M}$. Let $t = \min\{d, m\}$. Σ is a rectangular diagonal matrix consisting of the singular values $\sigma_1, \ldots, \sigma_t$ of $\boldsymbol{M}$, and it is of the form,

$$\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_t) \in \mathbb{R}^{d\times m}$$

with $\sigma_1 \geq \cdots \geq \sigma_t \geq 0$. By choosing the top r singular values while setting the others to 0 in Σ, we construct

$$\Sigma^r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0) \in \mathbb{R}^{d\times m}$$

and let

$$\boldsymbol{M}^r = \boldsymbol{U}\Sigma^r\boldsymbol{V}^\top.$$

$\boldsymbol{M}^r$ is the best rank-r approximation to $\boldsymbol{M}$ as measured by the matrix 2-norm and satisfies $||\boldsymbol{M} - \boldsymbol{M}^r||_2 = \sigma_{r+1}$ (see, for instance, Theorem 2.5.3 of Golub and Loan, 1996). By applying the left orthogonal matrix $\boldsymbol{U}^\top$ to $\boldsymbol{M}^r$, we have

$$\boldsymbol{U}^\top\boldsymbol{M}^r = \begin{pmatrix} \boldsymbol{P} \\ \boldsymbol{0} \end{pmatrix} \in \mathbb{R}^{d\times m},$$

where $\boldsymbol{P}$ is an r-by-m matrix with $\mathrm{rank}(\boldsymbol{P}) = r$. We call such a matrix $\boldsymbol{P}$ a *reduced matrix of $\boldsymbol{M}$ associated with r*. Since Assumption 2 holds for the $\boldsymbol{P}$, it is possible to perform an MVEE computation for a set of the column vectors.

4.1 Correctness for a Separable Matrix

We analyze the correctness property of Algorithm 1. Let $\boldsymbol{A}$ be a separable matrix of size d-by-m. Assume that Assumption 1 holds for $\boldsymbol{A}$. We run Algorithm 1 for $(\boldsymbol{A}, \mathrm{rank}(\boldsymbol{A}))$. Step 1 computes the reduced matrix $\boldsymbol{P}$ of $\boldsymbol{A}$. Since $r = \mathrm{rank}(\boldsymbol{A})$, we have $\boldsymbol{A} = \boldsymbol{A}^r$, where $\boldsymbol{A}^r$ is the best rank-r approximation matrix to $\boldsymbol{A}$. Let $\boldsymbol{U} \in \mathbb{R}^{d\times d}$ be the left orthogonal matrix of the SVD of $\boldsymbol{A}$. The reduced matrix $\boldsymbol{P} \in \mathbb{R}^{r\times m}$ of $\boldsymbol{A}$ is obtained as

$$\begin{aligned} \begin{pmatrix} \boldsymbol{P} \\ \boldsymbol{0} \end{pmatrix} &= \boldsymbol{U}^\top \boldsymbol{A} \\ &= \boldsymbol{U}^\top \boldsymbol{F}(\boldsymbol{I}, \boldsymbol{K})\boldsymbol{\Pi}. \end{aligned} \tag{10}$$

From the above, we see that

$$\boldsymbol{U}^\top \boldsymbol{F} = \begin{pmatrix} \boldsymbol{G} \\ \boldsymbol{0} \end{pmatrix} \in \mathbb{R}^{d\times m}, \text{ where } \boldsymbol{G} \in \mathbb{R}^{r\times r}. \tag{11}$$

Here, we have $\mathrm{rank}(\boldsymbol{G}) = r$ since $\mathrm{rank}(\boldsymbol{F}) = r$ by Assumption 1-b and $\boldsymbol{U}$ is an orthogonal matrix. By using $\boldsymbol{G}$, we rewrite $\boldsymbol{P}$ as

$$\boldsymbol{P} = (\boldsymbol{G}, \boldsymbol{G}\boldsymbol{K})\boldsymbol{\Pi}.$$

From Assumption 1-a, the column vectors $\boldsymbol{k}_i$ of the weight matrix $\boldsymbol{K} \in \mathbb{R}^{r\times \ell}$ satisfy the conditions

$$||\boldsymbol{k}_i||_1 = 1 \text{ and } \boldsymbol{k}_i \geq \boldsymbol{0}, \quad i = 1, \ldots, \ell. \tag{12}$$

In Step 2, we collect the column vectors of $\boldsymbol{P}$ and construct a set $\mathcal{S}$ of them. Let $\boldsymbol{B} = \boldsymbol{G}\boldsymbol{K}$, and let $\boldsymbol{g}_j$ and $\boldsymbol{b}_i$ be the column vector of $\boldsymbol{G}$ and $\boldsymbol{B}$, respectively. $\mathcal{S}$ is a set of vectors $\pm\boldsymbol{g}_1, \ldots, \pm\boldsymbol{g}_r, \pm\boldsymbol{b}_1, \ldots, \pm\boldsymbol{b}_\ell$. The following proposition guarantees that the active points of $\mathbb{Q}(\mathcal{S})$ are $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_r$. We can see from (10) and (11) that the index set of the column vectors of $\boldsymbol{G}$ is identical to that of of $\boldsymbol{F}$. Hence, the basis matrix $\boldsymbol{F}$ of a separable one $\boldsymbol{A}$ can be obtained by finding the active points of $\mathbb{Q}(\mathcal{S})$.

Proposition 3 *Let $\boldsymbol{G} \in \mathbb{R}^{r\times r}$ and $\boldsymbol{B} = \boldsymbol{G}\boldsymbol{K} \in \mathbb{R}^{r\times \ell}$ for $\boldsymbol{K} \in \mathbb{R}^{r\times \ell}$. For the column vectors $\boldsymbol{g}_j$ and $\boldsymbol{b}_i$ of $\boldsymbol{G}$ and $\boldsymbol{B}$, respectively, let $\mathcal{S} = \{\pm\boldsymbol{g}_1, \ldots \pm \boldsymbol{g}_r, \pm\boldsymbol{b}_1, \ldots, \pm\boldsymbol{b}_\ell\}$. Suppose that rank$(\boldsymbol{G}) = r$ and $\boldsymbol{K}$ satisfies the condition (12). Then, the active point set of $\mathbb{Q}(\mathcal{S})$ is $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_r\}$.*

Proof We show that an optimal solution $\boldsymbol{L}^*$ of $\mathbb{Q}(\mathcal{S})$ is $(\boldsymbol{G}\boldsymbol{G}^\top)^{-1}$ and its associated Lagrange multiplier $\boldsymbol{z}^*$ is $(\boldsymbol{e}; \boldsymbol{0})$, where $\boldsymbol{e}$ is an r-dimensional all-ones vector and $\boldsymbol{0}$ is an ℓ-dimensional zero vector. Here, the Lagrange multipliers are one for the constraints $\langle \boldsymbol{g}_j\boldsymbol{g}_j^\top, \boldsymbol{L}\rangle \leq 1$, and these are zero for $\langle \boldsymbol{b}_i\boldsymbol{b}_i^\top, \boldsymbol{L}\rangle \leq 1$.

Since $\boldsymbol{G}$ is nonsingular, the inverse of $\boldsymbol{G}\boldsymbol{G}^\top$ exists and it is positive definite. Now we check that $\boldsymbol{L}^* = (\boldsymbol{G}\boldsymbol{G}^\top)^{-1}$ and $\boldsymbol{z}^* = (\boldsymbol{e}; \boldsymbol{0})$ satisfy the KKT conditions (5)-(9) for the problem. It was already seen that the conditions (5), (8), and (9) are satisfied. For the remaining conditions, we have

$$\langle \boldsymbol{g}_j\boldsymbol{g}_j^\top, (\boldsymbol{G}\boldsymbol{G}^\top)^{-1}\rangle = (\boldsymbol{G}^\top(\boldsymbol{G}\boldsymbol{G}^\top)^{-1}\boldsymbol{G})_{jj} = 1 \tag{13}$$

and

$$\begin{aligned}\langle \boldsymbol{b}_i \boldsymbol{b}_i^\top, (\boldsymbol{G}\boldsymbol{G}^\top)^{-1} \rangle &= (\boldsymbol{B}^\top (\boldsymbol{G}\boldsymbol{G}^\top)^{-1} \boldsymbol{B})_{ii} \\ &= (\boldsymbol{K}^\top \boldsymbol{G}^\top (\boldsymbol{G}\boldsymbol{G}^\top)^{-1} \boldsymbol{G}\boldsymbol{K})_{ii} \\ &= \boldsymbol{k}_i^\top \boldsymbol{k}_i \\ &\leq ||\boldsymbol{k}_i||_1^2 = 1. \end{aligned} \tag{14}$$

Here, $(\cdot)_{ii}$ for a matrix denotes the (i,i)th element of the matrix. The inequality in (14) follows from condition (12). Also, the Lagrange multipliers are zero for the inequality constraints $\langle \boldsymbol{b}_i \boldsymbol{b}_i^\top, (\boldsymbol{G}\boldsymbol{G}^\top)^{-1} \rangle \leq 1$. Thus, conditions (6) and (7) are satisfied. Accordingly, $(\boldsymbol{G}\boldsymbol{G}^\top)^{-1}$ is an optimal solution of $\mathbb{Q}(\mathcal{S})$.

We can see from (13) that $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_r$ are the active points of the problem. Moreover, we may have equality in (14). In fact, equality holds if and only if $\boldsymbol{k}_i$ has only one nonzero element. For such $\boldsymbol{k}_i$, $\boldsymbol{b}_i = \boldsymbol{G}\boldsymbol{k}_i$ coincides with some vector in $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_r$. ■

From the above discussion, we can immediately notice that this proposition holds if for a matrix $\boldsymbol{K} \in \mathbb{R}^{r \times \ell}$, the column vectors $\boldsymbol{k}_i$ satisfy

$$||\boldsymbol{k}_i||_2 < 1, \quad i = 1, \ldots, m. \tag{15}$$

Note that in contrast with condition (12), this condition does not require the matrix to be nonnegative.

Corollary 4 *Proposition 3 holds even if we suppose that $\boldsymbol{K} \in \mathbb{R}^{r \times \ell}$ satisfies condition (15), instead of condition (12).*

Note that this corollary is used to show the robustness of Algorithm 1 on a noisy separable matrix. The correctness of Algorithm 1 for a separable matrix follows from the above discussion and Proposition 3.

Theorem 5 *Let $\boldsymbol{A}$ be a separable matrix. Assume that Assumption 1 holds for $\boldsymbol{A}$. Then, Algorithm 1 for $(\boldsymbol{A}, rank(\boldsymbol{A}))$ returns an index set $\mathcal{I}$ such that $\boldsymbol{A}(\mathcal{I}) = \boldsymbol{F}$.*

4.2 Robustness for a Noisy Separable Matrix

Next, we analyze the robustness property of Algorithm 1. Let $\boldsymbol{A}$ be a separable matrix of size d-by-m. Assume that Assumption 1 holds for $\boldsymbol{A}$. Let $\widetilde{\boldsymbol{A}}$ be a noisy separable matrix of the form $\boldsymbol{A} + \boldsymbol{N}$. We run Algorithm 1 for $(\widetilde{\boldsymbol{A}}, \mathrm{rank}(\boldsymbol{A}))$. Step 1 computes the reduced matrix $\boldsymbol{P}$ of $\widetilde{\boldsymbol{A}}$. Let $\boldsymbol{U} \in \mathbb{R}^{d \times d}$ be the left orthogonal matrix of the SVD of $\widetilde{\boldsymbol{A}}$, and $\widetilde{\boldsymbol{A}}^r$ be the best rank-r approximation matrix to $\widetilde{\boldsymbol{A}}$. We denote the residual matrix $\widetilde{\boldsymbol{A}} - \widetilde{\boldsymbol{A}}^r$ by $\widetilde{\boldsymbol{A}}^r_\diamond$.

For the reduced matrix $\boldsymbol{P}$ of $\widetilde{\boldsymbol{A}}$, we have

$$\begin{aligned}
\begin{pmatrix} \boldsymbol{P} \\ \boldsymbol{0} \end{pmatrix} &= \boldsymbol{U}^\top \widetilde{\boldsymbol{A}}^r \\
&= \boldsymbol{U}^\top (\widetilde{\boldsymbol{A}} - \widetilde{\boldsymbol{A}}^r_\diamond) && (16) \\
&= \boldsymbol{U}^\top (\boldsymbol{A} + \boldsymbol{N} - \widetilde{\boldsymbol{A}}^r_\diamond) \\
&= \boldsymbol{U}^\top (\boldsymbol{A} + \bar{\boldsymbol{N}}) && (17) \\
&= \boldsymbol{U}^\top ((\boldsymbol{F}, \boldsymbol{F}\boldsymbol{K})\boldsymbol{\Pi} + \bar{\boldsymbol{N}}) \\
&= \boldsymbol{U}^\top (\boldsymbol{F} + \bar{\boldsymbol{N}}^{(1)}, \boldsymbol{F}\boldsymbol{K} + \bar{\boldsymbol{N}}^{(2)})\boldsymbol{\Pi} && (18) \\
&= \boldsymbol{U}^\top (\widehat{\boldsymbol{F}}, \widehat{\boldsymbol{F}}\boldsymbol{K} + \widehat{\boldsymbol{N}})\boldsymbol{\Pi}. && (19)
\end{aligned}$$

The following notation is used in the above: $\bar{\boldsymbol{N}} = \boldsymbol{N} - \widetilde{\boldsymbol{A}}^r_\diamond$ in (17); $\bar{\boldsymbol{N}}^{(1)}$ and $\bar{\boldsymbol{N}}^{(2)}$ in (18) are the d-by-r and d-by-ℓ submatrices of $\bar{\boldsymbol{N}}$ such that $\bar{\boldsymbol{N}}\boldsymbol{\Pi}^{-1} = (\bar{\boldsymbol{N}}^{(1)}, \bar{\boldsymbol{N}}^{(2)})$; $\widehat{\boldsymbol{F}} = \boldsymbol{F} + \bar{\boldsymbol{N}}^{(1)}$ and $\widehat{\boldsymbol{N}} = -\bar{\boldsymbol{N}}^{(1)}\boldsymbol{K} + \bar{\boldsymbol{N}}^{(2)}$ in (19). This implies that

$$\boldsymbol{U}^\top \widehat{\boldsymbol{F}} = \begin{pmatrix} \widehat{\boldsymbol{G}} \\ \boldsymbol{0} \end{pmatrix}, \text{ where } \widehat{\boldsymbol{G}} \in \mathbb{R}^{r \times r}, \tag{20}$$

and

$$\boldsymbol{U}^\top \widehat{\boldsymbol{N}} = \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{pmatrix}, \text{ where } \boldsymbol{R} \in \mathbb{R}^{r \times \ell}. \tag{21}$$

Hence, we can rewrite $\boldsymbol{P}$ as

$$\boldsymbol{P} = (\widehat{\boldsymbol{G}}, \widehat{\boldsymbol{G}}\boldsymbol{K} + \boldsymbol{R})\boldsymbol{\Pi}.$$

$\widetilde{\boldsymbol{A}}$ is represented by (2) as

$$\widetilde{\boldsymbol{A}} = (\widetilde{\boldsymbol{F}}, \widetilde{\boldsymbol{F}}\boldsymbol{K} + \widetilde{\boldsymbol{N}})\boldsymbol{\Pi},$$

where $\widetilde{\boldsymbol{F}}$ and $\widetilde{\boldsymbol{N}}$ denote $\boldsymbol{F} + \boldsymbol{N}^{(1)}$ and $-\boldsymbol{N}^{(1)}\boldsymbol{K} + \boldsymbol{N}^{(2)}$, respectively. From (16), we have

$$\begin{pmatrix} (\widehat{\boldsymbol{G}}, \widehat{\boldsymbol{G}}\boldsymbol{K} + \boldsymbol{R})\boldsymbol{\Pi} \\ \boldsymbol{0} \end{pmatrix} = \boldsymbol{U}^\top ((\widetilde{\boldsymbol{F}}, \widetilde{\boldsymbol{F}}\boldsymbol{K} + \widetilde{\boldsymbol{N}})\boldsymbol{\Pi} - \widetilde{\boldsymbol{A}}^r_\diamond).$$

Therefore, the index set of the column vectors of $\widehat{\boldsymbol{G}}$ is identical to that of $\widetilde{\boldsymbol{F}}$. If all the column vectors of $\widehat{\boldsymbol{G}}$ are found in $\boldsymbol{P}$, we can identify $\widetilde{\boldsymbol{F}}$ hidden in $\widetilde{\boldsymbol{A}}$.

In Step 2, we collect the column vectors of $\boldsymbol{P}$ and construct a set $\mathcal{S}$ of them. Let

$$\widehat{\boldsymbol{B}} = \widehat{\boldsymbol{G}}\boldsymbol{K} + \boldsymbol{R}, \tag{22}$$

and let $\widehat{\boldsymbol{g}}_j$ and $\widehat{\boldsymbol{b}}_i$ respectively be the column vectors of $\widehat{\boldsymbol{G}}$ and $\widehat{\boldsymbol{B}}$. $\mathcal{S}$ is a set of vectors $\pm\widehat{\boldsymbol{g}}_1, \ldots, \pm\widehat{\boldsymbol{g}}_r, \pm\widehat{\boldsymbol{b}}_1, \ldots, \pm\widehat{\boldsymbol{b}}_\ell$. We can see from Corollary 4 that, if $\mathrm{rank}(\widehat{\boldsymbol{G}}) = r$ and $\widehat{\boldsymbol{b}}_i$ is written as $\widehat{\boldsymbol{b}}_i = \widehat{\boldsymbol{G}}\widehat{\boldsymbol{k}}_i$ by using $\widehat{\boldsymbol{k}}_i \in \mathbb{R}^r$ with $||\widehat{\boldsymbol{k}}_i||_2 < 1$, the active points of $\mathbb{Q}(\mathcal{S})$ are given as the column vectors $\widehat{\boldsymbol{g}}_1, \ldots, \widehat{\boldsymbol{g}}_r$ of $\widehat{\boldsymbol{G}}$. Below, we examine the amount of noise $\boldsymbol{N}$ such that the conditions of Corollary 4 still hold.

Lemma 6 *Let $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{N} \in \mathbb{R}^{d \times m}$. Then, $|\sigma_i(\widetilde{\boldsymbol{A}}) - \sigma_i(\boldsymbol{A})| \leq ||\boldsymbol{N}||_2$ for each $i = 1, \ldots, t$ where $t = \min\{d, m\}$.*

Proof See Corollary 8.6.2 of Golub and Loan (1996). ■

Lemma 7 *Let* $n = ||\boldsymbol{N}||_2$ *and* $\mu = \mu(\boldsymbol{K})$.

7-a) The matrix $\bar{\boldsymbol{N}}$ *of (17) satisfies* $||\bar{\boldsymbol{N}}||_2 \leq 2n$.

7-b) The column vectors $\boldsymbol{r}_i$ *of matrix* $\boldsymbol{R}$ *of (21) satisfy* $||\boldsymbol{r}_i||_2 \leq 2n(\mu+1)$ *for* $i = 1, \ldots, m$.

7-c) The singular values of matrix $\widehat{\boldsymbol{G}}$ *of (20) satisfy* $|\sigma_i(\widehat{\boldsymbol{G}}) - \sigma_i(\boldsymbol{F})| \leq 2n$ *for* $i = 1, \ldots, r$.

Proof 7-a) Since $\bar{\boldsymbol{N}} = \boldsymbol{N} - \widetilde{\boldsymbol{A}}^r_\diamond$,

$$||\bar{\boldsymbol{N}}||_2 \leq ||\boldsymbol{N}||_2 + ||\widetilde{\boldsymbol{A}}^r_\diamond||_2.$$

We have $||\widetilde{\boldsymbol{A}}^r_\diamond||_2 \leq n$ since $||\widetilde{\boldsymbol{A}}^r_\diamond||_2 = \sigma_{r+1}(\widetilde{\boldsymbol{A}})$ and from Lemma 6, $|\sigma_{r+1}(\widetilde{\boldsymbol{A}}) - \sigma_{r+1}(\boldsymbol{A})| \leq n$. Therefore, $||\bar{\boldsymbol{N}}||_2 \leq 2n$.

7-b) Let $\widehat{\boldsymbol{n}}_i$ be the column vector of the matrix $\widehat{\boldsymbol{N}}$ of (19). Since $\boldsymbol{U}^\top \widehat{\boldsymbol{n}}_i = (\boldsymbol{r}_i; \boldsymbol{0})$ for an orthogonal matrix $\boldsymbol{U}$, we have $||\widehat{\boldsymbol{n}}_i||_2 = ||\boldsymbol{r}_i||_2$. Therefore, we will evaluate $||\widehat{\boldsymbol{n}}_i||_2$. Let $\boldsymbol{k}_i$ and $\bar{\boldsymbol{n}}^{(2)}_i$ be the column vectors of $\boldsymbol{K}$ and $\bar{\boldsymbol{N}}^{(2)}$, respectively. Then, $\widehat{\boldsymbol{n}}_i$ can be represented as $-\bar{\boldsymbol{N}}^{(1)}\boldsymbol{k}_i + \bar{\boldsymbol{n}}^{(2)}_i$. Thus, by Lemma 7-a, we have

$$||\boldsymbol{r}_i||_2 = ||\widehat{\boldsymbol{n}}_i||_2 \leq ||\bar{\boldsymbol{N}}^{(1)}||_2||\boldsymbol{k}_i||_2 + ||\bar{\boldsymbol{n}}^{(2)}_i||_2 \leq 2n(\mu+1).$$

7-c) Since $\boldsymbol{U}^\top \widehat{\boldsymbol{F}} = (\widehat{\boldsymbol{G}}; \boldsymbol{0})$ for an orthogonal matrix $\boldsymbol{U}$, the singular values of $\widehat{\boldsymbol{F}}$ and $\widehat{\boldsymbol{G}}$ are identical. Also, since $\widehat{\boldsymbol{F}} = \boldsymbol{F} + \bar{\boldsymbol{N}}^{(1)}$ and Lemma 6, we have

$$|\sigma_i(\widehat{\boldsymbol{G}}) - \sigma_i(\boldsymbol{F})| = |\sigma_i(\widehat{\boldsymbol{F}}) - \sigma_i(\boldsymbol{F})| \leq ||\bar{\boldsymbol{N}}^{(1)}||_2 \leq 2n.$$

■

The following lemma ensures that the conditions of Corollary 4 hold if the amount of noise is smaller than a certain level.

Lemma 8 *Let* $\widehat{\boldsymbol{G}}$ *be the matrix of (20), and let* $\widehat{\boldsymbol{b}}_i$ *be the column vector of* $\widehat{\boldsymbol{B}}$ *of (22). Suppose that* $||\boldsymbol{N}||_2 < \epsilon$ *for* $\epsilon = \frac{1}{4}\sigma(1-\mu)$ *where* $\sigma = \sigma_r(\boldsymbol{F})$ *and* $\mu = \mu(\boldsymbol{K})$. *Then,*

8-a) $rank(\widehat{\boldsymbol{G}}) = r$.

8-b) $\widehat{\boldsymbol{b}}_i$ *is represented as* $\widehat{\boldsymbol{G}}\widehat{\boldsymbol{k}}_i = \widehat{\boldsymbol{b}}_i$ *by using* $\widehat{\boldsymbol{k}}_i$ *such that* $||\widehat{\boldsymbol{k}}_i||_2 < 1$.

In the proof below, n denotes $||\boldsymbol{N}||_2$.

Proof 8-a) From Lemma 7-c, the minimum singular value of $\widehat{\boldsymbol{G}}$ satisfies

$$\begin{aligned} \sigma_r(\widehat{\boldsymbol{G}}) &\geq \sigma - 2n \\ &> \sigma - 2\epsilon = \frac{1}{2}\sigma(1+\mu) > 0. \end{aligned}$$

The final inequality follows from $\sigma > 0$ due to Assumption 1-b. Hence, we have $\mathrm{rank}(\widehat{\boldsymbol{G}}) = r$.

8-b) Let $\boldsymbol{k}_i$ and $\boldsymbol{r}_i$ be the column vectors of $\boldsymbol{K}$ and $\boldsymbol{R}$, respectively. Then, we have $\widehat{\boldsymbol{b}}_i = \widehat{\boldsymbol{G}}\boldsymbol{k}_i + \boldsymbol{r}_i$. Since Lemma 8-a guarantees that $\widehat{\boldsymbol{G}}$ has an inverse, it can be represented as $\widehat{\boldsymbol{b}}_i = \widehat{\boldsymbol{G}}\widehat{\boldsymbol{k}}_i$ by $\widehat{\boldsymbol{k}}_i = \boldsymbol{k}_i + \widehat{\boldsymbol{G}}^{-1}\boldsymbol{r}_i$. It follows from Lemmas 7-b and 7-c that

$$\begin{aligned} ||\widehat{\boldsymbol{k}}_i||_2 &\leq ||\boldsymbol{k}_i||_2 + ||\widehat{\boldsymbol{G}}^{-1}||_2||\boldsymbol{r}_i||_2 \\ &\leq \mu + \frac{2n(\mu+1)}{\sigma - 2n}. \end{aligned}$$

Since $n < \frac{1}{4}\sigma(1-\mu)$, we have $||\widehat{\boldsymbol{k}}_i||_2 < 1$. ■

The robustness of Algorithm 1 for a noisy separable matrix follows from the above discussion, Corollary 4, and Lemma 8.

Theorem 9 *Let $\widetilde{\boldsymbol{A}}$ be a noisy separable matrix of the form $\boldsymbol{A}+\boldsymbol{N}$. Assume that Assumption 1 holds for the separable matrix $\boldsymbol{A}$ in $\widetilde{\boldsymbol{A}}$. Set $\epsilon = \frac{1}{4}\sigma(1-\mu)$ where $\sigma = \sigma_r(\boldsymbol{F})$ and $\mu = \mu(\boldsymbol{K})$ for the basis and weight matrices $\boldsymbol{F}$ and $\boldsymbol{K}$ of $\boldsymbol{A}$. If $||\boldsymbol{N}||_2 < \epsilon$, Algorithm 1 for $(\widetilde{\boldsymbol{A}}, \mathrm{rank}(\boldsymbol{A}))$ returns an index set $\mathcal{I}$ such that $||\widetilde{\boldsymbol{A}}(\mathcal{I}) - \boldsymbol{F}||_2 < \epsilon$.*

In Theorem 9, let $\boldsymbol{F}^* = \widetilde{\boldsymbol{A}}(\mathcal{I})$, and $\boldsymbol{W}^*$ be an optimal solution of the convex optimization problem,

$$\text{minimize } ||\widetilde{\boldsymbol{A}}(\mathcal{I})\boldsymbol{X} - \widetilde{\boldsymbol{A}}||_F^2 \text{ subject to } \boldsymbol{X} \geq \boldsymbol{0},$$

where the matrix $\boldsymbol{X}$ of size r-by-m is the decision variable. Then, $(\boldsymbol{F}^*, \boldsymbol{W}^*)$ serves as the NMF factor of $\widetilde{\boldsymbol{A}}$. It is possible to evaluate the residual error of this factorization in a similar way to the proof of Theorem 4 by Gillis and Vavasis (2014).

Corollary 10 *Let $\boldsymbol{w}_i^*$ and $\widetilde{\boldsymbol{a}}_i$ be the column vectors of $\boldsymbol{W}^*$ and $\widetilde{\boldsymbol{A}}$, respectively. Then, $||\boldsymbol{F}^*\boldsymbol{w}_i^* - \widetilde{\boldsymbol{a}}_i||_2 < 2\epsilon$ for $i = 1, \ldots, m$.*

Proof From Assumption 1-a, the column vectors $\boldsymbol{w}_i$ of $\boldsymbol{W}$ satisfy $||\boldsymbol{w}_i||_2 \leq 1$ for $i = 1, \ldots, m$. Therefore, for $i = 1, \ldots, m$,

$$\begin{aligned} ||\boldsymbol{F}^*\boldsymbol{w}_i^* - \widetilde{\boldsymbol{a}}_i||_2 &\leq ||\boldsymbol{F}^*\boldsymbol{w}_i - \widetilde{\boldsymbol{a}}_i||_2 \\ &= ||\boldsymbol{F}^*\boldsymbol{w}_i - \boldsymbol{F}\boldsymbol{w}_i + \boldsymbol{F}\boldsymbol{w}_i - \boldsymbol{a}_i - \boldsymbol{n}_i||_2 \\ &= ||(\boldsymbol{F}^* - \boldsymbol{F})\boldsymbol{w}_i - \boldsymbol{n}_i||_2 \\ &\leq ||\boldsymbol{F}^* - \boldsymbol{F}||_2||\boldsymbol{w}_i||_2 + ||\boldsymbol{n}_i||_2 < 2\epsilon, \end{aligned}$$

where $\boldsymbol{a}_i$ and $\boldsymbol{n}_i$ denote the ith column vector of $\boldsymbol{A}$ and $\boldsymbol{N}$, respectively. ■

5. Implementation in Practice

Theorem 9 guarantees that Algorithm 1 correctly identifies the near-basis matrix of a noisy separable matrix if the noise is smaller than some level. But in the NMFs of matrices arising from practical applications, it seems that the noise level would likely exceed the level for

which the theorem is valid. In such a situation, the algorithm might generate more active points than hoped. Therefore, we need to add a selection step in which r points are selected from the active points. Also, the number of active points depends on which dimension we choose in the computation of the reduced matrix $\boldsymbol{P}$. Algorithm 1 computes the reduced matrix $\boldsymbol{P}$ of the data matrix and draws an origin-centered MVEE for the column vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m$ of $\boldsymbol{P}$. As we will see in Lemma 11, the number of active points depends on the dimension of $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m$. Therefore, we introduce an input parameter ρ to control the dimension. By taking account of these considerations, we design a practical implementation of Algorithm 1.

Algorithm 2 Practical Implementation of Algorithm 1

Input: $\boldsymbol{M} \in \mathbb{R}_+^{d\times m}, r \in \mathbb{N}$, and $\rho \in \mathbb{N}$.
Output: $\mathcal{I}$.

1: Run Algorithm 1 for $(\boldsymbol{M}, \rho)$. Let $\mathcal{J}$ be the index set returned by the algorithm.

2: If $|\mathcal{J}| < r$, increase ρ by 1 and go back to Step 1. Otherwise, select r elements from $\mathcal{J}$ and construct the set $\mathcal{I}$ of these elements.

One may wonder whether Algorithm 2 infinitely loops or not. In fact, we can show that under some conditions, infinite loops do not occur.

Lemma 11 *For $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m \in \mathbb{R}^\rho$, let $\mathcal{S} = \{\pm\boldsymbol{p}_1, \ldots, \pm\boldsymbol{p}_m\}$. Suppose that Assumption 2 holds. Then, $\mathbb{Q}(\mathcal{S})$ has at least ρ active points.*

Proof Consider the KKT conditions (5)-(9) for $\mathbb{Q}(\mathcal{S})$. Condition (5) requires $\Omega(\boldsymbol{z}^*)$ to be nonsingular. Since $\mathrm{rank}(\boldsymbol{P}) = \rho$ from the assumption, at least ρ nonzero z_i^* exist. Therefore, we see from condition (6) that $\mathbb{Q}(\mathcal{S})$ has at least ρ active points. ■

Proposition 12 *Suppose that we choose r such that $r \leq \mathrm{rank}(\boldsymbol{M})$. Then, Algorithm 2 terminates after a finite number of iterations.*

Proof For the active index set $\mathcal{J}$ constructed in Step 1, Lemma 11 guarantees that $|\mathcal{J}| \geq \rho$. The parameter ρ increases by 1 if $|\mathcal{J}| < r$ in Step 2 and can continue to increase up to $\rho = \mathrm{rank}(\boldsymbol{M})$. Since $r \leq \mathrm{rank}(\boldsymbol{M})$, it is necessarily to satisfy $|\mathcal{J}| \geq \rho \geq r$ after a finite number of iterations. ■

Proposition 12 implies that ρ may not be an essential input parameter since Algorithm 2 always terminates under $r \leq \mathrm{rank}(\boldsymbol{M})$ even if starting with $\rho = 1$.

There are some concerns about Algorithm 2. One is in how to select r elements from an active index set $\mathcal{J}$ in Step 2. It is possible to have various ways to make the selection. We rely on existing algorithms, such as XRAY and SPA, and perform these existing algorithms for $(\boldsymbol{M}(\mathcal{J}), \rho)$. Thus, Algorithm 1 can be regarded as a preprocessor which filters out basis vector candidates from the data points and enhance the performance of existing algorithms. Another concern is in the computational cost of solving $\mathbb{Q}$. In the next section, we describe a cutting plane strategy for efficiently performing an interior-point algorithm.

5.1 Cutting Plane Strategy for Solving $\mathbb{Q}$

Let $\mathcal{S}$ be a set of m points in $\mathbb{R}^d$. As mentioned in Section 3, $O(m^3)$ arithmetic operations are required in each iteration of an interior-point algorithm for $\mathbb{Q}(\mathcal{S})$. A cutting plane strategy is a way to reduce the number of points which we need to deal with in solving $\mathbb{Q}(\mathcal{S})$. The strategy was originally used by Sun and Freund (2004). In this section, we describe the details of our implementation.

The cutting plane strategy for solving $\mathbb{Q}$ has a geometric interpretation. It is thought of that active points contribute a lot to the drawing the MVEE for a set of points but inactive points make less of a contribution. This geometric intuition can be justified by the following proposition. Let $\boldsymbol{L}$ be a d-by-d matrix. We use the notation $\delta_{\boldsymbol{L}}(\boldsymbol{p})$ to denote $\langle \boldsymbol{p}\boldsymbol{p}^\top, \boldsymbol{L}\rangle$ for an element $\boldsymbol{p} \in \mathbb{R}^d$ of $\mathcal{S}$.

Proposition 13 *Let $\bar{\mathcal{S}}$ be a subset of $\mathcal{S}$. If an optimal solution $\bar{\boldsymbol{L}}^*$ of $\mathbb{Q}(\bar{\mathcal{S}})$ satisfies $\delta_{\bar{\boldsymbol{L}}^*}(\boldsymbol{p}) \leq 1$ for all $p \in \mathcal{S} \setminus \bar{\mathcal{S}}$, then $\bar{\boldsymbol{L}}^*$ is an optimal solution of $\mathbb{Q}(\mathcal{S})$.*

The proof is omitted since it is obvious. The proposition implies that $\mathbb{Q}(\mathcal{S})$ can be solved by using its subset $\bar{\mathcal{S}}$ instead of $\mathcal{S}$. The cutting plane strategy offers a way of finding such a $\bar{\mathcal{S}}$, in which a smaller problem $\mathbb{Q}(\bar{\mathcal{S}})$ has the same optimal solution as $\mathbb{Q}(\mathcal{S})$. In this strategy, we first choose some points from $\mathcal{S}$ and construct a set $\mathcal{S}^1$ containing these points. Let $\mathcal{S}^k$ be the set constructed in the kth iteration. In the $(k+1)$th iteration, we choose some points from $\mathcal{S} \setminus \mathcal{S}^k$ and expand $\mathcal{S}^k$ to $\mathcal{S}^{k+1}$ by adding these points to $\mathcal{S}^k$. Besides expanding, we also shrink $\mathcal{S}^k$ by discarding some points which can be regarded as useless for drawing the origin-centered MVEE. These expanding and shrinking phases play an important role in constructing a small set. Algorithm 3 describes a cutting plane strategy for solving $\mathbb{Q}(\mathcal{S})$.

Algorithm 3 Cutting Plane Strategy for Solving $\mathbb{Q}(\mathcal{S})$

Input: $\mathcal{S} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m\}$.
Output: $\boldsymbol{L}^*$.

1: Choose an initial set $\mathcal{S}^1$ from $\mathcal{S}$ and let $k = 1$.

2: Solve $\mathbb{Q}(\mathcal{S}^k)$ and find the optimal solution $\boldsymbol{L}^k$. If $\delta_{\boldsymbol{L}^k}(\boldsymbol{p}) \leq 1$ holds for all $\boldsymbol{p} \in \mathcal{S} \setminus \mathcal{S}^k$, let $\boldsymbol{L}^* = \boldsymbol{L}^k$, and stop.

3: Choose a subset $\mathcal{F}$ of $\mathcal{S}^k$ and a subset $\mathcal{G}$ of $\{\boldsymbol{p} \in \mathcal{S} \setminus \mathcal{S}^k : \delta_{\boldsymbol{L}^k}(\boldsymbol{p}) > 1\}$. Update $\mathcal{S}^k$ as $\mathcal{S}^{k+1} = (\mathcal{S}^k \setminus \mathcal{F}) \cup \mathcal{G}$ and increase k by 1. Then, go back to Step 2.

Now, we give a description of our implementation of Algorithm 3. To construct the initial set $\mathcal{S}^1$ in Step 1, our implementation employs the algorithm used in the papers (Kumar and Yildirim, 2005; Todd and Yildirim, 2007; Ahipasaoglu et al., 2008). The algorithm constructs a set $\mathcal{S}^1$ by greedily choosing $2d$ points in a step-by-step manner such that the convex hull is a d-dimensional crosspolytope containing as many points in $\mathcal{S}$ as possible. We refer the reader to Algorithm 3.1 of Kumar and Yildirim (2005) for the precise description.

To shrink and expand $\mathcal{S}^k$ in Step 3, we use a shrinking threshold parameter θ such that $\theta < 1$, and an expanding size parameter η such that $\eta \geq 1$. These parameters are set before running Algorithm 3. For shrinking, we construct $\mathcal{F} = \{\boldsymbol{p} \in \mathcal{S}^k : \delta_{\boldsymbol{L}^k}(\boldsymbol{p}) \leq \theta\}$ by using θ.

For expanding, we arrange the points of $\{\boldsymbol{p} \in \mathcal{S} \setminus \mathcal{S}^k : \delta_{\boldsymbol{L}^k}(\boldsymbol{p}) > 1\}$ in descending order, as measured by $\delta_{\boldsymbol{L}^k}(\cdot)$, and construct $\mathcal{G}$ by choosing the top $(m-2d)/\eta$ points. If the set $\{\boldsymbol{p} \in \mathcal{S} \setminus \mathcal{S}^k : \delta_{\boldsymbol{L}^k}(\boldsymbol{p}) > 1\}$ has less than $(m-2d)/\eta$ points, we choose all the points and construct $\mathcal{G}$.

6. Experiments

We experimentally compared Algorithm 2 with SPA and the variants of XRAY. These two existing algorithms were chosen because their studies (Bittorf et al., 2012; Gillis and Luce, 2013; Kumar et al., 2013) report that they outperform AGKM and Hottopixx, and scale to the problem size. Two types of experiments were conducted: one is the evaluation for the robustness of the algorithms to noise on synthetic data sets, and the other is the application of the algorithms to clustering of real-world document corpora.

We implemented Algorithm 2, and three variants of XRAY, "max", "dist" and "greedy", in MATLAB. We put Algorithm 3 in Algorithm 2 so it would solve $\mathbb{Q}$ efficiently. The software package SDPT3 (Toh et al., 1999) was used for solving $\mathbb{Q}(\mathcal{S}^k)$ in Step 2 of Algorithm 3. The shrinking parameter θ and expanding size parameter η were set as 0.9999 and 5, respectively. The implementation of XRAY formulated the computation of the residual matrix $\boldsymbol{R} = \boldsymbol{A}(\mathcal{I}_k)\boldsymbol{X}^* - \boldsymbol{A}$ as a convex optimization problem,

$$\boldsymbol{X}^* = \arg\min_{\boldsymbol{X} \geq \boldsymbol{0}} ||\boldsymbol{A}(\mathcal{I}_k)\boldsymbol{X} - \boldsymbol{A}||_F^2.$$

For the implementation of SPA (Gillis and Vavasis, 2014), we used code from the first author's website. Note that SPA and XRAY are sensitive to the normalization of the column vectors of the data matrix (see, for instance, Kumar et al., 2013), and for this reason, we used a data matrix whose column vectors were not normalized. All experiments were done in MATLAB on a 3.2 GHz CPU processor and 12 GB memory.

We will use the following abbreviations to represent the variants of algorithms. For instance, Algorithm 2 with SPA for an index selection of Step 2 is referred to as ER-SPA. Also, the variant of XRAY with "max" selection policy is referred to as XRAY(max).

6.1 Synthetic Data

Experiments were conducted for the purpose of seeing how well Algorithm 2 could improve the robustness of SPA and XRAY to noise. Specifically, we compared it with SPA, XRAY(max), XRAY(dist), and XRAY(greedy). The robustness of algorithm was measured by a recovery rate. Let $\mathcal{I}$ be an index set of basis vectors in a noisy separable matrix, and $\mathcal{I}^*$ be an index set returned by an algorithm. The recovery rate is the ratio given by $|\mathcal{I} \cap \mathcal{I}^*| \ / \ |\mathcal{I}|$.

We used synthetic data sets of the form $\boldsymbol{F}(\boldsymbol{I}, \boldsymbol{K})\boldsymbol{\Pi} + \boldsymbol{N}$ with $d = 250$, $m = 5,000$, and $r = 10$. The matrices $\boldsymbol{F}, \boldsymbol{K}, \boldsymbol{\Pi}$ and $\boldsymbol{N}$ were synthetically generated as follows. The entries of $\boldsymbol{W} \in \mathbb{R}_+^{d \times r}$ were drawn from a uniform distribution on the interval $[0, 1]$. The column vectors of $\boldsymbol{K} \in \mathbb{R}_+^{r \times \ell}$ were from a Dirichlet distribution whose r parameters were uniformly from the interval $[0, 1]$. The permutation matrix $\boldsymbol{\Pi}$ was randomly generated. The entries of the noise matrix $\boldsymbol{N} \in \mathbb{R}^{d \times m}$ were from a normal distribution with mean 0 and standard deviation δ. The parameter δ determined the intensity of the noise, and it was chosen from

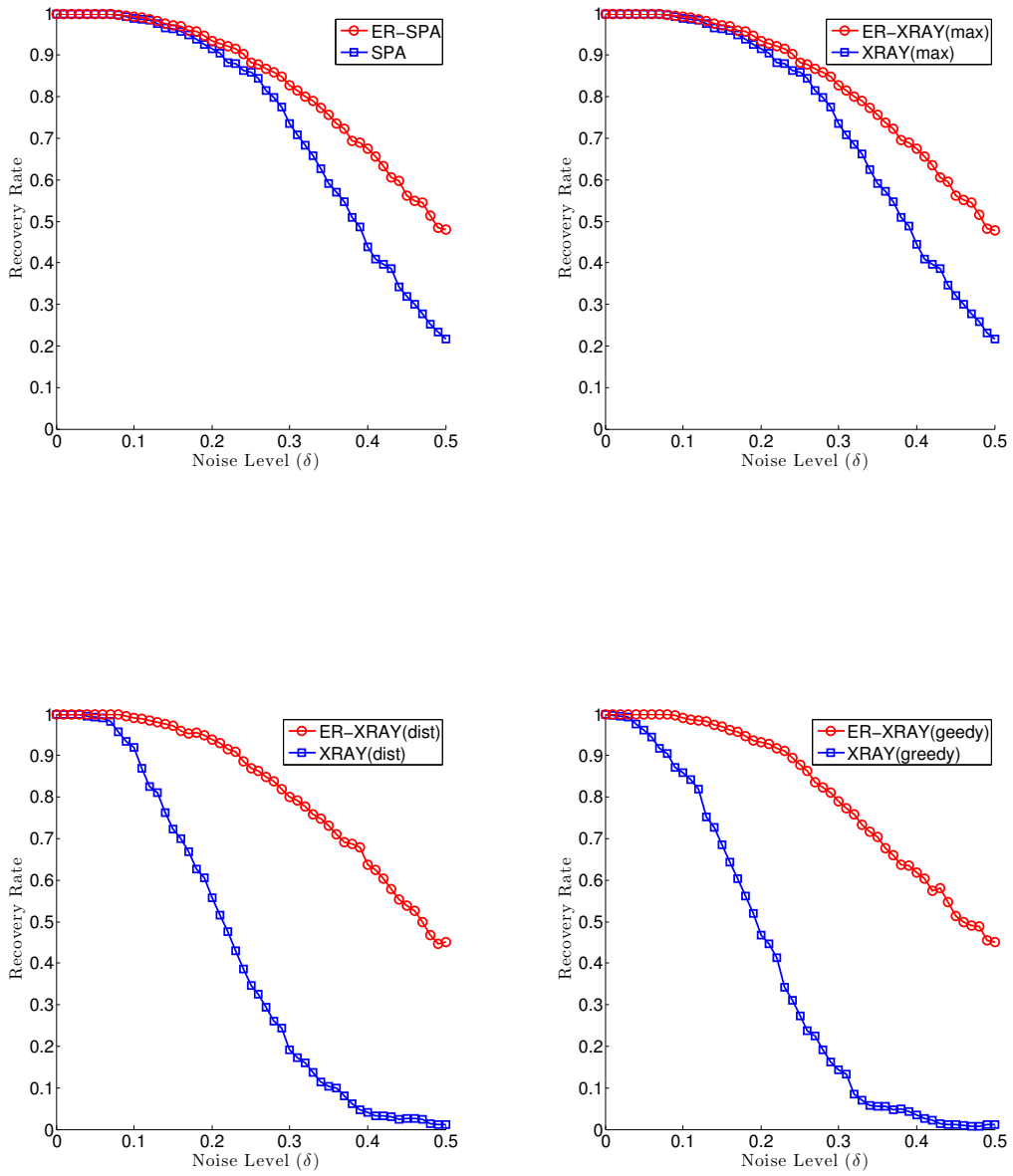


Figure 3: Comparison of the recovery rates of Algorithm 2 with SPA and XRAY.

Recovery rate	100%	90%	80%	70%
ER-SPA	0.06	0.24	0.32	0.37
SPA	0.05	0.21	0.27	0.31
ER-XRAY(max)	0.06	0.24	0.32	0.37
XRAY(max)	0.05	0.21	0.27	0.31
ER-XRAY(dist)	0.07	0.23	0.29	0.36
XRAY(dist)	0.03	0.10	0.13	0.16
ER-XRAY(greedy)	0.07	0.23	0.29	0.35
XRAY(greedy)	0.00	0.08	0.12	0.14

Table 1: Maximum values of noise level δ for different recovery rates in percentage.

0 to 0.5 in 0.01 increments. A single data set consisted of 51 matrices with various amounts of noise, and we made 50 different data sets. Algorithm 2 was performed in the setting that $\boldsymbol{M}$ is a matrix in the data set and r and ρ are each 10.

Figure 3 depicts the average recovery rate on the 50 data sets for Algorithm 2, SPA and XRAY. Table 1 summarizes the maximum values of noise level δ for different recovery rates in percentage. The noise level was measured by 0.01, and hence, for instance, the entry "0.00" at XRAY(greedy) for 100% recovery rate means that the maximum value is in the interval $[0.00, 0.01)$. We see from the figure that Algorithm 2 improved the recovery rates of the existing algorithms. In particular, the recovery rates of XRAY(dist) and XRAY(greedy) rapidly decrease as the noise level increases, but Algorithm 2 significantly improved them. Also, the figure shows that Algorithm 2 tended to slow the decrease in the recovery rate. We see from the table that Algorithm 2 is more robust to noise than SPA and XRAY.

Table 2 summarizes the average number of active points and elapsed time for 50 data sets taken by Algorithm 2 with $\delta = 0, 0.25$ and 0.5. We read from the table that the elapsed time increases with the number of active points. The average elapsed times of SPA, XRAY(max), XRAY(dist), and XRAY(greedy) was respectively 0.03, 1.18, 16.80 and 15.85 in seconds. Therefore, we see that the elapsed time of Algorithm 2 was within a reasonable range.

δ	Active points	Elapsed time (second)			
		ER-SPA	ER-XRAY(max)	ER-XRAY(dist)	ER-XRAY(greedy)
0	10	1.05	1.07	1.07	1.07
0.25	12	3.08	3.10	3.10	3.10
0.5	23	4.70	4.71	4.71	4.71

Table 2: Average number of active points and elapsed time of Algorithm 2.

6.2 Application to Document Clustering

Consider a set of d documents. Let m be the total number of words appearing in the document set. We represent the documents by a bag-of-words. That is, the ith document is represented as an m-dimensional vector $\boldsymbol{a}_i$, whose elements are the appearance frequencies of words in the document. A document vector $\boldsymbol{a}_i$ can be assumed to be generated by a

convex combination of several topic vectors $\boldsymbol{w}_1, \dots \boldsymbol{w}_r$. This type of generative model has been used in many papers (for instance, Xu et al., 2003; Shahnaz et al., 2006; Arora et al., 2012b, 2013; Ding et al., 2013; Kumar et al., 2013).

Let $\boldsymbol{W}$ be an r-by-m topic matrix such that $\boldsymbol{w}_1^\top, \dots, \boldsymbol{w}_r^\top$ are stacked from top to bottom and are of the form $(\boldsymbol{w}_1; \dots; \boldsymbol{w}_r)$. The model allows us to write a document vector in the form $\boldsymbol{a}_i^\top = \boldsymbol{f}_i^\top \boldsymbol{W}$ by using a coefficient vector $\boldsymbol{f}_i \in \mathbb{R}^r$ such that $\boldsymbol{e}^\top \boldsymbol{f}_i = 1$ and $\boldsymbol{f}_i \geq \boldsymbol{0}$. This means that we have $\boldsymbol{A} = \boldsymbol{F}\boldsymbol{W}$ for a document-by-word matrix $\boldsymbol{A} = (\boldsymbol{a}_1; \dots; \boldsymbol{a}_d) \in \mathbb{R}_+^{d \times m}$, a coefficient matrix $\boldsymbol{F} = (\boldsymbol{f}_1; \dots; \boldsymbol{f}_d) \in \mathbb{R}_+^{d \times r}$, and a topic matrix $\boldsymbol{W} = (\boldsymbol{w}_1; \dots; \boldsymbol{w}_r) \in \mathbb{R}_+^{r \times m}$. In the same way as the papers (Arora et al., 2012b, 2013; Ding et al., 2013; Kumar et al., 2013), we assume that a document-by-word matrix $\boldsymbol{A}$ is separable. This requires that $\boldsymbol{W}$ is of $(\boldsymbol{I}, \boldsymbol{K})\boldsymbol{\Pi}$, and it means that each topic has an *anchor word*. An anchor word is a word that is contained in one topic but not contained in the other topics. If an anchor word is found, it suggests that the associated topic exists.

Algorithms for Problem 1 can be used for clustering documents and finding topics for the above generative model. The algorithms for a document-word matrix $\boldsymbol{A}$ return an index set $\mathcal{I}$. Let $\boldsymbol{F} = \boldsymbol{A}(\mathcal{I})$. The row vector elements $f_{i1}, \dots, f_{ir}$ of $\boldsymbol{F}$ can be thought of as the contribution rate of topics $\boldsymbol{w}_1, \dots, \boldsymbol{w}_r$ for generating a document $\boldsymbol{a}_i$. The highest value f_{ij^*} among the elements implies that the topic $\boldsymbol{w}_{j^*}$ contributes the most to the generation of document $\boldsymbol{a}_i$. Hence, we assign document $\boldsymbol{a}_i$ to a cluster having the topic $\boldsymbol{w}_{j^*}$. There is an alternative to using $\boldsymbol{F}$ for measuring the contribution rates of the topics. Step 1 of Algorithm 1 produces a rank-r approximation matrix $\boldsymbol{A}^r$ to $\boldsymbol{A}$ as a by-product. Let $\boldsymbol{F}' = \boldsymbol{A}^r(\mathcal{I})$, and use it as an alternative to $\boldsymbol{F}$. We say that clustering with $\boldsymbol{F}$ is clustering with the original data matrix, and that clustering with $\boldsymbol{F}'$ is clustering with a low-rank approximation data matrix.

Experiments were conducted in the purpose of investigating clustering performance of algorithms and also checking whether meaningful topics could be extracted. To investigate the clustering performance, we used only SPA since our experimental results implied that XRAY would underperform. We assigned the values of the document-word matrix on the basis of the tf-idf weighting scheme, for which we refer the reader to Manning et al. (2008), and normalized the row vectors to the unit 1-norm.

To evaluate the clustering performance, we measured the accuracy (AC) and normalized mutual information (NMI). These measures are often used for this purpose (see, for instance, Xu et al., 2003; Manning et al., 2008). Let $\Omega_1, \dots, \Omega_r$ be the manually classified classes and $\mathcal{C}_1, \dots, \mathcal{C}_r$ be the clusters constructed by an algorithm. Both Ω_i and $\mathcal{C}_j$ are the subsets of the document set $\{\boldsymbol{a}_1, \dots, \boldsymbol{a}_m\}$ such that each subset does not share any documents and the union of all subsets coincides with the document set. AC is computed as follows. First, compute the correspondence between classes $\Omega_1, \dots, \Omega_r$ and clusters $\mathcal{C}_1, \dots, \mathcal{C}_r$ such that the total number of common documents $\Omega_i \cap \mathcal{C}_j$ is maximized. This computation can be done by solving an assignment problem. After that, rearrange the classes and clusters in the obtained order and compute

$$\frac{1}{d} \sum_{k=1}^{r} |\Omega_k \cap \mathcal{C}_k|.$$

This value is the AC for the clusters constructed by an algorithm. NMI is computed as

$$\frac{I(\Omega, \mathcal{C})}{\frac{1}{2}(E(\Omega) + E(\mathcal{C}))}.$$

I and E denote the mutual information and entropy for the class family Ω and cluster family $\mathcal{C}$ where $\Omega = \{\Omega_1, \ldots, \Omega_r\}$ and $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_r\}$. We refer the reader to Section 16.3 of Manning et al. (2008) for the precise forms of I and E.

Two document corpora were used for the clustering-performance evaluation: Reuters-21578 and 20 Newsgroups. These corpora are publicly available from the UCI Knowledge Discovery in Databases Archive (`http://kdd.ics.uci.edu`). In particular, we used the data preprocessing of Deng Cai, in which multiple classes are discarded. The data sets are available from the website (`http://www.cad.zju.edu.cn/home/dengcai`). The Reuters-21578 corpus consists of 21,578 documents appearing in the Reuters newswire in 1987, and these documents are manually classified into 135 classes. The text corpus is reduced by the preprocessing to 8,293 documents in 65 classes. Furthermore, we cut off classes with less than 5 documents. The resulting corpus contains 8,258 documents with 18,931 words in 48 classes, and the sizes of the classes range from 5 to 3,713. The 20 Newsgroups corpus consists of 18,846 documents with 26,213 words appearing in 20 different newsgroups. The size of each class is about 1,000.

We randomly picked some classes from the corpora and evaluated the clustering performance 50 times. Algorithm 2 was performed in the setting that $\boldsymbol{M}$ is a document-word matrix and r and ρ each are the number of classes. In clustering with a low-rank approximation data matrix, we used the rank-r approximation matrix to a document-word matrix.

	AC				NMI			
	Original		Low-rank approx.		Original		Low-rank approx.	
# Classes	ER-SPA	SPA	ER-SPA	SPA	ER-SPA	SPA	ER-SPA	SPA
6	0.605	0.586	0.658	0.636	0.407	0.397	0.532	0.466
8	0.534	0.539	0.583	0.581	0.388	0.387	0.491	0.456
10	0.515	0.508	0.572	0.560	0.406	0.393	0.511	0.475
12	0.482	0.467	0.532	0.522	0.399	0.388	0.492	0.469

Table 3: (Reuters-21578) Average AC and NMI of ER-SPA and SPA with the original data matrix and low-rank approximation data matrix.

Tables 3 and 4 show the results for Reuters-21578 and 20 Newsgroups, respectively. They summarize the average ACs and NMIs of ER-SPA and SPA. The column with "# Classes" lists the number of classes we chose. The columns labeled "Original" and "Low-rank approx." are respectively the averages of the corresponding clustering measurements with the original data matrix and low-rank approximation data matrix. The tables suggest that clustering with a low-rank approximation data matrix performed better than clustering with the original data matrix. We see from Table 3 that ER-SPA could achieve improvements in the AC and NMI of SPA on Reuters-21578 when the clustering was done with a

	AC				NMI			
	Original		Low-rank approx.		Original		Low-rank approx.	
# Classes	ER-SPA	SPA	ER-SPA	SPA	ER-SPA	SPA	ER-SPA	SPA
6	0.441	0.350	0.652	0.508	0.314	0.237	0.573	0.411
8	0.391	0.313	0.612	0.474	0.306	0.242	0.555	0.415
10	0.356	0.278	0.559	0.439	0.291	0.228	0.515	0.397
12	0.319	0.240	0.517	0.395	0.268	0.205	0.486	0.372

Table 4: (20 Newsgroups) Average AC and NMI of ER-SPA and SPA with the original data matrix and low-rank approximation data matrix.

low-rank approximation data matrix. Table 4 indicates that ER-SPA outperformed SPA in AC and NMI on 20 Newsgroups.

Finally, we compared the topics obtained by ER-SPA and SPA. We used the BBC corpus of Greene and Cunningham (2006), which is available from the website (`http://mlg.ucd.ie/datasets/bbc.html`). The documents in the corpus have been subjected by preprocessed such as stemming, stop-word removal, and low word frequency filtering. It consists of 2,225 documents with 9,636 words that appeared on the BBC news website in 2004-2005. The documents were news on 5 topics: "business", "entertainment", "politics", "sport" and "tech".

AC		NMI	
ER-SPA	SPA	ER-SPA	SPA
0.939	0.675	0.831	0.472

Table 5: AC and NMI of ER-SPA and SPA with low-rank approximation data matrix for BBC.

	Anchor word	1	2	3	4	5
ER-SPA	film	award	best	oscar	nomin	actor
SPA	film	award	best	oscar	nomin	star
ER-SPA	mobil	phone	user	softwar	microsoft	technolog
SPA	mobil	phone	user	microsoft	music	download
ER-SPA	bank	growth	economi	price	rate	oil
SPA	bank	growth	economi	price	rate	oil
ER-SPA	game	plai	player	win	england	club
SPA	fiat	sale	profit	euro	japan	firm
ER-SPA	elect	labour	parti	blair	tori	tax
SPA	blog	servic	peopl	site	firm	game

Table 6: Anchor words and top-5 frequent words in topics grouped by ER-SPA and SPA for BBC.

Table 5 shows the ACs and NMIs of ER-SPA and SPA on the low-rank approximation data matrix for the BBC corpus. The table indicates that the AC and NMI of ER-SPA are higher than those of SPA. Table 6 summarizes the words in the topics obtained by ER-SPA and SPA. The topics were computed by using a low-rank approximation data matrix. The table lists the anchor word and the 5 most frequent words in each topic from left to right. We computed the correspondence between topics obtained by ER-SPA and SPA and grouped the topics for each algorithm. Concretely, we measured the 2-norm of each topic vector and computed the correspondence by solving an assignment problem. We can see from the table that the topics obtained by these two algorithms are almost the same from the first to the third panel, and they seem to correspond to "entertainment", "tech" and "business". The topics in the fourth and fifth panels, however, are different. The topic in the fifth panel by ER-SPA seems to correspond to "politics". In contrast, it is difficult to find the topic corresponding to "politics" in the panels by SPA. These show that ER-SPA could extract more recognizable topics than SPA.

Remark 14 *Sparsity plays an important role in computing the SVD for a large document corpus. In general, a document-word matrix arising from a text corpus is quite sparse. Our implementation of Algorithm 2 used the MATLAB command* `svds` *that exploits the sparsity of a matrix in the SVD computation. The implementation could work on all data of 20 Newsgroups corpus, which formed a document-word matrix of size 18,846-by-26,213.*

7. Concluding Remarks

We presented Algorithm 1 for Problem 1 and formally showed that it has correctness and robustness properties. Numerical experiments on synthetic data sets demonstrated that Algorithm 2, which is the practical implementation of Algorithm 1, is robustness to noise. The robustness of the algorithm was measured in terms of the recovery rate. The results indicated that Algorithm 2 can improve the recovery rates of SPA and XRAY. The algorithm was then applied to document clustering. The experimental results implied that it outperformed SPA and extracted more recognizable topics.

We will conclude by suggesting a direction for future research. Algorithm 2 needs to do two computations: one is the SVD of the data matrix and the other is the MVEE for a set of reduced-dimensional data points. It would be ideal to have a single computation that could be parallelized. The MVEE computation requires that the convex hull of data points is full-dimensional. Hence, the SVD computation should be carried out on data points. However, if we could devise an alternative convex set for MVEE, it would possible to avoid SVD computation. It would be interesting to investigate the possibility of algorithms that find near-basis vectors by using the other type of convex set for data points.

Acknowledgments

The author would like to thank Akiko Takeda of the University of Tokyo for her insightful and enthusiastic discussions, and thank the referees for careful reading and helpful suggestions that considerably improved the quality of this paper.

References

S. D. Ahipasaoglu, P. Sun, and M. J. Todd. Linear convergence of a modified Frank-Wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optimization Methods and Software*, 23(1):5–19, 2008.

S. Arora, R. Ge, R. Kannan, and A. Moitra. Computing a nonnegative matrix factorization – Provably. In *Proceedings of the 44th Symposium on Theory of Computing (STOC)*, pages 145–162, 2012a.

S. Arora, R. Ge, and A. Moitra. Learning topic models – Going beyond SVD. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 2012b.

S. Arora, R. Ge, Y. Halpern, D. Mimno, and A. Moitra. A practical algorithm for topic modeling with provable guarantees. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

V. Bittorf, B. Recht, C. Re, and J. A. Tropp. Factoring nonnegative matrices with linear programs. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 1223–1231, 2012.

A. Cichocki, R. Zdunek, A. H. Phan, and S. Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley, 2009.

W. Ding, M. H. Rohban, P. Ishwar, and V. Saligrama. Topic discovery through data dependent and random projections. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems 16 (NIPS)*, pages 1141–1148, 2003.

N. Gillis. Robustness analysis of Hottopixx, a linear programming model for factoring nonnegative matrices. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1189–1212, 2013.

N. Gillis and R. Luce. Robust near-separable nonnegative matrix factorization using linear optimization. arXiv:1302.4385v1, 2013.

N. Gillis and S. A. Vavasis. Fast and robust recursive algorithms for separable nonnegative matrix factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):698–714, 2014.

G. H. Golub and C. F. Van Loan. *Matrix Computation*. The Johns Hopkins University Press, 3rd edition, 1996.

P. Gong and C. Zhang. Efficient nonnegative matrix factorization via projected newton method. *Pattern Recognition*, 45(9):3557–3565, 2012.

D. Greene and P. Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23th International Conference on Machine Learning (ICML)*, 2006.

L. G. Khachiyan. Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research*, 21(2):307–320, 1996.

H. Kim and H. Park. Non-negative matrix factorization based on alternating non-negativity constrained least squares and active set method. *SIAM Journal on Matrix Analysis and Applications*, 30(2):713–730, 2008.

H. Kim and H. Park. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, 33(6):3261–3281, 2011.

J. Kim, Y. He, and H. Park. Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework. *Journal of Global Optimization*, 58(2):285–319, 2014.

A. Kumar, V. Sindhwani, and P. Kambadur. Fast conical hull algorithms for near-separable non-negative matrix factorization. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

P. Kumar and E. A. Yildirim. Minimum-volume enclosing ellipsoids and core sets. *Journal of Optimization Theory and Applications*, 126(1), 2005.

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 (NIPS)*, pages 556–562, 2001.

C.-J. Lin. Projected gradient methods for non-negative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

C. D. Manning, P. Raghavan, and H. Schuetze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

L. Miao and H. Qi. Endmember extraction from highly mixed data using minimum volume constrained nonnegative matrix factorization. *IEEE Transactions on Geoscience and Remote Sensing*, 45(2):765–777, 2007.

J. M. P. Nascimento and J. M. B. Dias. Vertex component analysis: A fast algorithm to unmix hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 43 (4):898–910, 2005.

F. Shahnaz, M. W. Berry, V. P. Pauca, and R. J. Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing and Management*, 42(2):373–386, 2006.

P. Sun and R. M. Freund. Computation of minimum-volume covering ellipsoids. *Operations Research*, 52(5):690–706, 2004.

M. J. Todd and E. A. Yildirim. On Khachiyan's algorithm for the computation of minimum-volume enclosing ellipsoids. *Discrete Applied Mathematics*, 155(13):1731–1744, 2007.

K.-C. Toh. Primal-dual path-following algorithms for determinant maximization problems with linear matrix inequalities. *Computational Optimization and Applications*, 14(3): 309–330, 1999.

K.-C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3 – a MATLAB software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

T. Tsuchiya and Y. Xia. An extension of the standard polynomial-time primal-dual path-following algorithm to the weighted determinant maximization problem with semidefinite constraints. *Pacific Journal of Optimization*, 3(1):165–182, 2007.

L. Vandenberghe, S. Boyd, and S. P. Wu. Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications*, 19(2):499–533, 1998.

S. A. Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal of Optimization*, 20(3):1364–1377, 2009.

W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 267–273, 2003.

Journal of Machine Learning Research 15 (2014) 1041-1071 Submitted 2/13; Revised 11/13; Published 3/14

Improving Prediction from Dirichlet Process Mixtures via Enrichment*†

Sara Wade SARA.WADE@ENG.CAM.AC.UK
Department of Engineering
University of Cambridge
Cambridge, CB2 1PZ, UK

David B. Dunson DUNSON@STAT.DUKE.EDU
Department of Statistical Science
Duke University
Durham, NC 27708-0251, USA

Sonia Petrone SONIA.PETRONE@UNIBOCCONI.IT
Department of Decision Sciences
Bocconi University
Milan, 20136, Italy

Lorenzo Trippa LTRIPPA@JIMMY.HARVARD.EDU
Department of Biostatistics
Harvard University
Boston, MA 02115, USA

Editor: David Blei

Abstract

Flexible covariate-dependent density estimation can be achieved by modelling the joint density of the response and covariates as a Dirichlet process mixture. An appealing aspect of this approach is that computations are relatively easy. In this paper, we examine the predictive performance of these models with an increasing number of covariates. Even for a moderate number of covariates, we find that the likelihood for x tends to dominate the posterior of the latent random partition, degrading the predictive performance of the model. To overcome this, we suggest using a different nonparametric prior, namely an enriched Dirichlet process. Our proposal maintains a simple allocation rule, so that computations remain relatively simple. Advantages are shown through both predictive equations and examples, including an application to diagnosis Alzheimer's disease.

Keywords: Bayesian nonparametrics, density regression, predictive distribution, random partition, urn scheme

∗. For the Alzheimer's Disease Neuroimaging Initiative.

†. Data used in preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (`adni.loni.ucla.edu`). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in analysis or writing of this report. A complete listing of ADNI investigators can be found at: `http://adni.loni.ucla.edu/wp-content/uploads/how_to_apply/ADNI_Acknowledgement_List.pdf`.

1. Introduction

Dirichlet process (DP) mixture models have become popular tools for Bayesian nonparametric regression. In this paper, we examine their behavior in prediction and aim to highlight the difficulties that emerge with increasing dimension of the covariate space. To overcome these difficulties, we suggest a simple extension based on a nonparametric prior developed in Wade et al. (2011) that maintains desirable conjugacy properties of the Dirichlet process and leads to improved prediction. The motivating application is to Alzheimer's disease studies, where the focus is prediction of the disease status based on biomarkers obtained from neuroimages. In this problem, a flexible nonparametric approach is needed to account for the possible nonlinear behavior of the response and complex interaction terms, resulting in improved diagnostic accuracy.

DP mixtures are widely used for Bayesian density estimation, see, for example, Ghosal (2010) and references therein. A common way to extend these methods to nonparametric regression and conditional density estimation is by modelling the joint distribution of the response and the covariates (X, Y) as a mixture of multivariate Gaussians (or more general kernels). The regression function and conditional density estimates are indirectly obtained from inference on the joint density, an idea which is similarly employed in classical kernel regression (Scott, 1992, Chapter 8).

This approach, which we call the joint approach, was first introduced by Müller et al. (1996), and subsequently studied by many others including Kang and Ghosal (2009); Shahbaba and Neal (2009); Hannah et al. (2011); Park and Dunson (2010); and Müller and Quintana (2010). The DP model uses simple local linear regression models as building blocks and partitions the observed subjects into clusters, where within clusters, the linear regression model provides a good fit. Even though within clusters the model is parametric, globally, a wide range of complex distributions can describe the joint distribution, leading to a flexible model for both the regression function and the conditional density.

Another related class of models is based on what we term the conditional approach. In such models, the conditional density of Y given x, $f(y|x)$, is modelled directly, as a convolution of a parametric family $f(y|x, \theta)$ with an unknown mixing distribution $\mathbf{P}_x$ for θ. A prior is then given on the family of distributions $\{\mathbf{P}_x, x \in \mathcal{X}\}$ such that the $\mathbf{P}_x$'s are dependent. Examples for the law of $\{\mathbf{P}_x, x \in \mathcal{X}\}$ start from the dependent DPs of MacEachern (1999); Gelfand et al. (2005) and include Griffin and Steel (2006); Dunson and Park (2008); Ren et al. (2011); Chung and Dunson (2009); and Rodriguez and Dunson (2011), just to name a few. Such conditional models can approximate a wide range of response distributions that may change flexibly with the covariates. However, computations are often quite burdensome. One of the reasons the model examined here is so powerful is its simplicity. Together, the joint approach and the clustering of the DP provide a built-in technique to allow for changes in the response distribution across the covariate space, yet it is simple and generally less computationally intensive than the nonparametric conditional models based on dependent DPs.

The random allocation of subjects into groups in joint DP mixture models is driven by the need to obtain a good approximation of the joint distribution of X and Y. This means that subjects with similar covariates and similar relationship between the response and covariates will tend to cluster together. However, difficulties emerge as p, the dimension of

the covariate space, increases. As we will detail, even for moderately large p the likelihood of x tends to dominate the posterior of the random partition, so that clusters are based mainly on similarity in the covariate space. This behaviour is quite unappealing if the marginal density of X is complex, as is typical in high dimensions, because it causes the posterior to concentrate on partitions with many small clusters, as many kernels are needed to describe $f(x)$. This occurs even if the conditional density of Y given x is more well behaved, meaning, a few kernels suffice for its approximation. Typical results include poor estimates of the regression function and conditional density with unnecessarily wide credible intervals due to small clusters and, consequently, poor prediction.

This inefficient performance may not disappear with increasing samples. On one hand, appealing to recent theoretical results (Wu and Ghosal, 2008, 2010; Tokdar, 2011), one could expect that as the sample size increases, the posterior on the unknown density $f(x, y)$ induced by the DP joint mixture model is consistent at the true density. In turn, posterior consistency of the joint is likely to have positive implications for the behavior of the random conditional density and regression function; see Rodriguez et al. (2009); Hannah et al. (2011); and Norets and Pelenis (2012) for some developments in this direction. However, the unappealing behaviour of the random partition that we described above could be reflected in worse convergence rates. Indeed, recent results by Efromovich (2007) suggest that if the conditional density is smoother than the joint, it can be estimated at a faster rate. Thus, improving inference on the random partition to take into account the different degree of smoothness of $f(x)$ and $f(y|x)$ appears to be a crucial issue.

Our goal in this paper is to show that a simple modification of the nonparametric prior on the mixing distribution, that better models the random partition, can more efficiently convey the information present in the sample, leading to more efficient conditional density estimates in term of smaller errors and less variability, for finite samples. To achieve this aim, we consider a prior that allows *local* clustering, that is, the clustering structure for the marginal of X and the regression of Y on x may be different. We achieve this by replacing the DP with the enriched Dirichlet process (EDP) developed in Wade et al. (2011). Like the DP, the EDP is a conjugate nonparametric prior, but it allows a nested clustering structure that can overcome the above issues and lead to improved predictions. An alternative proposal is outlined in Petrone and Trippa (2009) and in unpublished work by Dunson et al. (2011). However, the EDP offers a richer parametrization. In a Bayesian nonparametric framework, several extensions of the DP have been proposed to allow local clustering (see, e.g., Dunson et al. 2008; Dunson 2009; Petrone et al. 2009). However, the greater flexibility is often achieved at the price of more complex computations. Instead, our proposal maintains an analytically computable allocation rule, and therefore, computations are a straightforward extension of those used for the joint DP mixture model. Thus, our main contributions are to highlight the problematic behavior in prediction of the joint DP mixture model for increasing p and also offer a simple solution based on the EDP that maintains computational ease. In addition, we give results on random nested partitions that are implied by the proposed prior.

This paper is organized as follows. In Section 2, we review the joint DP mixture model, discuss the behavior of the random partition, and examine the predictive performance. In Section 3, we propose a joint EDP mixture model, derive its random partition model, and emphasize the predictive improvements of the model. Section 4 covers computational

procedures. We provide a simulated example in Section 5 to demonstrate how the EDP model can lead to more efficient estimators by making better use of information contained in the sample. Finally, in Section 6, we apply the model to predict Alzheimer's disease status based on measurements of various brain structures.

2. Joint DP Mixture Model

A joint DP mixture model for multivariate density estimation and nonparametric regression assumes that

$$(X_i, Y_i)|P \overset{iid}{\sim} f(x, y|P) = \int K(x, y|\xi) dP(\xi),$$

where X_i is p-dimensional, Y is usually univariate, and the mixing distribution P is given a DP prior with scale parameter $\alpha > 0$ and base measure P_0, denoted by $\mathbf{P} \sim \mathrm{DP}(\alpha P_0)$. Due to the a.s. discrete nature of the DP, the model reduces to a countable mixture

$$f(x, y|P) = \sum_{j=1}^{\infty} w_j K(x, y|\tilde{\xi}_j),$$

where the mixing weights w_j have a stick breaking prior with parameter α and $\tilde{\xi}_j \overset{iid}{\sim} P_0$, independently of the w_j. This model was first developed for Bayesian nonparametric regression by Müller et al. (1996), who assume multivariate Gaussian kernels $\mathrm{N}_{p+1}(\mu, \Sigma)$ with $\xi = (\mu, \Sigma)$ and use a conjugate normal inverse Wishart prior as the base measure of the DP. However, for even moderately large p, this approach is practically unfeasible. Indeed, the computational cost of dealing with the full $p+1$ by $p+1$ covariance matrix greatly increases with large p. Furthermore, the conjugate inverse Wishart prior is known to be too poorly parametrized; in particular, there is a single parameter ν to control variability, regardless of p (see Consonni and Veronese, 2001).

A more effective formulation of this model has been recently proposed by Shahbaba and Neal (2009), based on two simple modifications. First, the joint kernel is decomposed as the product of the marginal of X and the conditional of $Y|x$, and the parameter space consequently expressed in terms of the parameters ψ of the marginal and the parameters θ of the conditional. This is a classic reparametrization which, in the Gaussian case, is the basis of generalizations of the inverse Wishart conjugate prior; see Brown et al. (1994). Secondly, they suggest using simple kernels, assuming local independence among the covariates, that is, the covariance matrix of the kernel for X is diagonal. These two simple modifications allow several important improvements. Computationally, reducing the covariance matrix to p variances can greatly ease calculations. Regarding flexibility of the base measure, the conjugate prior now includes a separate parameter to control variability for each of the p variances. Furthermore, the model still allows for local correlations between Y and X through the conditional kernel of $Y|x$ and the parameter θ. In addition, the factorization in terms of the marginal and conditional and the assumption of local independence of the covariates allow for easy inclusion of discrete or other types of response or covariates. Note that even though, within each component, we assume independence of the covariates, globally, there may be dependence.

This extended model, in full generality, can be described through latent parameter vectors as follows:

$$Y_i|x_i,\theta_i \overset{ind}{\sim} F_y(\cdot|x_i,\theta_i), \quad X_i|\psi_i \overset{ind}{\sim} F_x(\cdot|\psi_i), \tag{1}$$
$$(\theta_i,\psi_i)|P \overset{iid}{\sim} P, \quad \mathbf{P} \sim \mathrm{DP}(\alpha P_{0\theta} \times P_{0\psi}).$$

Here the base measure $P_{0\theta} \times P_{0\psi}$ of the DP assumes independence between the θ and the ψ parameters, as this is the structurally conjugate prior for (θ,ψ), and thus, results in simplified computations, but the model could be extended to more general choices. We further assume that $P_{0\theta}$ and $P_{0\psi}$ are absolutely continuous, with densities $p_{0\theta}$ and $p_{0\psi}$. Since $\mathbf{P}$ is discrete a.s., integrating out the subject-specific parameters (θ_i,ψ_i), the model for the joint density is

$$f(y_i,x_i|P) = \sum_{j=1}^{\infty} w_j K(y_i|x_i,\tilde{\theta}_j)K(x_i|\tilde{\psi}_j), \tag{2}$$

where the kernels $K(y|x,\theta)$ and $K(x|\psi)$ are the densities associated to $F_y(\cdot|x,\theta)$ and $F_x(\cdot|\psi)$. In the Gaussian case, $K(x|\psi)$ is the product of $\mathrm{N}(\mu_{x,h},\sigma^2_{x,h})$ Gaussians, $h=1,\ldots,p$, and $K(y|x,\theta)$ is $\mathrm{N}(\underline{x}\beta,\sigma^2_{y|x})$, where $\underline{x}=(1,x')$. Shahbaba and Neal (2009) focus on the case when Y is categorical and the local model for $Y|x$ is a multinomial logit. Hannah et al. (2011) extend the model to the case when, locally, the conditional distribution of $Y|x$ belongs to the class of generalized linear models (GLM), that is, the distribution of the response belongs to the exponential family and the mean of the response can be expressed a function of a linear combination of the covariates.

Model (2) allows for flexible conditional densities

$$f(y|x,P) \;=\; \frac{\sum_{j=1}^{\infty} w_j K(x|\tilde{\psi}_j)K(y|x,\tilde{\theta}_j)}{\sum_{j'=1}^{\infty} w_{j'} K(x|\tilde{\psi}_{j'})} \equiv \sum_j w_j(x)K(y|x,\tilde{\theta}_j),$$

and nonlinear regression

$$\mathrm{E}[Y|x,P] = \sum_{j=1}^{\infty} w_j(x)\,\mathrm{E}[Y|x,\tilde{\theta}_j],$$

with $\mathrm{E}[Y|x,\tilde{\theta}_j] = \underline{x}\beta^*_j$ for Gaussian kernels. Thus, the model provides flexible kernel based density and regression estimation, and MCMC computations are standard. However, the DP only allows joint clusters of the parameters (θ_i,ψ_i), $i=1,\ldots,n$. We underline drawbacks of such joint clustering in the following subsections.

2.1 Random Partition and Inference

One of the crucial features of DP mixture models is the dimension reduction and clustering obtained due to the almost sure discreteness of $\mathbf{P}$. In fact, this implies that a sample (θ_i,ψ_i), $i=1,\ldots,n$ from a DP presents ties with positive probability and can be conveniently described in terms of the random partition and the distinct values. Using a standard notation, we denote the random partition by a vector of cluster allocation labels $\rho_n =$

$(s_1, \ldots, s_n)$, with $s_i = j$ if (θ_i, ψ_i) is equal to the j^{th} unique value observed, (θ_j^*, ψ_j^*), for $j = 1, \ldots, k$, where $k = k(\rho_n)$ is the number of groups in the partition ρ_n. Additionally, we will denote by $S_j = \{i : s_i = j\}$ the set of subject indices in the j^{th} cluster and use the notation $y_j^* = \{y_i : i \in S_j\}$ and $x_j^* = \{x_i : i \in S_j\}$. We also make use of the short notation $x_{1:n} = (x_1, \ldots, x_n)$.

Often, the random probability measure $\mathbf{P}$ is integrated out and inference is based on the posterior of the random partition ρ_n and the cluster-specific parameters $(\theta^*, \psi^*) \equiv (\theta_j^*, \psi_j^*)_{j=1}^k$;

$$p(\rho_n, \theta^*, \psi^* | x_{1:n}, y_{1:n}) \propto p(\rho_n) \prod_{j=1}^{k} p_{0\theta}(\theta_j^*) p_{0\psi}(\psi_j^*) \prod_{j=1}^{k} \prod_{i \in S_j} K(x_i|\psi_j^*) K(y_i|x_i, \theta_j^*).$$

As is well known (Antoniak, 1974), the prior induced by the DP on the random partition is $p(\rho_n) \propto \alpha^k \prod_{j=1}^k \Gamma(n_j)$, where n_j is the size of S_j. Marginalizing out the θ^*, ψ^*, the posterior of the random partition is

$$p(\rho_n | x_{1:n}, y_{1:n}) \propto \alpha^k \prod_{j=1}^{k} \Gamma(n_j)\, h_x(x_j^*)\, h_y(y_j^* | x_j^*). \tag{3}$$

Notice that the marginal likelihood component in (3), say $h(x_{1:n}, y_{1:n}|\rho_n)$, is factorized as the product of the cluster-specific marginal likelihoods $h_x(x_j^*) h_y(y_j^*|x_j^*)$ where $h_x(x_j^*) = \int_\Psi \prod_{i \in S_j} K(x_i|\psi) dP_{0\psi}(\psi)$ and $h_y(y_j^*|x_j^*) = \int_\Theta \prod_{i \in S_j} K(y_i|x_i, \theta) dP_{0\theta}(\theta)$.

Given the partition and the data, the conditional distribution of the distinct values is simple, as they are independent across clusters, with posterior densities

$$\begin{aligned} p(\theta_j^* | \rho_n, x_{1:n}, y_{1:n}) &\propto p_{0\theta}(\theta_j^*) \prod_{i \in S_j} K(y_i|x_i, \theta_j^*), \\ p(\psi_j^* | \rho_n, x_{1:n}, y_{1:n}) &\propto p_{0\psi}(\psi_j^*) \prod_{i \in S_j} K(x_i|\psi_j^*). \end{aligned} \tag{4}$$

Thus, given ρ_n, the cluster-specific parameters (θ_j^*, ψ_j^*) are updated based only on the observations in cluster S_j. Computations are simple if $p_{0\theta}$ and $p_{0\psi}$ are conjugate priors.

The above expressions show the crucial role of the random partition. From Equation (3), we have that given the data, subjects are clustered in groups with similar behaviour in the covariate space and similar relationship with the response. However, even for moderate p the likelihood for x tends to dominate the posterior of the random partition, so that clusters are determined only by similarity in the covariate space. This is particularly evident when the covariates are assumed to be independent locally, that is, $K(x_i|\psi_j^*) = \prod_{h=1}^p K(x_{i,h}|\psi_{j,h}^*)$. Clearly, for large p, the scale and magnitude of changes in $\prod_{h=1}^p K(x_{i,h}|\psi_{j,h}^*)$ will wash out any information given in the univariate likelihood $K(y_i|\theta_j^*, x_i)$. Indeed, this is just the behavior we have observed in practice in running simulations for large p (results not shown).

For a simple example demonstrating how the number of components needed to approximate the marginal of X can blow up with p, imagine X is uniformly distributed on a cuboid of side length $r > 1$. Consider approximating

$$f_0(x) = \frac{1}{r^p} \mathbf{1}(x \in [0, r]^p) \quad \text{by} \quad f_k(x) = \sum_{j=1}^{k} w_j \mathrm{N}_p(x|\mu_j, \sigma_j^2 I_p).$$

Since the true distribution of x is uniform on the cube $[0,r]^p$, to obtain a good approximation, the weighted components must place most of their mass on values of x contained in the cuboid. Let $B_\sigma(\mu)$ denote a ball of radius σ centered at μ. If a random vector V is normally distributed with mean μ and variance $\sigma^2 I_p$, then for $0<\epsilon<1$,

$$P(V \in B_{\sigma z(\epsilon)}(\mu)) = 1-\epsilon, \quad \text{where} \quad z(\epsilon)^2 = (\chi^2_p)^{-1}(1-\epsilon),$$

that is, the square of $z(\epsilon)$ is the $(1-\epsilon)$ quantile of the chi-squared distribution with p degrees of freedom. For small ϵ, this means that the density of V places most of its mass on values contained in a ball of radius $\sigma z(\epsilon)$ centered at μ. For $\epsilon>0$, define

$$\tilde{f}_k(x) = \sum_{j=1}^{k} w_j \mathrm{N}(x;\mu_j,\sigma_j^2 I_p) * \mathbf{1}(x \in B_{\sigma_j z(\epsilon_j)}(\mu_j)),$$

where $\epsilon_j = \epsilon/(kw_j)$. Then, $\tilde{f}_k$ is close to f_k (in the L_1 sense):

$$\int_{\mathbb{R}^p} |f_k(x) - \tilde{f}_k(x)|dx = \int_{\mathbb{R}^p} \sum_{j=1}^{k} w_j \mathrm{N}(x;\mu_j,\sigma_j^2 I_p) * \mathbf{1}(x \in B^c_{\sigma_j z(\epsilon_j)}(\mu_j))dx = \epsilon.$$

For $\tilde{f}_k$ to be close to f_0, the parameters μ_j, σ_j, w_j need to be chosen so that the balls $B_{\sigma_j z(\epsilon/(kw_j))}(\mu_j)$ are contained in the cuboid. That means that centers of the balls are contained in the cuboid,

$$\mu_j \in [0,r]^p, \tag{5}$$

with further constraints on σ_j^2 and w_j, so that the radius is small enough. In particular,

$$\sigma_j z\left(\frac{\epsilon}{kw_j}\right) \leq \min(\mu_1, r-\mu_1, \ldots, \mu_p, r-\mu_p) \leq \frac{r}{2}. \tag{6}$$

However, as p increases the volume of the cuboid goes to infinity, but the volume of any ball $B_{\sigma_j z(\epsilon/(kw_j))}(\mu_j)$ defined by (5) and (6) goes to 0 (see Clarke et al., 2009, Section 1.1). Thus, just to reasonably cover the cuboid with the balls of interest, the number of components will increase dramatically, and more so, when we consider the approximation error of the density estimate. Now, as an extreme example, imagine that $f_0(y|x)$ is a linear regression model. Even though one component is sufficient for $f_0(y|x)$, a large number of components will be required to approximate $f_0(x)$, especially as p increases.

It appears evident from (4) this behavior of the random partition also negatively affects inference on the cluster-specific parameters. In particular, when many kernels are required to approximate the density of X with few observations within each cluster, the posterior for θ_j^* may be based on a sample of unnecessarily small size, leading to a flat posterior with an unreliable posterior mean and large influence of the prior.

2.2 Covariate-dependent Urn Scheme and Prediction

Difficulties associated to the behavior of the random partition also deteriorate the predictive performance of the model. Prediction in DP joint mixture models is based on a covariate-dependent urn scheme (Park and Dunson, 2010; Müller and Quintana, 2010), such that

conditionally on the partition ρ_n and $x_{1:n+1}$, the cluster allocation s_{n+1} of a new subject with covariate value x_{n+1} is determined as

$$s_{n+1}|\rho_n, x_{1:n+1} \sim \frac{\omega_{k+1}(x_{n+1})}{c_0}\,\delta_{k+1} + \sum_{j=1}^{k} \frac{\omega_j(x_{n+1})}{c_0}\delta_j, \tag{7}$$

where

$$\omega_{k+1}(x_{n+1}) = \frac{\alpha\, h_{0,x}(x_{n+1})}{\alpha+n},$$
$$\omega_j(x_{n+1}) = \frac{n_j \int K(x_{n+1}|\psi)p(\psi|x_j^*)d\psi}{\alpha+n} \equiv \frac{n_j\, h_{j,x}(x_{n+1})}{\alpha+n},$$

and $c_0 = p(x_{n+1}|\rho_n, x_{1:n})$ is the normalizing constant. This urn scheme is a generalization of the classic Pólya urn scheme that allows the cluster allocation probability to depend on the covariates; the probability of allocation to cluster j depends on the similarity of x_{n+1} to the x_i in cluster j as measured by the predictive density $h_{j,x}$. See Park and Dunson (2010) for more details.

From the urn scheme (7) one obtains the structure of the prediction. The predictive density at y for a new subject with a covariate of x_{n+1} is computed as

$$\begin{aligned}
&f(y|y_{1:n}, x_{1:n+1}) \\
&= \sum_{\rho_n}\sum_{s_{n+1}} f(y|y_{1:n}, x_{1:n+1}, \rho_n, s_{n+1})p(s_{n+1}|y_{1:n}, x_{1:n+1}, \rho_n)\, p(\rho_n|y_{1:n}, x_{1:n+1}) \\
&= \sum_{\rho_n}\left(\frac{\omega_{k+1}(x_{n+1})}{c_0}\, h_{0,y}(y|x_{n+1}) + \sum_{j=1}^{k}\frac{\omega_j(x_{n+1})}{c_0}\, h_{j,y}(y|x_{n+1})\right)\frac{c_0 p(\rho_n|x_{1:n}, y_{1:n})}{p(x_{n+1}|x_{1:n}, y_{1:n})} \\
&= \sum_{\rho_n}\left(\frac{\omega_{k+1}(x_{n+1})}{c}\, h_{0,y}(y|x_{n+1}) + \sum_{j=1}^{k}\frac{\omega_j(x_{n+1})}{c}\, h_{j,y}(y|x_{n+1})\right) p(\rho_n|x_{1:n}, y_{1:n}),
\end{aligned} \tag{8}$$

where $c = p(x_{n+1}|x_{1:n}, y_{1:n})$. Thus, given the partition, the conditional predictive density is a weighted average of the prior guess $h_{0,y}(y|x) \equiv \int K(y|x,\theta)dP_{0\theta}(\theta)$ and the cluster-specific predictive densities of y at x_{n+1},

$$h_{j,y}(y|x_{n+1}) = \int K(y|x_{n+1},\theta)p(\theta|x_j^*, y_j^*)d\theta,$$

with covariate-dependent weights. The predictive density is obtained by averaging with respect to the posterior of ρ_n. However, for moderate to large p, the posterior of the random partition suffers the drawbacks discussed in the previous subsection. In particular, too many small x-clusters lead to unreliable within cluster predictions based on small sample sizes. Furthermore, the measure which determines similarity of x_{n+1} and the j^{th} cluster will be too rigid. Consequently, the resulting overall prediction may be quite poor.

These drawbacks will also affect the point prediction, which, under quadratic loss, is

$$\mathrm{E}[Y_{n+1}|y_{1:n}, x_{1:n+1}]$$
$$= \sum_{\rho_n} \left(\frac{\omega_{k+1}(x_{n+1})}{c} \mathrm{E}_0[Y_{n+1}|x_{n+1}] + \sum_{j=1}^{k} \frac{\omega_j(x_{n+1})}{c} \mathrm{E}_j[Y_{n+1}|x_{n+1}] \right) p(\rho_n|x_{1:n}, y_{1:n}), \quad (9)$$

where $\mathrm{E}_0[Y_{n+1}|x_{n+1}]$ is the expectation of Y_{n+1} with respect to $h_{0,y}$ and $\mathrm{E}_j[Y_{n+1}|x_{n+1}] \equiv \mathrm{E}[\mathrm{E}[Y_{n+1}|x_{n+1}, \theta_j^*]|x_j^*, y_j^*]$ is the expectation of Y_{n+1} with respect to $h_{j,y}$.
Example. When $K(y|x, \theta) = \mathrm{N}(y; \underline{x}\beta, \sigma^2)$ and the prior for (β, σ^2) is the multivariate normal inverse gamma with parameters $(\beta_0, C^{-1}, a_y, b_y)$, (9) is

$$\sum_{\rho_n} \left(\frac{\omega_{k+1}(x_{n+1})}{c} \underline{x}_{n+1}\beta_0 + \sum_{j=1}^{k} \frac{\omega_j(x_{n+1})}{c} \underline{x}_{n+1}\hat{\beta}_j \right) p(\rho_n|x_{1:n}, y_{1:n}),$$

where $\hat{\beta}_j = \hat{C}_j^{-1}(C\beta_0 + \underline{X}_j' y_j^*)$, $\hat{C}_j = C + \underline{X}_j'\underline{X}_j$, $\underline{X}_j$ is a n_j by $p+1$ matrix with rows $\underline{x}_i$ for $i \in S_j$, and (8) is

$$\frac{\omega_{k+1}(x_{n+1})}{c} \mathcal{T}\left(y|\underline{x}_{n+1}\beta_0, \frac{b_y}{a_y} W^{-1}, 2a_y\right) + \sum_{j=1}^{k} \frac{\omega_j(x_{n+1})}{c} \mathcal{T}\left(y|\underline{x}_{n+1}\hat{\beta}_j, \frac{\hat{b}_{y,j}}{\hat{a}_{y,j}} W_j^{-1}, 2\hat{a}_{y,j}\right),$$

where $\mathcal{T}(\cdot; \mu, \sigma^2, \nu)$ denotes the density of a random variable V such that $(V-\mu)/\sigma$ has a t-distribution with ν degrees of freedom; $\hat{a}_{y,j} = a_y + n_j/2$;

$$\hat{b}_{y,j} = b_y + \frac{1}{2}(y_j^* - \underline{X}_j\beta_0)'(I_{n_j} - \underline{X}_j\hat{C}_j^{-1}\underline{X}_j')(y_j^* - \underline{X}_j\beta_0);$$

$$W = 1 - \underline{x}_{n+1}(C + \underline{x}_{n+1}'\underline{x}_{n+1})^{-1}\underline{x}_{n+1}';$$

and W_j is defined as W with C replaced by $\hat{C}_j$.

3. Joint EDP Mixture Model

As seen, the global clustering of the DP prior on $\mathcal{P}(\Theta \times \Psi)$, the space of probability measures on $\Theta \times \Psi$, does not allow one to efficiently model the types of data discussed in the previous section. Instead, it is desirable to use a nonparametric prior that allows many ψ-clusters, to fit the complex marginal of X, and fewer θ-clusters. At the same time, we want to preserve the desirable conjugacy properties of the DP, in order to maintain fairly simple computations. To these aims, our proposal is to replace the DP with the more richly parametrized *enriched Dirichlet process* (Wade et al., 2011). The EDP is conjugate and has an analytically computable urn scheme, but it gives a nested partition structure that can model the desired clustering behavior.

Recall that the model (1) was obtained by decomposing the joint kernel as the product of the marginal and conditional kernels. The EDP is a natural alternative for the mixing distribution of this model, as it is similarly based on the idea of expressing the unknown

random joint probability measure $\mathbf{P}$ of (θ, ψ) in terms of the random marginal and conditionals. This requires the choice of an ordering of θ and ψ, and this choice is problem specific. In the situation described here, it is natural to consider the random marginal distribution $\mathbf{P}_\theta$ and the random conditional $\mathbf{P}_{\psi|\theta}$, to obtain the desired clustering structure. Then, the EDP prior is defined by

$$\begin{aligned} \mathbf{P}_\theta &\sim \mathrm{DP}(\alpha_\theta P_{0\theta}), \\ \mathbf{P}_{\psi|\theta}(\cdot|\theta) &\sim \mathrm{DP}(\alpha_\psi(\theta) P_{0\psi|\theta}(\cdot|\theta)), \quad \forall \theta \in \Theta, \end{aligned}$$

and $\mathbf{P}_{\psi|\theta}(\cdot|\theta)$ for $\theta \in \Theta$ are independent among themselves and from $\mathbf{P}_\theta$. Together these assumptions induce a prior for the random joint $\mathbf{P}$ through the joint law of the marginal and conditionals and the mapping $(P_\theta, P_{\psi|\theta}) \to \int P_{\psi|\theta}(\cdot|\theta) dP_\theta(\theta)$. The prior is parametrized by the base measure P_0, expressed as

$$P_0(A \times B) = \int_A P_{0\psi|\theta}(B|\theta) dP_{0\theta}(\theta)$$

for all Borel sets A and B, and by a precision parameter α_θ associated to θ and a collection of precision parameters $\alpha_\psi(\theta)$ for every $\theta \in \Theta$ associated to $\psi|\theta$. Note the contrast with the DP, which only allows one precision parameter to regulate the uncertainty around P_0.

The proposed EDP mixture model for regression is as in (1), but with

$$\mathbf{P} \sim \mathrm{EDP}(\alpha_\theta, \alpha_\psi(\theta), P_0)$$

in place of $\mathbf{P} \sim \mathrm{DP}(\alpha P_{0\theta} \times P_{0\psi})$. In general, P_0 is such that θ and ψ are dependent, but here we assume the same structurally conjugate base measure as for the DP model (1), so $P_0 = P_{0\theta} \times P_{0\psi}$. Using the square breaking representation of the EDP (Wade et al., 2011, Proposition 4) and integrating out the (θ_i, ψ_i) parameters, the model for the joint density is

$$f(x, y|P) = \sum_{j=1}^{\infty} \sum_{l=1}^{\infty} w_j w_{l|j} K(x|\tilde{\psi}_{l|j}) K(y|x, \tilde{\theta}_j).$$

This gives a mixture model for the conditional densities with more flexible weights

$$f(y|x, P) = \sum_{j=1}^{\infty} \frac{\sum_{l=1}^{\infty} w_{l|j} K(x|\tilde{\psi}_{l|j})}{\sum_{j'=1}^{\infty} w_{j'} \sum_{l'=1}^{\infty} w_{l'|j'} K(x|\tilde{\psi}_{l'|j'})} K(y|x, \tilde{\theta}_{j'}) \equiv \sum_{j=1}^{\infty} \tilde{w}_j(x) K(y|x, \tilde{\theta}_j).$$

3.1 Random Partition and Inference

The advantage of the EDP is the implied nested clustering. The EDP partitions subjects in θ-clusters and ψ-subclusters within each θ-cluster, allowing the use of more kernels to describe the marginal of X for each kernel used for the conditional of $Y|x$. The random partition model induced from the EDP can be described as a nested Chinese Restaurant Process (nCRP).

First, customers choose restaurants according to the CRP induced by $\mathbf{P}_\theta \sim \mathrm{DP}(\alpha_\theta P_{0\theta})$, that is, with probability proportional to the number n_j of customers eating at restaurant j,

the $(n+1)^{\text{th}}$ customer eats at restaurant j, and with probability proportional to α_θ, she eats at a new restaurant. Restaurant are then colored with colors $\theta_j^* \overset{iid}{\sim} P_{0\theta}$. Within restaurant j, customers sit at tables as in the CRP induced by the $\mathbf{P}_{\psi|\theta_j^*} \sim \text{DP}(\alpha_\psi(\theta_j^*)P_{0\psi|\theta}(\cdot|\theta_j^*))$. Tables in restaurant j are then colored with colors $\psi_{l|j}^* \overset{iid}{\sim} P_{0\psi|\theta}(\cdot|\theta_j^*)$.

This differs from the nCRP proposed by Blei et al. (2010), which had the alternative aim of learning topic hierarchies by clustering parameters (topics) hierarchically along a tree of infinite CRPs. In particular, each subject follows a path down the tree according to a sequence of nested CRPs and the parameters of subject i are associated with the cluster visited at a latent subject-specific level l of this path. Although related, the EDP is not a special case with the tree depth fixed to 2; the EDP defines a prior on a multivariate random probability measure on $\Theta \times \Psi$ and induces a nested partition of the multivariate parameter (θ, ψ), where the first level of the tree corresponds to the clustering of the θ parameters and the second corresponds to the clustering of the ψ parameters. A generalization of the EDP to a depth of $D \leq \infty$ is related to Blei et al.'s nCRP with depth $D \leq \infty$, but only if one regards the parameters of each subject as the vector of parameters $(\xi_1, \ldots, \xi_D)$ associated to each level of the tree. Furthermore, this generalization of the EDP would allow a more flexible specification of the mass parameters and possible correlation among the nested parameters.

The nested partition of the EDP is described by $\rho_n = (\rho_{n,y}, \rho_{n,x})$, where $\rho_{n,y} = (s_{y,1}, ..., s_{y,n})$ and $\rho_{n,x} = (s_{x,1}, ..., s_{x,n})$ with $s_{y,i} = j$ if $\theta_i = \theta_j^*$, the j^{th} distinct θ-value in order of appearance, and $s_{x,i} = l$ if $\psi_i = \psi_{l|j}^*$, the l^{th} color that appeared inside the j^{th} θ-cluster. Additionally, we use the notation $S_{j+} = \{i : s_{y,i} = j\}$, with size n_j, $j = 1, \ldots, k$, and $S_{l|j} = \{i : s_{y,i} = j, s_{x,i} = l\}$, with size $n_{l|j}$, $l = 1, \ldots, k_j$. The unique parameters will be denoted by $\theta^* = (\theta_j^*)_{j=1}^k$ and $\psi^* = (\psi_{1|j}^*, \ldots, \psi_{k_j|j}^*)_{j=1}^k$. Furthermore, we use the notation $\rho_{n_j,x} = (s_{x,i} : i \in S_{j+})$ and $y_j^* = \{y_i : i \in S_{j+}\}$, $x_j^* = \{x_i : i \in S_{j+}\}$, $x_{l|j}^* = \{x_i : i \in S_{l|j}\}$.

Proposition 1 *The probability law of the nested random partition defined from the EDP is*

$$p(\rho_n) = \frac{\Gamma(\alpha_\theta)}{\Gamma(\alpha_\theta + n)} \alpha_\theta^k \prod_{j=1}^k \int_\Theta \alpha_\psi(\theta)^{k_j} \frac{\Gamma(\alpha_\psi(\theta))\Gamma(n_j)}{\Gamma(\alpha_\psi(\theta) + n_j)} dP_{0\theta}(\theta) \prod_{l=1}^{k_j} \Gamma(n_{l|j}).$$

Proof From independence of random conditional distributions among $\theta \in \Theta$,

$$p(\rho_n, \theta^*) = p(\rho_{n,y}) \prod_{j=1}^k p_{0\theta}(\theta_j^*) p(\rho_{n,x}|\rho_{n,y}, \theta^*) = p(\rho_{n,y}) \prod_{j=1}^k p_{0\theta}(\theta_j^*) p(\rho_{n_j,x}|\theta_j^*).$$

Next, using the results of the random partition model of the DP (Antoniak, 1974), we have

$$p(\rho_n, \theta^*) = \frac{\Gamma(\alpha_\theta)}{\Gamma(\alpha_\theta + n)} \alpha_\theta^k \prod_{j=1}^k p_{0\theta}(\theta_j^*) \alpha_\psi(\theta_j^*)^{k_j} \frac{\Gamma(\alpha_\psi(\theta_j^*))\Gamma(n_j)}{\Gamma(\alpha_\psi(\theta_j^*) + n_j)} \prod_{l=1}^{k_j} \Gamma(n_{l|j}).$$

Integrating out θ^* leads to the result. ■

From Proposition 1, we gain an understanding of the types of partitions preferred by the

EDP and the effect of the parameters. If for all θ, $\alpha_\psi(\theta) = \alpha_\theta P_{0\theta}(\{\theta\})$, that is $\alpha_\psi(\theta) = 0$ if $P_{0\theta}$ is non-atomic, we are back to the DP random partition model, see Proposition 2 of Wade et al. (2011). In the case when $P_{0\theta}$ is non-atomic, this means that the conditional $\mathbf{P}_{\psi|\theta}$ is degenerate at some random location with probability one (for each restaurant—one table).

In general, $\alpha_\psi(\theta)$ may be a flexible function of θ, reflecting the fact that within some θ-clusters more kernels may be required for good approximation of the marginal of X. In practice, a common situation that we observe is a high value of $\alpha_\psi(\theta)$ for average values of θ and lower values of $\alpha_\psi(\theta)$ for more extreme θ values, capturing homogeneous outlying groups. In this case, a small value of α_θ will encourage few θ-clusters, and, given θ^*, a large $\alpha_\psi(\theta_j^*)$ will encourage more ψ-clusters within the j^{th} θ-cluster. The term $\prod_{j=1}^k \prod_{l=1}^{k_j} \Gamma(n_{l|j})$ will encourage asymmetrical (θ, ψ)-clusters, preferring one large cluster and several small clusters, while, given θ^*, the term involving the product of beta functions contains parts that both encourage and discourage asymmetrical θ-clusters. In the special case when $\alpha_\psi(\theta) = \alpha_\psi$ for all $\theta \in \Theta$, the random partition model simplifies to

$$p(\rho_n) = \frac{\Gamma(\alpha_\theta)}{\Gamma(\alpha_\theta + n)} \alpha_\theta^k \prod_{j=1}^{k} \alpha_\psi^{k_j} \frac{\Gamma(\alpha_\psi)\Gamma(n_j)}{\Gamma(\alpha_\psi + n_j)} \prod_{l=1}^{k_j} \Gamma(n_{l|j}).$$

In this case, the tendency of the term involving the product of beta functions is to slightly prefer asymmetrical θ-clusters with large values of α_ψ boosting this preference.

As discussed in the previous section, the random partition plays a crucial role, as its posterior distribution affects both inference on the cluster-specific parameters and prediction. For the EDP, it is given by the following proposition.

Proposition 2 *The posterior of the random partition of the EDP model is*

$$p(\rho_n|\ x_{1:n}, y_{1:n}) \propto \alpha_\theta^k \prod_{j=1}^{k} \int_\Theta \frac{\Gamma(\alpha_\psi(\theta))\Gamma(n_j)}{\Gamma(\alpha_\psi(\theta) + n_j)} \alpha_\psi(\theta)^{k_j} dP_{0\theta}(\theta) h_y(y_j^*|x_j^*) \prod_{l=1}^{k_j} \Gamma(n_{l|j}) h_x(x_{l|j}^*).$$

The proof relies on a simple application of Bayes theorem. In the case of constant $\alpha_\psi(\theta)$, the expression for the posterior of ρ_n simplifies to

$$p(\rho_n|\ x_{1:n}, y_{1:n}) \propto \alpha_\theta^k \prod_{j=1}^{k} \frac{\Gamma(\alpha_\psi)\Gamma(n_j)}{\Gamma(\alpha_\psi + n_j)} \alpha_\psi^{k_j} h_y(y_j^*|x_j^*) \prod_{l=1}^{k_j} \Gamma(n_{l|j}) h_x(x_{l|j}^*).$$

Again, as in (3), the marginal likelihood component in the posterior distribution of ρ_n is the product of the cluster-specific marginal likelihoods, but now the nested clustering structure of the EDP separates the factors relative to x and $y|x$, being $h(x_{1:n}, y_{1:n}|\rho_n) = \prod_{j=1}^k h_y(y_j^*|x_j^*) \prod_{l=1}^{k_j} h_x(x_{l|j}^*)$. Even if the x-likelihood favors many ψ-clusters, now these can be obtained by subpartitioning a coarser θ-partition, and the number k of θ-clusters can be expected to be much smaller than in (3).

Further insights into the behavior of the random partition are given by the induced covariate-dependent random partition of the θ_i parameters given the covariates, which is detailed in the following propositions. We will use the notation $\mathcal{P}_n$ to denote the set of all possible partitions of the first n integers.

Proposition 3 *The covariate-dependent random partition model induced by the EDP prior is*

$$p(\rho_{n,y}|x_{1:n}) \propto \alpha_\theta^k \prod_{j=1}^{k} \sum_{\rho_{n_j,x} \in \mathcal{P}_{n_j}} \int_\Theta \frac{\Gamma(\alpha_\psi(\theta))\Gamma(n_j)}{\Gamma(\alpha_\psi(\theta)+n_j)} \alpha_\psi(\theta)^{k_j} dP_{0\theta}(\theta) \prod_{l=1}^{k_j} \Gamma(n_{l|j}) h_x(x^*_{l|j}).$$

Proof An application of Bayes theorem implies that

$$p(\rho_n|x_{1:n}) \propto \alpha_\theta^k \prod_{j=1}^{k} \int_\Theta \frac{\Gamma(\alpha_\psi(\theta))\Gamma(n_j)}{\Gamma(\alpha_\psi(\theta)+n_j)} \alpha_\psi(\theta)^{k_j} dP_{0\theta}(\theta) \prod_{l=1}^{k_j} \Gamma(n_{l|j}) h_x(x^*_{l|j}). \tag{10}$$

Integrating over $\rho_{n,x}$, or equivalently summing over all $\rho_{n_j,x}$ in $\mathcal{P}_{n_j,x}$ for $j = 1, \ldots, k$ leads to,

$$p(\rho_{n,y}|x_{1:n}) \propto \sum_{\rho_{n_{1+},x}} \cdots \sum_{\rho_{n_{k+},x}} \alpha_\theta^k \prod_{j=1}^{k} \int_\Theta \frac{\Gamma(\alpha_\psi(\theta))\Gamma(n_j)}{\Gamma(\alpha_\psi(\theta)+n_j)} \alpha_\psi(\theta)^{k_j} dP_{0\theta}(\theta) \prod_{l=1}^{k_j} \Gamma(n_{l|j}) h_x(x^*_{l|j}),$$

and, finally, since (10) is the product over the j terms, we can pull the sum over $\rho_{n_j,x}$ within the product. ■

This covariate-dependent random partition model will favor θ-partitions of the subjects which can be further partitioned into groups with similar covariates, where a partition with many desirable subpartitions will have higher mass.

Proposition 4 *The posterior of the random covariate-dependent partition induced from the EDP model is*

$$\begin{aligned} p(\rho_{n,y}|x_{1:n}, y_{1:n}) \propto & \alpha_\theta^k \prod_{j=1}^{k} h_y(y^*_j|x^*_j) \\ & \times \sum_{\rho_{n_j,x} \in \mathcal{P}_{n_j}} \int_\Theta \frac{\Gamma(\alpha_\psi(\theta))\Gamma(n_j)}{\Gamma(\alpha_\psi(\theta)+n_j)} \alpha_\psi(\theta)^{k_j} dP_{0\theta}(\theta) \prod_{h=1}^{k_j} \Gamma(n_{l|j}) h_x(x^*_{l|j}). \end{aligned}$$

The proof is similar in spirit to that of Proposition 3. Notice the preferred θ-partitions will consist of clusters with a similar relationship between y and x, as measured by marginal local model h_y for $y|x$ and similar x behavior, which is measured much more flexibly as a mixture of the previous marginal local models.

The behavior of the random partition, detailed above, has important implications for the posterior of the unique parameters. Conditionally on the partition, the cluster-specific parameters (θ^*, ψ^*) are still independent, their posterior density being

$$p(\theta^*, \psi^*|y_{1:n}, x_{1:n}, \rho_n) = \prod_{j=1}^{k} p(\theta^*_j|y^*_j, x^*_j) \prod_{l=1}^{k_j} p(\psi^*_{l|j}|x^*_{l|j}),$$

where

$$p(\theta_j^*|y_j^*, x_j^*) \propto p_{0\theta}(\theta_j^*) \prod_{i \in S_{j+}} K(y_i|\theta_j^*, x_i), \quad p(\psi_{l|j}^*|x_{l|j}^*) \propto p_{0\psi}(\psi_{l|j}^*) \prod_{i \in S_{j,l}} K(x_i|\psi_{l|j}^*).$$

The important point is that the posterior of θ_j^* can now be updated with much larger sample sizes if the data determines that a coarser θ-partition is present. This will result in a more reliable posterior mean, a smaller posterior variance, and larger influence of the data compared with the prior.

3.2 Covariate-dependent Urn Scheme and Prediction

Similar to the DP model, computation of the predictive estimates relies on a covariate-dependent urn scheme. For the EDP, we have

$$s_{y,n+1}|\rho_n, x_{1:n+1}, y_{1:n}) \sim \frac{\omega_{k+1}(x_{n+1})}{c_0}\, \delta_{k+1} + \sum_{j=1}^{k} \frac{\omega_j(x_{n+1})}{c_0} \delta_j, \tag{11}$$

where $c_0 = p(x_{n+1}|\rho_n, x_{1:n})$, but now the expression of the $\omega_j(x_{n+1})$ takes into account the possible allocation of x_{n+1} in subgroups, being

$$\omega_j(x_{n+1}) = \sum_{s_{x,n+1}} p(x_{n+1}|\rho_n, x_{1:n}, y_{1:n}, s_{y,n+1} = j, s_{x,n+1}) p(s_{x,n+1}|\rho_n, x_{1:n}, y_{1:n}, s_{y,n+1} = j).$$

From this, it can be easily found that

$$\omega_{k+1}(x_{n+1}) = \frac{\alpha_\theta}{\alpha_\theta + n}\, h_{0,x}(x_{n+1}),$$
$$\omega_j(x_{n+1}) = \frac{n_j}{\alpha_\theta + n} \left(\pi_{k_j+1|j} h_{0,x}(x_{n+1}) + \sum_{l=1}^{k_j} \pi_{l|j} h_{l|j,x}(x_{n+1}) \right),$$

where

$$\pi_{l|j} = \int \frac{n_{l|j}}{\alpha_\psi(\theta) + n_j} dP_{0\theta}(\theta|x_j^*, y_j^*), \quad \pi_{k_j+1|j} = \int \frac{\alpha_\psi(\theta)}{\alpha_\psi(\theta) + n_j} dP_{0\theta}(\theta).$$

In the case of constant $\alpha_\psi(\theta) = \alpha_\psi$, these expressions simplify to

$$\pi_{l|j} = \frac{n_{l|j}}{\alpha_\psi + n_j}, \quad \pi_{k_j+1|j} = \frac{\alpha_\psi}{\alpha_\psi + n_j}.$$

Notice that (11) is similar to the covariate-dependent urn scheme of the DP model. The important difference is that the weights, which measure the similarity between x_{n+1} and the j^{th} cluster, are much more flexible.

It follows, from similar computations as in Section 2.2, that the predictive density at y for a new subject with a covariate value of x_{n+1} is

$$f(y|y_{1:n}, x_{1:n+1})$$
$$= \sum_{\rho_n} \left(\frac{\omega_{k+1}(x_{n+1})}{c} h_{0,y}(y|x_{n+1}) + \sum_{j=1}^{k} \frac{\omega_j(x_{n+1})}{c} h_{j,y}(y|x_{n+1}) \right) p(\rho_n|x_{1:n}, y_{1:n}), \tag{12}$$

where $c = p(x_{n+1}|x_{1:n}, y_{1:n})$.

Under the squared error loss function, the point prediction of y_{n+1} is

$$\begin{aligned}&\mathrm{E}[Y_{n+1}|y_{1:n}, x_{1:n+1}]\\&= \sum_{\rho_n} \left(\frac{\omega_{k+1}(x_{n+1})}{c} \mathrm{E}_0[Y_{n+1}|x_{n+1}] + \sum_{j=1}^{k} \frac{\omega_j(x_{n+1})}{c} \mathrm{E}_j[Y_{n+1}|x_{n+1}] \right) p(\rho_n|x_{1:n}, y_{1:n}). \quad (13)\end{aligned}$$

The expressions for the prediction density (12) and point prediction (13) are quite similar to those of the DP, (8) and (9), respectively; in both cases, the cluster-specific predictive estimates are averaged with covariate-dependent weights. However, there are two important differences for the EDP model. The first is that the weights in (11) are defined with a more flexible kernel; in fact, it is a mixture of the original kernels used in the DP model. This means that we have a more flexible measure of similarity in the covariate space. The second difference is that k will be smaller and n_j will be larger with a high posterior probability, leading to a more reliable posterior distribution of θ_j^* due to larger sample sizes and better cluster-specific predictive estimates. We will demonstrate the advantage of these two key differences in simulated and applied examples of Sections 5 and 6.

4. Computations

Inference for the EDP model cannot be obtained analytically and must therefore be approximated. To obtain approximate inference, we rely on Markov Chain Monte Carlo (MCMC) methods and consider an extension of Algorithm 2 of Neal (2000) for the DP mixture model. In this approach, the random probability measure, $\mathbf{P}$, is integrated out, and the model is viewed in terms of $(\rho_n, \theta^*, \psi^*)$. This algorithm requires the use of conjugate base measures $P_{0\theta}$ and $P_{0\psi}$. To deal with non-conjugate base measures, the approach used in Algorithm 8 of Neal (2000) can be directly adapted to the EDP mixture model.

Algorithm 2 is a Gibbs sampler which first samples the cluster label of each subject conditional on the partition of all other subjects, the data, and (θ^*, ψ^*), and then samples (θ^*, ψ^*) given the partition and the data. The first step can be easily performed thanks to the Pólya urn which marginalizes the DP.

Extending Algorithm 2 for the EDP model is straightforward, since the EDP maintains a simple, analytically computable urn scheme. In particular, the conditional probabilities $p(s_i|\rho_{n-1}^{-i}, \theta^*, \psi^*, x_{1:n}, y_{1:n})$ (provided in the Appendix) have a simple closed form, which allows conditional sampling of the individual cluster membership indicators s_i, where ρ_{n-1}^{-i} denotes the partition of the $n-1$ subjects with the i^{th} subject removed. To improve mixing, we include an additional Metropolis-Hastings step; at each iteration, after performing the n Gibbs updates for each s_i, we propose a shuffle of the nested partition structure obtained by moving a ψ-cluster to be nested within a different or new θ-cluster. This move greatly improves mixing. A detailed description of the sampler, including the Metropolis-Hastings step, can be found in the Appendix.

MCMC produces approximate samples, $\{\rho_n^s, \psi^{*s}, \theta^{*s}\}_{s=1}^S$, from the posterior. The prediction given in Equation (13) can be approximated by

$$\frac{1}{S}\sum_{s=1}^{S}\frac{\omega_{k+1}^s(x_{n+1})}{\hat{c}}\mathrm{E}_{h_y}[Y_{n+1}|x_{n+1}] + \sum_{j=1}^{k^s}\frac{\omega_j^s(x_{n+1})}{\hat{c}}\mathrm{E}_{F_y}[Y_{n+1}|x_{n+1},\theta_j^{*s}],$$

where $\omega_j^s(x_{n+1})$ for $j = 1,\ldots,k^s+1$, are as previously defined in (11) with $(\rho_n, \psi^*, \theta^*)$ replaced by $(\rho_n^s, \psi^{*s}, \theta^{*s})$ and

$$\hat{c} = \frac{1}{S}\sum_{s=1}^{S}\omega_{k+1}^s(x_{n+1}) + \sum_{j=1}^{k^s}\omega_j^s(x_{n+1}).$$

For the predictive density estimate at x_{n+1}, we define a grid of new y values and for each y in the grid, we compute

$$\frac{1}{S}\sum_{s=1}^{S}\frac{\omega_{k+1}^s(x_{n+1})}{\hat{c}}h_y(y|x_{n+1}) + \sum_{j=1}^{k^s}\frac{\omega_j^s(x_{n+1})}{\hat{c}}K(y|x_{n+1},\theta_j^{*s}). \tag{14}$$

Note that hyperpriors may be included for the precision parameters, α_θ and $\alpha_\psi(\cdot)$, and the parameters of the base measures. For the simulated examples and application, a Gamma hyperprior is assigned to α_θ, and $\alpha_\psi(\theta)$ for $\theta \in \Theta$ are assumed to be i.i.d. from a Gamma hyperprior. At each iteration, α_θ^s and $\alpha_\psi^s(\theta)$ at θ_j^{*s} for $j = 1,\ldots,k^s$ are approximate samples from the posterior using the method described in Escobar and West (1995).

5. Simulated Example

We consider a toy example that demonstrates two key advantages of the EDP model; first, it can recover the true coarser θ-partition; second, improved prediction and smaller credible intervals result. The example shows that these advantages are evident even for a moderate value of p, with more drastic differences as p increases. A data set of $n = 200$ points were generated where only the first covariate is a predictor for Y. The true model for Y is a nonlinear regression model obtained as a mixture of two normals with linear regression functions and weights depending only on the first covariate;

$$Y_i|x_i \overset{ind}{\sim} p(x_{i,1})\mathrm{N}(y_i|\beta_{1,0}+\beta_{1,1}x_{i,1},\sigma_1^2) + (1-p(x_{i,1}))\mathrm{N}(y_i|\beta_{2,0}+\beta_{2,1}x_{i,1},\sigma_2^2),$$

where

$$p(x_{i,1}) = \frac{\tau_1\exp\left(-\frac{\tau_1^2}{2}(x_{1,i}-\mu_1)^2\right)}{\tau_1\exp\left(-\frac{\tau_1^2}{2}(x_{1,i}-\mu_1)^2\right)+\tau_2\exp\left(-\frac{\tau_2^2}{2}(x_{1,i}-\mu_2)^2\right)},$$

with $\beta_1 = (0,1)'$, $\sigma_1^2 = 1/16$, $\beta_2 = (4.5, 0.1)'$, $\sigma_2^2 = 1/8$ and $\mu_1 = 4$, $\mu_2 = 6$, $\tau_1 = \tau_2 = 2$. The covariates are sampled from a multivariate normal,

$$X_i = (X_{i,1},\ldots,X_{i,p})' \overset{iid}{\sim} \mathrm{N}(\mu,\Sigma), \tag{15}$$

centered at $\mu = (4, \ldots, 4)'$ with a standard deviation of 2 along each dimension, that is, $\Sigma_{h,h} = 4$. The covariance matrix Σ models two groups of covariates: those in the first group are positively correlated among each other and the first covariate, but independent of the second group of covariates, which are positively correlated among each other but independent of the first covariate. In particular, we take $\Sigma_{h,l} = 3.5$ for $h \neq l$ in $\{1, 2, 4, \ldots, 2\lfloor p/2 \rfloor\}$ or $h \neq l$ in $\{3, 5, \ldots, 2\lfloor (p-1)/2 \rfloor + 1\}$ and $\Sigma_{h,l} = 0$ for all other cases of $h \neq l$.

We examine both the DP and EDP mixture models;

$$Y_i | x_i, \beta_i, \sigma^2_{y,i} \overset{ind}{\sim} \mathrm{N}(\underline{x}_i \beta_i, \sigma^2_{y,i}), \quad X_i | \mu_i, \sigma^2_{x,i} \overset{ind}{\sim} \prod_{h=1}^{p} \mathrm{N}(\mu_{i,h}, \sigma^2_{x,h,i}),$$
$$(\beta_i, \sigma^2_{y,i}, \mu_i, \sigma^2_{x,i}) | P \overset{iid}{\sim} P$$

with $P \sim$ DP or $P \sim$ EDP. The conjugate base measure is selected; $P_{0\theta}$ is a multivariate normal inverse gamma prior and $P_{0\psi}$ is the product of p normal inverse gamma priors, that is

$$p_{0\theta}(\beta, \sigma^2_y) = \mathrm{N}(\beta; \beta_0, \sigma^2_y C^{-1}) \mathrm{IG}(\sigma^2_y; a_y, b_y),$$

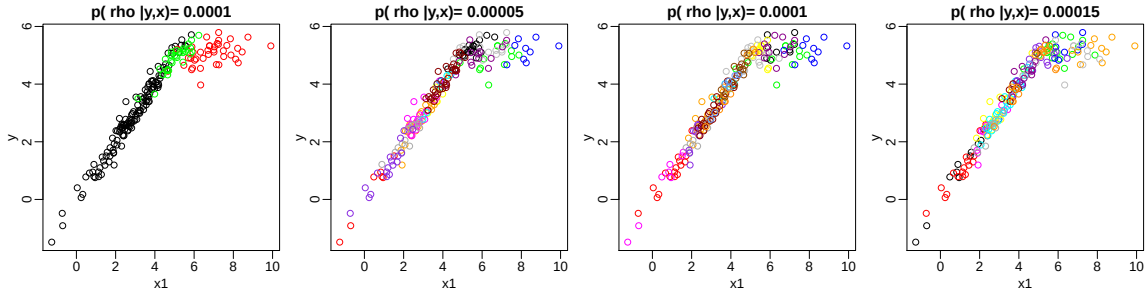
and

$$p_{0\psi}(\mu, \sigma^2_x) = \prod_{h=1}^{p} \mathrm{N}(\mu_h; \mu_{0,h}, \sigma^2_{x,h} c_h^{-1}) \mathrm{IG}(\sigma^2_{x,h}; a_{x,h}, b_{x,h}).$$

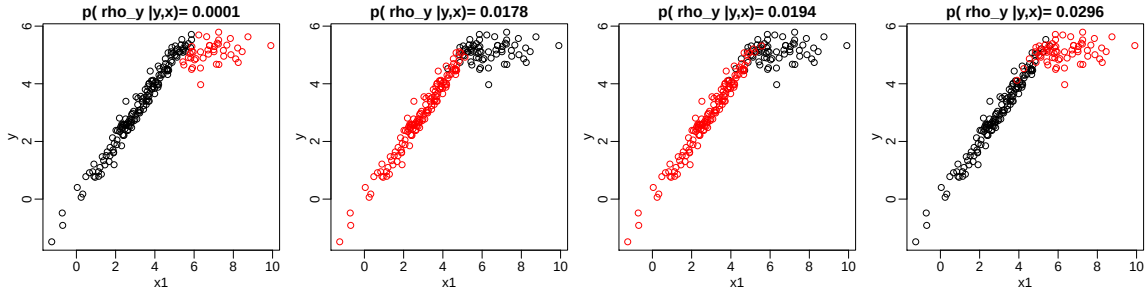
For both models, we use the same subjective choice of the parameters of the base measure. In particular, we center the base measure on an average of the true parameters values with enough variability to recover the true model. A list of the prior parameters can be found in the Appendix. We assign hyperpriors to the mass parameters, where for the DP model, $\alpha \sim \mathrm{Gamma}(1, 1)$, and for the EDP model, $\alpha_\theta \sim \mathrm{Gamma}(1, 1)$, $\alpha_\psi(\beta, \sigma^2_y) \overset{iid}{\sim} \mathrm{Gamma}(1, 1)$ for all $\beta, \sigma^2_y \in \mathbb{R}^{p+1} \times \mathbb{R}_+$.

The computational procedures described in Section 4 were used to obtain posterior inference with 20,000 iterations and burn in period of 5,000. An examination of the trace and autocorrelation plots for the subject-specific parameters $(\beta_i, \sigma^2_{y,i}, \mu_i, \sigma^2_{x,i})$ provided evidence of convergence. Additional criteria for assessing the convergence of chain, in particular, the Geweke diagnostic, also suggested convergence, and the results are given in Table 6 of the Appendix (see the **R** package *coda* for implementation and further details of the diagnostic). It should be noted that running times for both models are quite similar, although slightly faster for the DP.

The first main point to emphasize is the improved behavior of the posterior of the random partition for the EDP. We note that for both models, the posterior of the partition is spread out. This is because the space of partitions is very large and many partitions are very similar, differing only in a few subjects; thus, many partitions fit the data well. We depict representative partitions of both models with increasing p in Figure 1. Observations are plotted in the x_1–y space and colored according to the partition for the DP and the θ-partition for the EDP. As expected, for $p = 1$ the DP does well at recovering the true partition, but as clearly seen from Figure 1, for large values of p, the DP partition is comprised of many clusters, which are needed to approximate the multivariate density of X. In fact, the density of $Y|x$ can be recovered with only two kernels regardless of p, and



(a) DP, p=1 (b) DP, p=5 (c) DP, p=10 (d) DP, p=15



(e) EDP, p=1 (f) EDP, p=5 (g) EDP, p=10 (h) EDP, p=15

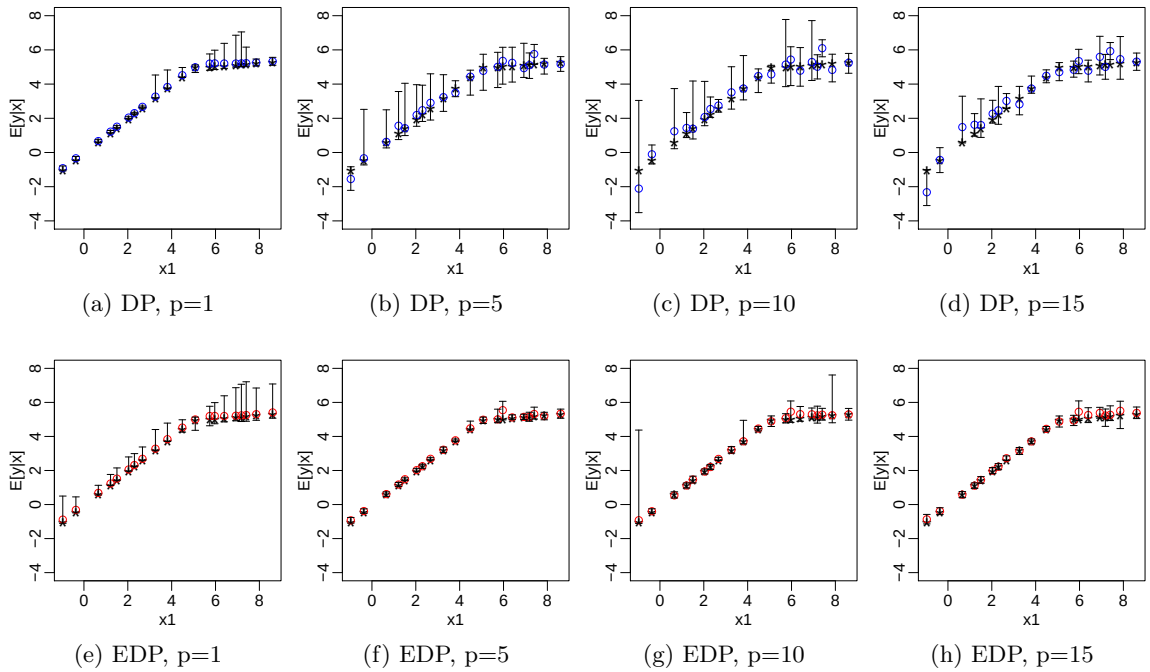
Figure 1: The y partition with the highest estimated posterior probability. Data points are plotted in the x_1 vs. y space and colored by cluster membership with estimated posterior probability included in the plot title.

	$p=1$		$p=5$		$p=10$		$p=15$	
	$\hat{k}$	$\hat{\alpha}$	$\hat{k}$	$\hat{\alpha}$	$\hat{k}$	$\hat{\alpha}$	$\hat{k}$	$\hat{\alpha}$
DP	3	0.51	14	2.75	16	3.25	15	3.09
EDP	3	0.56	2	0.35	2	0.39	2	0.36

Table 1: The posterior mode number of y clusters, denoted $\hat{k}$ for both models, and the posterior mean of α, α_θ, denoted $\hat{\alpha}$ for both models, as p increases.

the θ-partitions of the EDP depicted in Figure 1, with only two θ-clusters, are very similar to the true configuration, even for increasing p. On the other hand, the (θ, ψ)-partition of the EDP (not shown) consists of many clusters and resembles the partition of the DP model.

This behavior is representative of the posterior distribution of the random partition that, for the DP, has a large posterior mode of k and large posterior mean of α for larger values of p, while most of the EDP θ-partitions are composed of only 2 clusters with only a handful of subjects placed in the incorrect cluster and the posterior mean of α_θ is much smaller for large p. Table 1 summarizes the posterior of k for both models and the posterior of the precision parameters α, α_θ. It is interesting to note that posterior samples of $\alpha_\psi(\theta)$



(a) DP, p=1 (b) DP, p=5 (c) DP, p=10 (d) DP, p=15

(e) EDP, p=1 (f) EDP, p=5 (g) EDP, p=10 (h) EDP, p=15

Figure 2: The point predictions for 20 test samples of the covariates are plotted against x_1 and represented with circles (DP in blue and EDP in red) with true prediction as black stars. The bars about the prediction depict the 95% credible intervals.

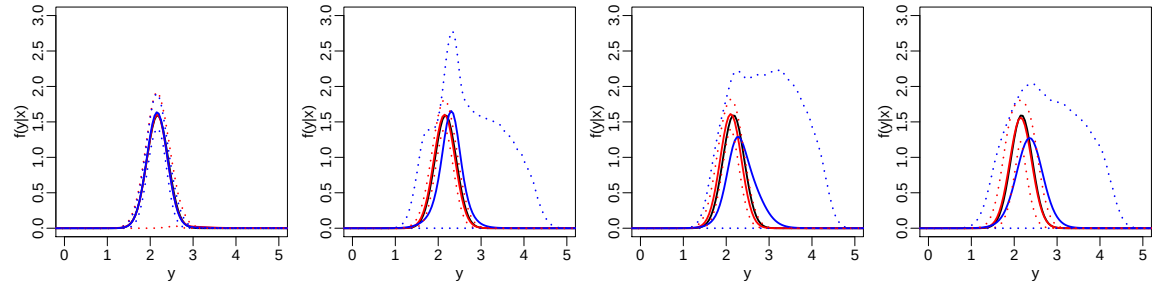
	$p=1$		$p=5$		$p=10$		$p=15$	
	$\hat{l}_1$	$\hat{l}_2$	$\hat{l}_1$	$\hat{l}_2$	$\hat{l}_1$	$\hat{l}_2$	$\hat{l}_1$	$\hat{l}_2$
DP	**0.03**	**0.05**	0.16	0.2	0.25	0.34	0.26	0.34
EDP	0.04	**0.05**	**0.06**	**0.1**	**0.09**	**0.16**	**0.12**	**0.21**

Table 2: Prediction error for both models as p increases.

for θ characteristic of the first cluster tend to be higher than posterior samples of $\alpha_\psi(\theta)$ for the second cluster; that is, more clusters are needed to approximate the density of X within the first cluster. A non-constant $\alpha_\psi(\theta)$ allows us to capture this behavior.

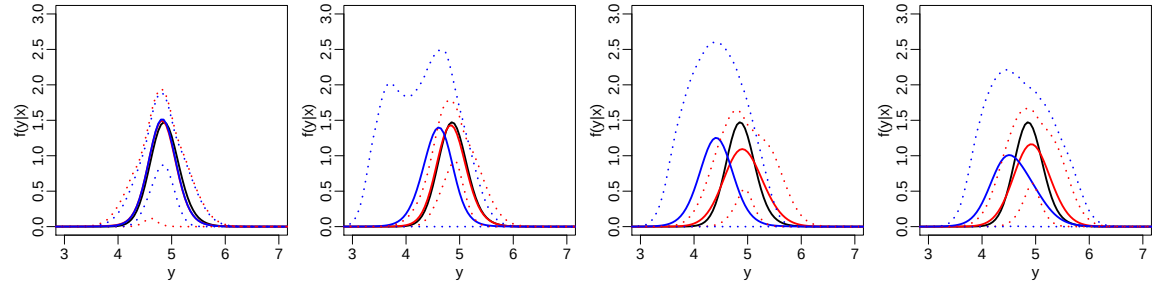
As discussed in Section 3, a main point of our proposal is the increased finite sample efficiency of the EDP model. To illustrate this, we create a test set with $m = 200$ new covariates values simulated from (15) with maximum dimension $p = 15$ and compute the true regression function $\mathrm{E}[Y_{n+j}|x_{n+j}]$ and conditional density $f(y|x_{n+j})$ for each new subject. To quantify the gain in efficiency of the EDP model, we calculate the point prediction and predictive density estimates from both models and compare them with the truth.

Judging from both the empirical l_1 and l_2 prediction errors, the EDP model outperforms the DP model, with greater improvement for larger p; see Table 2. Figure 2 displays the prediction against x_1 for 20 new subjects. Recall that each new x_1 is associated with different values of $(x_2, \ldots, x_p)$, which accounts for the somewhat erratic behavior of the



(a) p=1 (b) p=5 (c) p=10 (d) p=15

Figure 3: Predictive density estimate (DP in blue and EDP in red) for the first new subject with true conditional density in black. The pointwise 95% credible intervals are depicted with dashed lines (DP in blue and EDP in red).



(a) p=1 (b) p=5 (c) p=10 (d) p=15

Figure 4: Predictive density estimate (DP in blue and EDP in red) for the fifth new subject with true conditional density in black. The pointwise 95% credible intervals are depicted with dashed lines (DP in blue and EDP in red).

	$p=1$	$p=5$	$p=10$	$p=15$
DP	0.13	0.5	0.66	0.68
EDP	**0.12**	**0.15**	**0.24**	**0.29**

Table 3: l_1 density regression error for both models as p increases.

prediction as a function of x_1 for increasing p. The comparison of the credible intervals is quite interesting. For $p > 1$, the unnecessarily wide credible intervals for the DP regression model stand out in the first row of Figure 2. This is due to small cluster samples sizes for the DP model with $p > 1$.

The density regression estimates for all new subjects were computed by evaluating (14) at a grid of y-values. As a measure of the performance of the models, the empirical l_1 distance between the true and estimated conditional densities for each of the new covariate values is shown in Table 3. Again the EDP model outperforms the DP model. Figures

3 and 4 display the true conditional density (in black) for two new covariate values with the estimated conditional densities in blue for the DP and red for the EDP. It is evident that for $p > 1$ the density regression estimates are improved and that the pointwise 95% credible intervals are almost uniformly wider both in y and x for the DP model, sometimes drastically so. It is important to note that while the credible intervals of the EDP model are considerably tighter, they still contain the true density.

6. Alzheimer's Disease Study

Our primary goal in this section is to show that the EDP model leads to improved inference in a real data study with the important goal of diagnosing Alzheimer's disease (AD). In particular, EDP leads to improved predictive accuracy, tighter credible intervals around the predictive probability of having the disease, and a more interpretable clustering structure.

Alzheimer's disease is a prevalent form of dementia that slowly destroys memory and thinking skills, and eventually even the ability to carry out the simplest tasks. Unfortunately, a definitive diagnosis cannot be made until autopsy. However, the brain may show severe evidence of neurobiological damage even at early stages of the disease before the onset of memory disturbances. As this damage may be difficult to detect visually, improved methods for automatically diagnosing disease based on MRI neuroimaging is needed.

Data were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database, which is publicly accessible at UCLA's Laboratory of Neuroimaging.[1] The data set consists of summaries of fifteen brain structures computed from structural MRI obtained at the first visit for 377 patients, of which 159 have been diagnosed with AD and 218 are cognitively normal (CN). The covariates include whole brain volume (BV), intracranial volume (ICV), volume of the ventricles (VV), left and right hippocampal volume (LHV, RHV), volume of the left and right inferior lateral ventricle (LILV, RILV), thickness of the left and right middle temporal cortex (LMT, RMT), thickness of the left and right inferior temporal cortex (LIT, RIT), thickness of the left and right fusiform cortex (LF, RF), and thickness of the left and right entorhinal cortex (LE, RE). Volume is measured in cm^3 and cortical thickness is measured in mm.

The response is a binary variable with 1 indicating a cognitively normal subject and 0 indicating a subject who has been diagnosed with AD. The covariate is the 15-dimensional

1. The ADNI was launched in 2003 by the National Institute on Ageing (NIA), the National Institute of Biomedical Imaging and Bioengineering (NIBIB), the Food and Drug Administration (FDA), private pharmaceutical companies and non-profit organizations, as a $ 60 million, 5-year public-private partnership. The primary goal of ADNI has been to test whether serial magnetic resonance imaging (MRI), positron emission tomography (PET), other biological markers, and clinical and neuropsychological assessment can be combined to measure the progression of mild cognitive impairment (MCI) and early Alzheimer's disease (AD). Determination of sensitive and specific markers of very early AD progression is intended to aid researchers and clinicians to develop new treatments and monitor their effectiveness, as well as lessen the time and cost of clinical trials. The Principal Investigator of this initiative is Michael W. Weiner, MD, VA Medical Center and University of California-San Francisco. ADNI is the result of efforts of many co-investigators from a broad range of academic institutions and private corporations, and subjects have been recruited from over 50 sites across the U.S. and Canada. The initial goal of ADNI was to recruit 800 adults, ages 55 to 90, to participate in the research, approximately 200 cognitively normal older individuals to be followed for 3 years, 400 people with MCI to be followed for 3 years and 200 people with early AD to be followed for 2 years. For up-to-date information, see `www.adni-info.org`.

vector of measurements of various brain structures. Our model builds on local probit models and can be stated as follows:

$$Y_i|x_i,\beta_i \overset{ind}{\sim} \text{Bern}(\Phi(\underline{x}_i\beta_i)), \quad X_i|\mu_i,\sigma_i^2 \overset{ind}{\sim} \prod_{h=1}^{p} \text{N}(\mu_{i,h},\sigma_{i,h}^2),$$
$$(\beta_i,\mu_i,\sigma_i^2)|P \overset{iid}{\sim} P, \quad \mathbf{P}\sim Q.$$

The analysis is first carried using a DP prior for $\mathbf{P}$ with mass parameter α and base measure $P_{0\beta}\times P_{0\psi}$, with $P_{0\beta}=\text{N}(0_p,C^{-1})$ and $P_{0\psi}$ defined as the product of p normal inverse gamma measures. A list of the prior parameters can be found in the Appendix. The mass parameter is given a hyperprior of $\alpha\sim\text{Gamma}(1,1)$.

The prior parameters were carefully selected based on prior knowledge of the brain structures and their relationship with the disease and empirical evidence. The base measure for β was chosen to be centered on zero because even though we have prior belief about how each structure is related to AD individually, the joint relationship may be more complex. For simplicity, the covariance matrix is diagonal. The variances were chosen to reflect belief in the maximum range of the coefficient for each brain structure. We also explored the idea of defining C through a g-prior, where $C^{-1}=g(\underline{X}'\underline{X})^{-1}$ with g fixed or given a hyperprior. However, this proposal was unsatisfactory because prior information about the maximum range of the coefficient for each brain structure is condensed in a single parameter g. For example, there was no way to incorporate the belief that while the variability of hippocampal volume and inferior lateral ventricular volume are similar, the correlation between hippocampal volume and disease status is stronger. The parameters of the base measure for X were chosen based on prior knowledge and exploratory analysis of the average volume and cortical thickness of the brain structures (μ_0) and variability (b_x). The parameter a_x was chosen to equal 2, so that mean of the inverse gamma prior is properly defined and the variance is relatively large. The parameter c_x is equal to 1/2 to increase variability of μ given σ_x.

In this example, correlation between the measurements of the brain structures is expected. Furthermore, univariate histograms of the covariates show non-normal behavior. These facts suggest that many Gaussian kernels with local independence of the covariates will be needed to approximate the density of X. The conditional density of the response, on the other hand, may not be so complicated. This motivates the choice of an EDP prior. We emphasize that the same conjugate base measure is used with the identical subjective choice of parameters. The hyperpriors for the mass parameter of

$$\alpha_\beta\sim\text{Gamma}(1,1), \quad \alpha_\psi(\beta)\overset{iid}{\sim}\text{Gamma}(1,1) \quad \forall\beta\in\mathbb{R}^{p+1}.$$

If $\alpha_\psi(\beta)\approx 0$ for all $\beta\in\mathbb{R}^{p+1}$ the model converges to a DP mixture model, suggesting that the extra flexibility of the EDP is unnecessary. On the other hand, $\alpha_\beta\approx 0$ suggests that a linear model is sufficient for modelling the conditional response distribution.

The data were randomly split into a training sample of size 185 and a test sample of size 192. Inference is based on the algorithm explained in Section 4 with the added step of sampling a latent normal variable to deal with the binary response. For both results the number of iterations is 50,000 with burn in period of 10,000. An examination of the trace

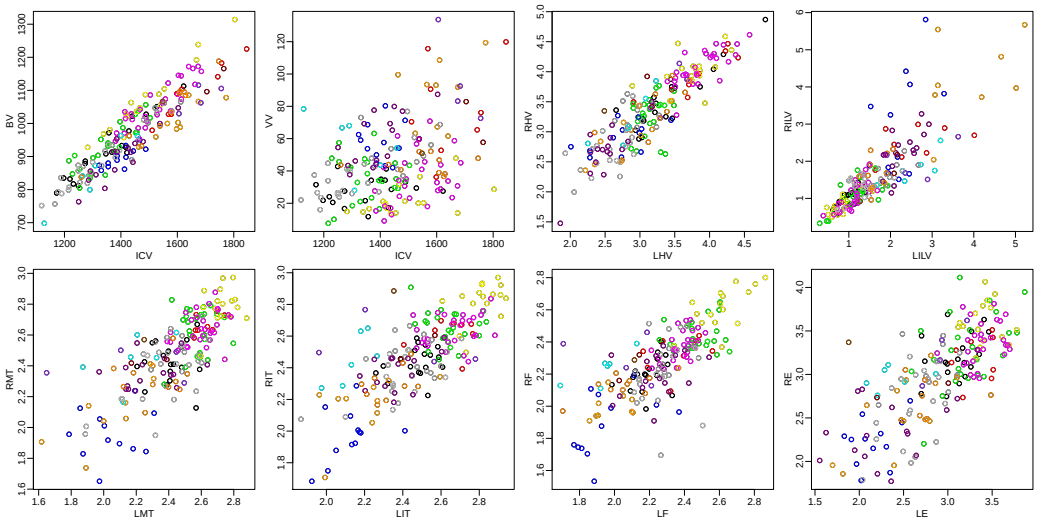


Figure 5: Data points are plotted in the covariate space and colored by the partition with the highest posterior probability for the DP model.

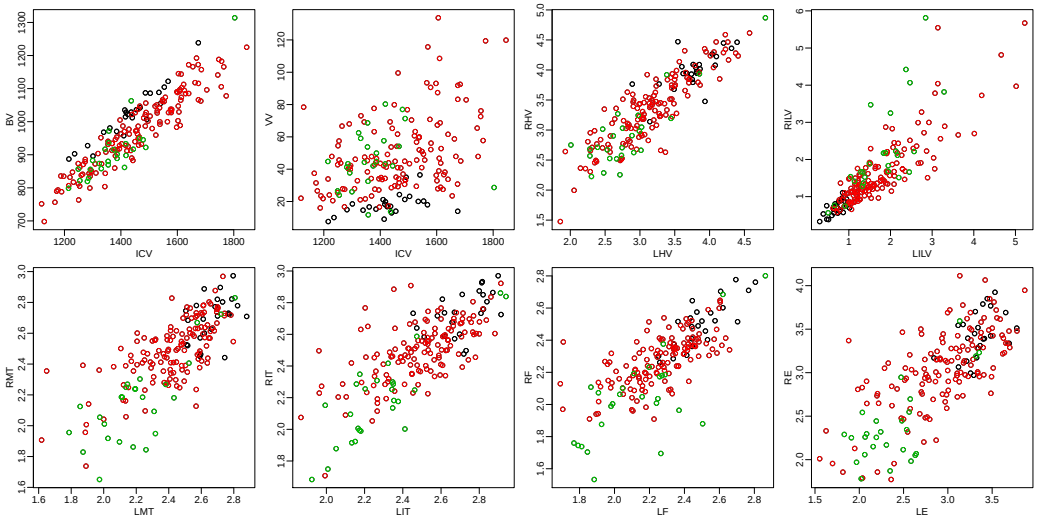


Figure 6: Data points are plotted in the covariate space and colored by the β-partition with the highest posterior probability for the EDP model.

and autocorrelation plots for the subject-specific parameters $(\beta_i, \mu_i, \sigma_i^2)$ provided evidence of convergence, which was further checked via the Geweke and Raftery and Lewis methods. Computation times are quite similar for both models, although slightly faster for the DP.

The DP based model requires many kernels to approximate the joint distribution, while the EDP prefers a coarser β-partition for the conditional density of $Y|x$. The posterior of k and the precision parameters α and α_β are summarized and compared for the two models in

	$\hat{k}$	$\tilde{k}$	$[k_l, k_u]$	$\hat{\alpha}$	$[\alpha_l, \alpha_u]$
DP	16	16	[14,19]	3.41	[1.79, 5.59]
EDP	3	4	[2,7]	0.71	[0.11, 1.76]

Table 4: The posterior of k for both models is summarized through the posterior mode, denoted $\hat{k}$; the posterior median, denoted $\tilde{k}$; and the 95% credible intervals, denoted $[k_l, k_u]$. The posterior of the precision parameter α for the DP and α_β for the EDP is summarized through the posterior mode, denoted $\hat{\alpha}$, and 95% credible intervals, denoted $[\alpha_l, \alpha_u]$.

	Accuracy	AUC	MXE
DP	82.81%	0.88	0.42
EDP	86.46%	0.90	0.38

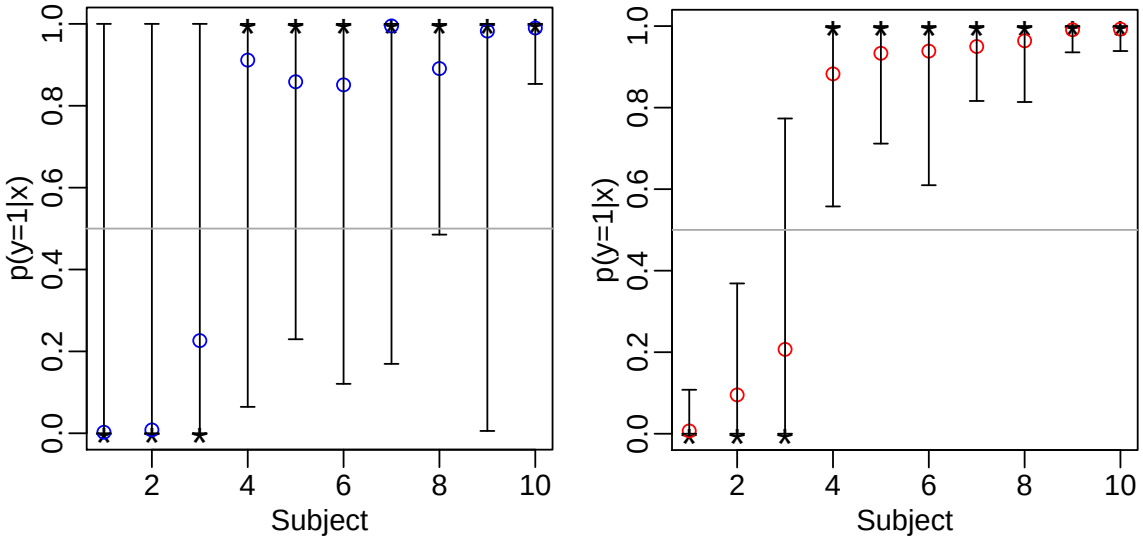
Table 5: Predictive accuracy, area under the ROC curve (AUC), and mean cross entropy (MXE) for the test set.

Table 4. Posterior samples of $\alpha_\psi(\beta)$ tend to be higher for average values of β, meaning that more kernels are needed for the density of X within such β-clusters. The added flexibility of a β-dependent precision parameter for ψ allows us to capture this behavior.

The posterior of the partition is quite spread out across many similar partitions for the DP and EDP models. A representative partition, the partition with the highest estimated posterior probability, for the DP mixture models is depicted in Figure 5, where the data points are plotted in the covariate space and colored by the partition. Notice the high number of kernels with small sample sizes within each cluster. Figure 6 depicts a representative β-partition for the EDP mixture model, where the data points are colored by the β-partition. Not only are there fewer clusters with larger sample sizes, but the clusters are much more interpretable as well. In particular, the posterior concentrates on partitions similar to the one depicted in Figure 6 with a general cluster and two extreme clusters of 100% AD and 100% non-AD patients (the green and black clusters, respectively). The black cluster of non-AD patients display high brain volume compared to intracranial volume, low ventricular volume, high hippocampal volume, and high cortical thickness; this behavior could be expected as AD is associated with shrinking brain tissue and increased cerebrospinal fluid, while intracranial volume remained fixed. The green cluster of AD patients display lower hippocampal volume and interestingly, low intracranial volume and low cortical thickness with a "right-less-than-left" pattern.

To quantify the gain in efficiency with the EDP model, we computed the predictive accuracy, area under the ROC curve (AUC), and mean cross entropy (MXE) for the test set, using the *ROCR* package in **R**. The larger sample sizes of the EDP model improve all predictive criteria when compared to the DP model (Table 5).

Finally, we note that by allowing for a coarser β-partition when appropriate, the increased cluster sample sizes not only result in improved accuracy for the EDP model but also allow for much tighter credible intervals. This is shown in Figure 7 which depicts the



(a) DP (b) EDP

Figure 7: Predicted probability of being healthy against subject index for 10 new subjects represented with circles (DP in blue and EDP in red) with the true outcome as black stars. The bars about the prediction depict the 95% credible intervals.

estimated probability of being healthy for 10 subjects along with lower and upper bounds for 95% credible intervals, as a function of subject index. Notice the tighter credible intervals for the EDP model with some dramatic examples given by subjects 1 and 8.

We also compared with Gaussian process (GP), support vector machine (SVM), and random forest (RF) models, which are implemented in the *kernlab* and *randomForest* packages in **R**. The results of the EDP are comparable with these other standard nonparametric classification methods, in particular the predictive accuracy of the GP, SVM, and RF models are 85.42%, 86.46%, and 86.46%, respectively. The results remain quite comparable with different random splits into training and test sets, which also confirmed the conclusions suggested by Figures 5, 6, and 7 and Tables 4 and 5.

7. Discussion

In this paper, we have highlighted a drawback of DP mixture models when the aim is estimation of the regression function and conditional distribution. We have proposed a simple, but efficient, solution based on the EDP, which overcomes the problems of the DP mixture model by introducing a nested partition structure. An important feature of the proposed EDP mixture model is that computations remain relatively simple; unlike other modifications of the DP for conditional distribution modeling we maintain the ability to marginalize out the random measure and induce a simple urn scheme. In scaling up to larger numbers of predictors p, this simplified structure should be advantageous.

Acknowledgments

We thank the referees and associate editor for their helpful comments. Data collection and sharing for the application in Section 6 of this work was funded by the Alzheimer's Disease Neuroimaging Initiative (ADNI) (National Institutes of Health Grant U01 AG024904). We acknowledge the funding contributions of ADNI supporters (`adni-info.org/Scientists/ADNISponsors.aspx`).

Appendix A. Computations

This appendix contains further details of the MCMC algorithm described in Section 4.

The conditional distribution of $s_i = (s_{i,y}, s_{i,x})$, which denotes the vector containing the y-cluster and x-cluster membership for subject i, is

$$s_i|\rho_{n-1}^{-i}, \theta^*, \psi^*, x_{1:n}, y_{1:n} \sim \frac{\omega_{k^{-i}+1,1}(y_i, x_i)}{c_1}\delta_{(k^{-i}+1,1)} + \sum_{j=1}^{k^{-i}}\left(\frac{\omega_{j,k_j^{-i}+1}(y_i, x_i)}{c_1}\delta_{(j,k_j^{-i}+1)} + \sum_{l=1}^{k_j^{-i}}\frac{\omega_{j,l}(y_i, x_i)}{c_1}\delta_{(j,l)}\right), \tag{16}$$

where for $j = 1, \ldots, k^{-i}$ and $l = 1, \ldots, k_j^{-i}$,

$$\omega_{j,l}(y_i, x_i) = \frac{n_j^{-i} n_{l|j}^{-i}}{\alpha_\psi(\theta_j^{*-i}) + n_j^{-i}} K(y_i|x_i, \theta_j^{*-i}) K(x_i|\psi_{l|j}^{*-i}),$$

for $j = 1, \ldots, k^{-i}$,

$$\omega_{j,k_j^{-i}+1}(y_i, x_i) = \frac{n_j^{-i}\alpha_\psi(\theta_j^{*-i})}{\alpha_\psi(\theta_j^{*-i}) + n_j^{-i}} K(y_i|x_i, \theta_j^{*-i}) h_x(x_i),$$

$$\omega_{k^{-i}+1,1}(y_i, x_i) = \alpha_\theta h_y(y_i|x_i) h_x(x_i),$$

and

$$c_1 = \omega_{k^{-i}+1,1}(y_i, x_i) + \sum_{j=1}^{k^{-i}}\left(\omega_{j,k_j^{-i}+1}(y_i, x_i) + \sum_{l=1}^{k_j^{-i}}\omega_{j,l}(y_i, x_i)\right).$$

Here, ρ_{n-1}^{-i} represents the partition of the $n-1$ subjects with the i^{th} subject removed where $k^{-i}, k_j^{-i}, n_j^{-i}, n_{l|j}^{-i}$ are defined from ρ_{n-1}^{-i}. Similarly, θ_j^{*-i} and $\psi_{l|j}^{*-i}$ are the unique cluster parameters associated to the clusters of ρ_{n-1}^{-i}.

Next, we describe the Metropolis-Hastings step, which proposes to move a ψ-cluster to be nested with a different or new θ-cluster. This step is separated into three possible moves: 1) a ψ-cluster, among those within θ-clusters with more than one ψ-cluster, is moved to a different θ-cluster; 2) a ψ-cluster, among those within θ-clusters with more than one ψ-cluster, is moved to a new θ-cluster; 3) a ψ-cluster, among those within θ-clusters with only one ψ-cluster, is moved to a different θ-cluster.

Let $k_{x,2+}$ be the number of ψ-clusters within a θ-cluster with more than one ψ-cluster and $k_{x,1}$ be the number of ψ-clusters within a θ-cluster with only one ψ-cluster. The proposal distributions for the three moves are as follows. For the first move, the ψ-cluster is uniformly selected with probability $k_{x,2+}^{-1}$ and moved within a different θ-cluster selected uniformly with probability $(k-1)^{-1}$. For the second, the ψ-cluster is again uniformly selected with probability $k_{x,2+}^{-1}$ and moved to a new cluster. Lastly, for the third, the ψ-cluster is uniformly selected with probability $k_{x,1}^{-1}$ and moved within a different θ-cluster selected uniformly with probability $(k-1)^{-1}$.

Let ρ_n^* be the proposed partition defined by moving ψ-cluster l in θ-cluster j to θ-cluster h. For the first move, $h \in \{1, \ldots, j-1, j+1, \ldots, k\}$ and the acceptance probability is $\min(1, p)$, where

$$p = \frac{\Gamma(n_j - n_{l|j})\Gamma(n_h + n_{l|j})}{\Gamma(n_j)\Gamma(n_h)} \frac{\Gamma(\alpha_\psi(\theta_j^*) + n_j)\Gamma(\alpha_\psi(\theta_h^*) + n_h)}{\Gamma(\alpha_\psi(\theta_j^*) + n_j - n_{l|j})\Gamma(\alpha_\psi(\theta_h^*) + n_h + n_{l|j})} \frac{\alpha_\psi(\theta_h^*)}{\alpha_\psi(\theta_j^*)} \frac{\prod_{i \in S_{l|j}} K(y_i|x_i, \theta_h^*)}{\prod_{i \in S_{l|j}} K(y_i|x_i, \theta_j^*)} \frac{k_{x,2+}}{k_{x,2+}^*},$$

and $k_{x,2+}^*$ is the number of ψ-clusters within a θ-cluster with more than one ψ-cluster under the proposed partition. For the second move, $h = k+1$ and let $\theta_{k+1}^* \sim P(\theta|y_{l|j}^*, x_{l|j}^*)$ be the proposed parameter of the $k+1$ θ-cluster. The acceptance probability is $\min(1, p)$, where

$$p = \frac{\Gamma(n_j - n_{l|j})\Gamma(n_{l|j})}{\Gamma(n_j)} \frac{\Gamma(\alpha_\psi(\theta_j^*) + n_j)\Gamma(\alpha_\psi(\theta_{k+1}^*))}{\Gamma(\alpha_\psi(\theta_j^*) + n_j - n_{l|j})\Gamma(\alpha_\psi(\theta_{k+1}^*) + n_{l|j})} \alpha_\theta \frac{\alpha_\psi(\theta_{k+1}^*)}{\alpha_\psi(\theta_j^*)} \frac{h_y(y_{l|j}^*|x_{l|j}^*)}{\prod_{i \in S_{l|j}} K(y_i|x_i, \theta_j^*)} \frac{k_{x,2+}}{k_{x,1}^* k},$$

and $k_{x,1}^*$ is the number of ψ-clusters within a θ-cluster with only one ψ-cluster under the proposed partition. Finally, for the third move, $h \in \{1, \ldots, j-1, j+1, \ldots, k\}$ and the acceptance probability is $\min(1, p)$, where

$$p = \frac{\Gamma(n_h + n_{l|j})}{\Gamma(n_{l|j})\Gamma(n_h)} \frac{\Gamma(\alpha_\psi(\theta_j^*) + n_{l|j})\Gamma(\alpha_\psi(\theta_h^*) + n_h)}{\Gamma(\alpha_\psi(\theta_j^*))\Gamma(\alpha_\psi(\theta_h^*) + n_h + n_{l|j})} \frac{1}{\alpha_\theta} \frac{\alpha_\psi(\theta_h^*)}{\alpha_\psi(\theta_j^*)} \frac{\prod_{i \in S_{l|j}} K(y_i|x_i, \theta_h^*)}{h_y(y_{l|j}^*|x_{l|j}^*)} \frac{k_{x,1}k - 1}{k_{x,2+}^*}.$$

Each iteration of the MCMC algorithm is summarized as follows:

- For $i = 1, \ldots, n$,
  - if $s_{i,y} = j$ and $n_j^{-i} = 0$,
    - then remove θ_j^* and $\psi_{l|j}^*$ from (θ^*, ψ^*).
  - Otherwise, if $s_{i,y} = j$, $s_{i,x} = l$ and $n_{l|j}^{-i} = 0$,
    - then remove $\psi_{l|j}^*$ from ψ^*.
  - Next, sample s_i given $\rho_{n-1}^{-i}, \theta^*, \psi^*, x_{1:n}, y_{1:n}$ as defined by Equation (16).
  - If $s_{i,y} = k^{-i} + 1$,

		$p=1$	$p=5$	$p=10$	$p=15$
$\beta_{0,i}$	DP	-0.91	-0.13	-1.22	-3.15
	EDP	-1.86	0.59	1.24	-0.67
$\beta_{1,i}$	DP	0.61	-0.22	0.79	3.25
	EDP	1.54	-0.64	2.08	-0.42
$\sigma^2_{y,i}$	DP	-1.51	1.35	-0.13	1.06
	EDP	0.49	1.1	-2.09	-3.51

Table 6: The Z-score from Geweke diagnostic for the subject-specific θ-parameters of one subject.

    - $*$ sample $\theta^*_{k^{-i}+1}$ given y_i, x_i and $\psi^*_{1|k^{-i}+1}$ given x_i and concatenate them to (θ^*, ψ^*).
  - Otherwise, if $s_{i,y} = j$ and $s_{i,x} = k_j^{-i} + 1$,
    - $*$ sample $\psi^*_{k_j^{-i}+1|j}$ given x_i and concatenate it to ψ^*.

- Carry out the first move described in the Metropolis-Hastings step.
- Sample $u \sim \mathrm{U}(0,1)$. If $u < 0.5$, perform move 2, otherwise perform move 3.
- For $j = 1, \ldots, k$,
  - sample θ^*_j given (y^*_j, x^*_j), that is, from the posterior based on $p_{0\theta}(\theta^*_j)$ and $\prod_{i \in S_{j+}} K(y_i|x_i, \theta^*_j)$,
  - and for $l = 1, \ldots, k_j$,
    - $*$ sample $\psi^*_{l|j}$ given $x^*_{l|j}$, that is, from the posterior based on $p_{0\psi}(\psi^*_{l|j})$ and $\prod_{i \in S_{j,l}} K(x_i|\psi^*_{l|j})$.

Appendix B. Simulation Study

The prior parameters used for the simulation study in Section 5 are $\beta_0 = (2.25, 0.55, 0, \ldots, 0)'$, $C = \mathrm{diag}\,(0.05, 1, \ldots, 1)$; $a_y = 2$, $b_y = 0.1$, $\mu_0 = (4, \ldots, 4)'$, $c = (0.25, \ldots, 0.25)'$; $a_x = (2, \ldots, 2)'$, $b_x = (1, \ldots, 1)'$.

Table 6 lists the Z-scores for the θ parameters of the one subject from the Geweke diagnostic for assessing convergence of the MCMC chain, which are slightly high for $p = 10$ and $p = 15$ but a thinning of 2 improves the scores.

Appendix C. Alzheimer's Disease Study

For the prior parameters for the AD study in Section 6, C^{-1} is a diagonal matrix with diagonal elements $(400, .0001, .0001, 0.0004, 4, 4, .25, .25, 4, 4, 4, 4, 1, 1, 1, 1)$, and $\mu_0 = (1000, 1450, 45, 3.25, 3.25, 2, 2, 2.4, 2.4, 2.5, 2.5, 2.3, 2.3, 2.75, 2.75)'$; $c_{x,h} = 1/2, a_{x,h} = 2\ \forall h$; and $b_x = (10000, 10000, 150, .25, .25, .25, .25, .04, .04, .04, .04, .04, .04, .1, .1)'$.

References

C.E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics*, 2:1152–1174, 1974.

D.M. Blei, T.L. Griffiths, and M.I. Jordan. The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57:1–30, 2010.

P.J. Brown, N.D. Le, and J.V. Zidek. Inference for a covariance matrix. In P.R. Freeman and A.F.M. Smith, editors, *Aspects of Uncertainty. A Tribute to D.V. Lindley*, pages 77–92. Wiley, Chichester, 1994.

Y. Chung and D.B. Dunson. Nonparametric Bayes conditional distribution modeling with variable selection. *Journal of the American Statistical Association*, 104:1646–1660, 2009.

B.S. Clarke, E. Fokoué, and H.H. Zhang. *Principles and Theory for Data Mining and Machine Learning*. Springer Series in Statistics, New York, 2009.

G. Consonni and P. Veronese. Conditionally reducible natural exponential families and enriched conjugate priors. *Scandinavian Journal of Statistics*, 28:377–406, 2001.

D.B. Dunson. Nonparametric Bayes local partition models for random effects. *Biometrika*, 96:249–262, 2009.

D.B. Dunson and J.H. Park. Kernel stick-breaking processes. *Biometrika*, 95:307–323, 2008.

D.B. Dunson, J. Xue, and L. Carin. The matrix stick breaking process: Flexible Bayes meta analysis. *Journal of the American Statistical Association*, 103:317–327, 2008.

D.B. Dunson, S. Petrone, and L. Trippa. Partially hierarchical Dirichlet mixtures for flexible clustering and regression. 2011. Unpublished manuscript.

S. Efromovich. Conditional density estimation in a regression setting. *Annals of Statistics*, 35:2504–2535, 2007.

M.D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588, 1995.

A.E. Gelfand, A. Kottas, and S.N. MacEachern. Bayesian nonparametric spatial modeling with Dirichlet process mixing. *Journal of the American Statistical Association*, pages 1021–1035, 2005.

S. Ghosal. Dirichlet process, related priors, and posterior asymptotics. In N.L. Hjort, C. Holmes, P. Müller, and S.G. Walker, editors, *Bayesian Nonparametrics: Principles and Practice*. Cambridge University Press, 2010.

J.E. Griffin and M.F.J. Steel. Order-based dependent Dirichlet processes. *Journal of the American Statistical Association*, 10:179–194, 2006.

L.A. Hannah, D.M. Blei, and W.B. Powell. Dirichlet process mixtures of generalized linear models. *Journal of Machine Learning Research*, 12:1923–1953, 2011.

C. Kang and S. Ghosal. Clusterwise regression using Dirichlet process mixtures. In A. Sengupta, editor, *Advances in Multivariate Statistical Methods*, pages 305–325. 2009.

S.N. MacEachern. Dependent nonparametric processes. In *ASA Proceedings of the Section on Bayesian Statistical Science*, pages 50–55, Alexandria, VA, 1999. American Statistical Association.

P. Müller and F.A. Quintana. Random partition models with regression on covariates. *Journal of Statistical Planning and Inference*, 140:2801–2808, 2010.

P. Müller, A. Erkanli, and M. West. Bayesian curve fitting using multivariate normal mixtures. *Biometrika*, 88:67–79, 1996.

R.M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistcs*, 9:249–265, 2000.

A. Norets and J. Pelenis. Bayesian modeling of joint and conditional distributions. *Journal of Econometrics*, 168:332–346, 2012.

J.H. Park and D.B. Dunson. Bayesian generalized product partition model. *Statistica Sinica*, 20:1203–1226, 2010.

S. Petrone and L. Trippa. Bayesian modeling via nested random partitions. In *Proceedings of the International Conference on Complex Data Modelling and Computationally Intensive Statistical Methods*, Milan, Italy, 2009. Politecnico di Milano.

S. Petrone, M. Guindani, and A.E. Gelfand. Hybrid Dirichlet mixture models for functional data. *Journal of the Royal Statistical Society, Series B*, 71:755–782, 2009.

L. Ren, L. Du, D.B. Dunson, and L. Carin. The logistic stick-breaking process. *Journal of Machine Learning and Research*, 12:203–239, 2011.

A. Rodriguez and D.B. Dunson. Nonparametric Bayesian models through probit stick-breaking processes. *Bayesian Analysis*, 6:145–178, 2011.

A. Rodriguez, D.B. Dunson, and A.E. Gelfand. Bayesian nonparametric functional data analysis through density estimation. *Biometrika*, 96:149–162, 2009.

D.W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, Inc., Hoboken, NJ, 1992.

B. Shahbaba and R.M. Neal. Nonlinear models using Dirichlet process mixtures. *Journal of Machine Learning Research*, 10:1829–1850, 2009.

S.T. Tokdar. Adaptive convergence rates of a Dirichlet process mixture of multivariate normals. 2011. `arXiv:1111.4148 [math.ST]`.

S.K. Wade, S. Mongelluzzo, and S. Petrone. An enriched conjugate prior for Bayesian nonparametric inference. *Bayesian Analysis*, 6:359–386, 2011.

Y. Wu and S. Ghosal. Kullback Leibler property of kernel mixture priors in Bayesian density estimation. *Electronic Journal of Statistics*, 2:298–331, 2008.

Y. Wu and S. Ghosal. The L_1-consistency of Dirichlet mixtures in multivariate density estimation. *Journal of Multivariate Analysis*, 101:2411–2419, 2010.

Journal of Machine Learning Research 15 (2014) 1073-1110 Submitted 3/13; Revised 9/13; Published 3/14

Gibbs Max-margin Topic Models with Data Augmentation

Jun Zhu DCSZJ@MAIL.TSINGHUA.EDU.CN
Ning Chen NINGCHEN@MAIL.TSINGHUA.EDU.CN
Hugh Perkins NGLS11@MAILS.TSINGHUA.EDU.CN
Bo Zhang DCSZB@MAIL.TSINGHUA.EDU.CN
State Key Lab of Intelligent Technology and Systems
Tsinghua National Lab for Information Science and Technology
Department of Computer Science and Technology
Tsinghua University
Beijing, 100084, China

Editor: David Blei

Abstract

Max-margin learning is a powerful approach to building classifiers and structured output predictors. Recent work on max-margin supervised topic models has successfully integrated it with Bayesian topic models to discover discriminative latent semantic structures and make accurate predictions for unseen testing data. However, the resulting learning problems are usually hard to solve because of the non-smoothness of the margin loss. Existing approaches to building max-margin supervised topic models rely on an iterative procedure to solve multiple latent SVM subproblems with additional mean-field assumptions on the desired posterior distributions. This paper presents an alternative approach by defining a new max-margin loss. Namely, we present Gibbs max-margin supervised topic models, a latent variable Gibbs classifier to discover hidden topic representations for various tasks, including classification, regression and multi-task learning. Gibbs max-margin supervised topic models minimize an expected margin loss, which is an upper bound of the existing margin loss derived from an expected prediction rule. By introducing augmented variables and integrating out the Dirichlet variables analytically by conjugacy, we develop simple Gibbs sampling algorithms with no restrictive assumptions and no need to solve SVM subproblems. Furthermore, each step of the "augment-and-collapse" Gibbs sampling algorithms has an analytical conditional distribution, from which samples can be easily drawn. Experimental results on several medium-sized and large-scale data sets demonstrate significant improvements on time efficiency. The classification performance is also improved over competitors on binary, multi-class and multi-label classification tasks.

Keywords: supervised topic models, max-margin learning, Gibbs classifiers, regularized Bayesian inference, support vector machines

1. Introduction

As the availability and scope of complex data increase, developing statistical tools to discover latent structures and reveal hidden explanatory factors has become a major theme in statistics and machine learning. Topic models represent one type of such useful tools to discover latent semantic structures that are organized in an automatically learned latent topic space, where each topic (i.e., a coordinate of the latent space) is a unigram distribution over

the terms in a vocabulary. Due to its nice interpretability and extensibility, the Bayesian formulation of topic models (Blei et al., 2003) has motivated substantially broad extensions and applications to various fields, such as document analysis, image categorization (Fei-Fei and Perona, 2005), and network data analysis (Airoldi et al., 2008). Besides discovering latent topic representations, many models usually have a goal to make good predictions, such as relational topic models (Chang and Blei, 2009; Chen et al., 2013) whose major goal is to make accurate predictions on the link structures of a document network. Another example is supervised topic models, our focus in this paper, which learn a prediction model for regression and classification tasks. As supervising information (e.g., user-input rating scores for product reviews) gets easier to obtain on the Web, developing supervised latent topic models has attracted a lot of attention. Both maximum likelihood estimation (MLE) and max-margin learning have been applied to learn supervised topic models. Different from the MLE-based approaches (Blei and McAuliffe, 2007), which define a normalized likelihood model for response variables, max-margin supervised topic models, such as maximum entropy discrimination LDA (MedLDA) (Zhu et al., 2012), directly minimize a margin-based loss derived from an expected (or averaging) prediction rule.

By performing discriminative learning, max-margin supervised topic models can discover predictive latent topic representations and have shown promising performance in various prediction tasks, such as text document categorization (Zhu et al., 2012) and image annotation (Yang et al., 2010). However, their learning problems are generally hard to solve due to the non-smoothness of the margin-based loss function. Most existing solvers rely on a variational approximation scheme with strict mean-field assumptions on posterior distributions, and they normally need to solve multiple latent SVM subproblems in an EM-type iterative procedure. By showing a new interpretation of MedLDA as a regularized Bayesian inference method, the recent work (Jiang et al., 2012) successfully developed Monte Carlo methods for such max-margin topic models, with a weaker mean-field assumption. Though the prediction performance is improved because of more accurate inference, the Monte Carlo methods still need to solve multiple SVM subproblems. Thus, their efficiency could be limited as learning SVMs is normally computationally demanding. Furthermore, due to the dependence on SVM solvers, it is not easy to parallelize these algorithms for large-scale data analysis tasks, although substantial efforts have been made to develop parallel Monte Carlo methods for unsupervised topic models (Newman et al., 2009; Smola and Narayanamurthy, 2010; Ahmed et al., 2012).

This paper presents Gibbs MedLDA, an alternative formulation of max-margin supervised topic models, for which we can develop simple and efficient inference algorithms. Technically, instead of minimizing the margin loss of an expected (averaging) prediction rule as adopted in existing max-margin topic models, Gibbs MedLDA minimizes the expected margin loss of many latent prediction rules, of which each rule corresponds to a configuration of topic assignments and the prediction model, drawn from a post-data posterior distribution. Theoretically, the expected margin loss is an upper bound of the existing margin loss of an expected prediction rule. Computationally, although the expected margin loss can still be hard to optimize using variational algorithms, we successfully develop simple and fast Gibbs sampling algorithms without any restrictive assumptions on the posterior distribution and without solving multiple latent SVM subproblems. By introducing a set of auxiliary variables and integrating out the Dirichlet variables by conjugacy, each of the

sampling substeps has a closed-form conditional distribution, from which samples can be efficiently drawn.

Our algorithms represent an extension of the classical ideas of data augmentation (Dempster et al., 1977; Tanner and Wong, 1987; van Dyk and Meng, 2001) and its recent developments in learning fully observed max-margin classifiers (Polson and Scott, 2011) to learn the sophisticated latent topic models. On the other hand, Gibbs MedLDA represents a generalization of Gibbs (or stochastic) classifiers (McAllester, 2003; Catoni, 2007; Germain et al., 2009) to incorporate a hierarchy of latent variables for discovering latent topic representations. We further generalize the ideas to develop a Gibbs MedLDA regression model and a multi-task Gibbs MedLDA model, for which we also develop efficient "augment-and-collapse" Gibbs sampling algorithms by exploring the same ideas of data augmentation. Empirical results on real data sets demonstrate significant improvements in time efficiency. The classification performance is also significantly improved in binary, multi-class, and multi-label classification tasks.

The rest of the paper is structured as follows. Section 2 summarizes some related work. Section 3 reviews MedLDA and its EM-type algorithms. Section 4 presents Gibbs MedLDA and its sampling algorithms for classification. Section 5 presents two extensions of Gibbs MedLDA for regression and multi-task learning. Section 6 presents empirical results. Finally, Section 7 concludes and discusses future directions.

2. Related Work

Max-margin learning has been very successful in building classifiers (Vapnik, 1995) and structured output prediction models (Taskar et al., 2003) in the last decade. Recently, research on learning max-margin models in the presence of latent variable models has received increasing attention because of the promise of using latent variables to capture the underlying structures of the complex problems. Deterministic approaches (Yu and Joachims, 2009) fill in the unknown values of the hidden structures by using some estimates (e.g., MAP estimates), and then a max-margin loss function is defined with the filled-in hidden structures, while probabilistic approaches aim to infer an entire distribution profile of the hidden structures given evidence and some prior distribution, following the Bayes' way of thinking. Though the former works well in practice, we focus on Bayesian approaches, which can naturally incorporate prior beliefs, maintain the entire distribution profile of latent structures, and be extensible to nonparametric methods. One representative work along this line is maximum entropy discrimination (MED) (Jaakkola et al., 1999; Jebara, 2001), which learns a distribution of model parameters given a set of labeled training data.

MedLDA (Zhu et al., 2012) is one extension of MED to infer hidden topical structures from data and MMH (max-margin Harmoniums) (Chen et al., 2012) is another extension that infers the hidden semantic features from multi-view data. Along similar lines, recent work has also successfully developed nonparametric Bayesian max-margin models, such as infinite SVMs (iSVM) (Zhu et al., 2011) for discovering clustering structures when building SVM classifiers and infinite latent SVMs (iLSVM) (Zhu et al., 2014) for automatically learning predictive features for SVM classifiers. Both iSVM and iLSVM can automatically resolve the model complexity, for example, the number of components in a mixture model or the number of latent features in a factor analysis model. The nonparametric Bayesian max-

margin ideas have been proven to be effective in dealing with more challenging problems, such as link prediction in social networks (Zhu, 2012) and low-rank matrix factorization for collaborative recommendation (Xu et al., 2012, 2013).

One common challenge of these Bayesian max-margin latent variable models is on the posterior inference, which is normally intractable. Almost all the existing work adopts a variational approximation scheme, with some mean-field assumptions. Very little research has been done on developing Monte Carlo methods, except the work (Jiang et al., 2012) which still makes mean-field assumptions. The work in the present paper provides a novel way to formulate Bayesian max-margin models and we show that these new formulations can have very simple and efficient Monte Carlo inference algorithms without making restrictive assumptions. The key step to deriving our algorithms is a data augmentation formulation of the expected margin-based loss. Other work on inferring the posterior distributions of latent variables includes max-margin min-entropy models (Miller et al., 2012) which learn a single set of model parameters, different from our focus of inferring the model posterior distribution.

Data augmentation refers to methods of augmenting the observed data so as to make it easy to analyze with an iterative optimization or sampling algorithm. For deterministic algorithms, the technique has been popularized in the statistics community by the seminal expectation-maximization (EM) algorithm (Dempster et al., 1977) for maximum likelihood estimation (MLE) with missing values. For stochastic algorithms, the technique has been popularized in statistics by Tanner and Wong's data augmentation algorithm for posterior sampling (Tanner and Wong, 1987) and in physics by Swendsen and Wang's sampling algorithms for Ising and Potts models (Swendsen and Wang, 1987). When using the idea to solve estimation or posterior inference problems, the key step is to find a set of augmented variables, conditioned on which the distribution of our models can be easily sampled. The speed of mixing or convergence is another important concern when designing a data augmentation method. While the conflict between simplicity and speed is a common phenomenon with many standard augmentation schemes, some work has demonstrated that with more creative augmentation schemes it is possible to construct EM-type algorithms (Meng and van Dyk, 1997) or Markov Chain Monte Carlo methods (known as slice sampling) (Neal, 1997) that are both fast and simple. We refer the readers to van Dyk and Meng (2001) for an excellent review of the broad literature of data augmentation and an effective search strategy for selecting good augmentation schemes.

For our focus on max-margin classifiers, the recent work (Polson and Scott, 2011) provides an elegant data augmentation formulation for support vector machines (SVM) with fully observed input data, which leads to analytical conditional distributions that are easy to sample from and fast to mix. Our work in the present paper builds on the method of Polson et al. and presents a successful implementation of data augmentation to deal with the challenging posterior inference problems of Bayesian max-margin latent topic models. Our approach can be generalized to deal with other Bayesian max-margin latent variable models, for example, max-margin matrix factorization (Xu et al., 2013), as reviewed above.

Finally, some preliminary results were presented in a conference paper (Zhu et al., 2013a). This paper presents a full extension.

3. MedLDA

We begin with a brief overview of MedLDA and its learning algorithms, which motivate our developments of Gibbs MedLDA.

3.1 MedLDA: A Regularized Bayesian Model

We consider binary classification with a labeled training set $\mathcal{D} = \{(\mathbf{w}_d, y_d)\}_{d=1}^D$, where $\mathbf{w}_d = \{w_{dn}\}_{n=1}^{N_d}$ denotes the bag-of-words appearing in document d and the response variable Y takes values from the output space $\mathcal{Y} = \{-1, +1\}$. D is the data set size. Basically, MedLDA consists of two parts—an LDA model for describing input documents $\mathbf{W} = \{\mathbf{w}_d\}_{d=1}^D$, and an expected classifier for considering the supervising signal $\mathbf{y} = \{y_d\}_{d=1}^D$. Below, we introduce each of them in turn.

LDA: Latent Dirichlet allocation (LDA) (Blei et al., 2003) is a hierarchical Bayesian model that posits each document as an admixture of K topics, where each topic $\boldsymbol{\Phi}_k$ is a multinomial distribution over a V-word vocabulary. For document d, the generating process can be described as

1. draw a topic proportion $\boldsymbol{\theta}_d \sim \text{Dir}(\boldsymbol{\alpha})$
2. for each word n $(1 \leq n \leq N_d)$:
   (a) draw a topic assignment[1] $z_{dn}|\boldsymbol{\theta}_d \sim \text{Mult}(\boldsymbol{\theta}_d)$
   (b) draw the observed word $w_{dn}|z_{dn}, \boldsymbol{\Phi} \sim \text{Mult}(\boldsymbol{\Phi}_{z_{dn}})$

where $\text{Dir}(\cdot)$ is a Dirichlet distribution; $\text{Mult}(\cdot)$ is multinomial; and $\boldsymbol{\Phi}_{z_{dn}}$ denotes the topic selected by the non-zero entry of z_{dn}. For a fully-Bayesian LDA, the topics are random samples drawn from a prior, for example, $\boldsymbol{\Phi}_k \sim \text{Dir}(\boldsymbol{\beta})$.

Given a set of documents $\mathbf{W}$, we let $\mathbf{z}_d = \{z_{dn}\}_{n=1}^{N_d}$ denote the set of topic assignments for document d and let $\mathbf{Z} = \{\mathbf{z}_d\}_{d=1}^D$ and $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_d\}_{d=1}^D$ denote all the topic assignments and mixing proportions for the whole corpus, respectively. Then, LDA infers the posterior distribution using Bayes' rule

$$p(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}|\mathbf{W}) = \frac{p_0(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) p(\mathbf{W}|\mathbf{Z}, \boldsymbol{\Phi})}{p(\mathbf{W})},$$

where $p_0(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) = \prod_{k=1}^K p_0(\boldsymbol{\Phi}_k|\boldsymbol{\beta}) \prod_{d=1}^D p_0(\boldsymbol{\theta}_d|\boldsymbol{\alpha}) \prod_{n=1}^{N_d} p(z_{dn}|\boldsymbol{\theta}_d)$ according to the generating process of LDA; and $p(\mathbf{W})$ is the marginal evidence. We can show that the posterior distribution by Bayes' rule is the solution of an information theoretical optimization problem

$$\begin{aligned} \min_{q(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})} \; & \text{KL}\left[q(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) \| p_0(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right] - \mathbb{E}_q\left[\log p(\mathbf{W}|\mathbf{Z}, \boldsymbol{\Phi})\right] \\ \text{s.t.} : \; & q(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) \in \mathcal{P}, \end{aligned} \tag{1}$$

where $\text{KL}(q||p)$ is the Kullback-Leibler divergence and $\mathcal{P}$ is the space of probability distributions with an appropriate dimension. In fact, if we add the constant $\log p(\mathbf{W})$ to the objective, the problem is the minimization of the KL-divergence $\text{KL}(q(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) \| p(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}|\mathbf{W}))$,

1. z_{dn} is a K-dimensional binary vector with only one nonzero entry.

whose solution is the desired posterior distribution by Bayes' rule. One advantage of this variational formulation of Bayesian inference is that it can be naturally extended to include some regularization terms on the desired post-data posterior distribution q. This insight has been taken to develop regularized Bayesian inference (RegBayes) (Zhu et al., 2014), a computational framework for doing Bayesian inference with posterior regularization.[2] As shown in Jiang et al. (2012) and detailed below, MedLDA is one example of RegBayes models. Moreover, as we shall see in Section 4, our Gibbs max-margin topic models follow this similar idea too.

Expected Classifier: Given a training set $\mathcal{D}$, an expected (or averaging) classifier chooses a posterior distribution $q(h|\mathcal{D})$ over a hypothesis space $\mathcal{H}$ of classifiers such that the q-weighted (expected) classifier

$$h_q(\mathbf{w}) = \text{sign } \mathbb{E}_q[h(\mathbf{w})]$$

will have the smallest possible risk. MedLDA follows this principle to learn a posterior distribution $q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}|\mathcal{D})$ such that the expected classifier

$$\hat{y} = \text{sign } F(\mathbf{w}) \tag{2}$$

has the smallest possible risk, approximated by the training error $\mathcal{R}_{\mathcal{D}}(q) = \sum_{d=1}^{D} \mathbb{I}(\hat{y}_d \neq y_d)$. The discriminant function is defined as

$$\begin{aligned} F(\mathbf{w}) &= \mathbb{E}_{q(\boldsymbol{\eta}, \mathbf{z}|\mathcal{D})}[F(\boldsymbol{\eta}, \mathbf{z}; \mathbf{w})], \\ F(\boldsymbol{\eta}, \mathbf{z}; \mathbf{w}) &= \boldsymbol{\eta}^\top \bar{\mathbf{z}}, \end{aligned}$$

where $\bar{\mathbf{z}}$ is the average topic assignment associated with the words $\mathbf{w}$, a vector with element $\bar{z}_k = \frac{1}{N}\sum_{n=1}^{N} z_n^k$, and $\boldsymbol{\eta}$ is the classifier weights. Note that the expected classifier and the LDA likelihood are coupled via the latent topic assignments $\mathbf{Z}$. The strong coupling makes it possible for MedLDA to learn a posterior distribution that can describe the observed words well and make accurate predictions.

Regularized Bayesian Inference: To integrate the above two components for hybrid learning, MedLDA regularizes the properties of the topic representations by imposing the following max-margin constraints derived from the classifier (2) to a standard LDA inference problem (1)

$$y_d F(\mathbf{w}_d) \geq \ell - \xi_d, \ \forall d,$$

where ℓ (≥ 1) is the cost of making a wrong prediction; and $\boldsymbol{\xi} = \{\xi_d\}_{d=1}^{D}$ are non-negative slack variables for inseparable cases. Let $\mathcal{L}(q) = \text{KL}(q||p_0(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})) - \mathbb{E}_q[\log p(\mathbf{W}|\mathbf{Z}, \boldsymbol{\Phi})]$ be the objective for doing standard Bayesian inference with the classifier $\boldsymbol{\eta}$ and $p_0(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) = p_0(\boldsymbol{\eta})p_0(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})$. MedLDA solves the regularized Bayesian inference (Zhu et al., 2014) problem

$$\begin{aligned} \min_{q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) \in \mathcal{P}, \boldsymbol{\xi}} \ & \mathcal{L}\left(q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right) + 2c\sum_{d=1}^{D} \xi_d \\ \text{s.t.:} \ & y_d F(\mathbf{w}_d) \geq \ell - \xi_d, \ \xi_d \geq 0, \forall d, \end{aligned} \tag{3}$$

2. Posterior regularization was first used in Ganchev et al. (2010) for maximum likelihood estimation and was later extended in Zhu et al. (2014) to Bayesian and nonparametric Bayesian methods.

where the margin constraints directly regularize the properties of the post-data distribution and c is the positive regularization parameter. Equivalently, MedLDA solves the unconstrained problem[3]

$$\min_{q(\boldsymbol{\eta},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi})} \mathcal{L}\left(q(\boldsymbol{\eta},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi})\right) + 2c\mathcal{R}\left(q(\boldsymbol{\eta},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi})\right), \tag{4}$$

where $\mathcal{R}(q) = \sum_{d=1}^{D} \max(0, \ell - y_d F(\mathbf{w}_d))$ is the hinge loss that upper-bounds the training error $\mathcal{R}_{\mathcal{D}}(q)$ of the expected classifier (2). Note that the constant 2 is included simply for convenience.

3.2 Existing Iterative Algorithms

Since it is difficult to solve problem (3) or (4) directly because of the non-conjugacy (between priors and likelihood) and the max-margin constraints, corresponding to a non-smooth posterior regularization term in (4), both variational and Monte Carlo methods have been developed for approximate solutions. It can be shown that the variational method (Zhu et al., 2012) is a coordinate descent algorithm to solve problem (4) with the fully-factorized assumption that

$$q(\boldsymbol{\eta}, \Theta, \mathbf{Z}, \boldsymbol{\Phi}) = q(\boldsymbol{\eta}) \left(\prod_{d=1}^{D} q(\boldsymbol{\theta}_d) \prod_{n=1}^{N_d} q(z_{dn}) \right) \prod_{k=1}^{K} q(\boldsymbol{\Phi}_k);$$

while the Monte Carlo methods (Jiang et al., 2012) make a weaker assumption that

$$q(\boldsymbol{\eta}, \Theta, \mathbf{Z}, \boldsymbol{\Phi}) = q(\boldsymbol{\eta}) q(\Theta, \mathbf{Z}, \boldsymbol{\Phi}).$$

All these methods have a similar EM-type iterative procedure, which solves many latent SVM subproblems, as outlined below.

Estimate $q(\boldsymbol{\eta})$: Given $q(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})$, we solve problem (4) with respect to $q(\boldsymbol{\eta})$. In the equivalent constrained form, this step solves

$$\begin{aligned} \min_{q(\boldsymbol{\eta}),\boldsymbol{\xi}} \quad & \mathrm{KL}\left(q(\boldsymbol{\eta}) \| p_0(\boldsymbol{\eta})\right) + 2c \sum_{d=1}^{D} \xi_d \\ \text{s.t.}: \quad & y_d \mathbb{E}_q[\boldsymbol{\eta}]^\top \mathbb{E}_q[\bar{\mathbf{z}}_d] \geq \ell - \xi_d,\ \xi_d \geq 0, \forall d. \end{aligned} \tag{5}$$

This problem is convex and can be solved with Lagrangian methods. Specifically, let μ_d be the Lagrange multipliers, one per constraint. When the prior $p_0(\boldsymbol{\eta})$ is the commonly used standard normal distribution, we have the optimum solution $q(\boldsymbol{\eta}) = \mathcal{N}(\boldsymbol{\kappa}, I)$, where $\boldsymbol{\kappa} = \sum_{d=1}^{D} y_d \mu_d \mathbb{E}_q[\bar{\mathbf{z}}_d]$. It can be shown that the dual problem of (5) is the dual of a standard binary linear SVM and we can solve it or its primal form efficiently using existing high-performance SVM learners. We denote the optimum solution of this problem by $(q^*(\boldsymbol{\eta}), \boldsymbol{\kappa}^*, \boldsymbol{\xi}^*, \boldsymbol{\mu}^*)$.

3. If not specified, q is subject to the constraint $q \in \mathcal{P}$.

Estimate $q(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})$: Given $q(\boldsymbol{\eta})$, we solve problem (4) with respect to $q(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})$. In the constrained form, this step solves

$$\begin{aligned} \min_{q(\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi}),\boldsymbol{\xi}} \quad & \mathcal{L}\left(q(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right) + 2c\sum_{d=1}^{D} \xi_d \\ \text{s.t.}: \quad & y_d(\boldsymbol{\kappa}^*)^\top \mathbb{E}_q[\bar{\mathbf{z}}_d] \geq \ell - \xi_d,\ \xi_d \geq 0, \forall d. \end{aligned} \tag{6}$$

Although we can solve this problem using Lagrangian methods, it would be hard to derive the dual objective. An effective approximation strategy was used in Zhu et al. (2012) and Jiang et al. (2012), which updates $q(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})$ for only one step with $\boldsymbol{\xi}$ fixed at $\boldsymbol{\xi}^*$. By fixing $\boldsymbol{\xi}$ at $\boldsymbol{\xi}^*$, we have the solution

$$q(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) \propto p(\mathbf{W}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) \exp\left\{(\boldsymbol{\kappa}^*)^\top \sum_d \mu_d^* \bar{\mathbf{z}}_d\right\},$$

where the second term indicates the regularization effects due to the max-margin posterior constraints. For those data with non-zero Lagrange multipliers (i.e., support vectors), the second term will bias MedLDA towards a new posterior distribution that favors more discriminative representations on these "hard" data points. The Monte Carlo methods (Jiang et al., 2012) directly draw samples from the posterior distribution $q(\boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})$ or its collapsed form using Gibbs sampling to estimate $\mathbb{E}_q[\bar{\mathbf{z}}_d]$, the expectations required to learn $q(\boldsymbol{\eta})$. In contrast, the variational methods (Zhu et al., 2012) solve problem (6) using coordinate descent to estimate $\mathbb{E}_q[\bar{\mathbf{z}}_d]$ with a fully factorized assumption.

4. Gibbs MedLDA

Now, we present Gibbs max-margin topic models for binary classification and their "augment-and-collapse" sampling algorithms. We will discuss further extensions in the next section.

4.1 Learning with an Expected Margin Loss

As stated above, MedLDA chooses the strategy to minimize the hinge loss of an expected classifier. In learning theory, an alternative approach to building classifiers with a posterior distribution of models is to minimize an expected loss, under the framework known as Gibbs classifiers (or stochastic classifiers) (McAllester, 2003; Catoni, 2007; Germain et al., 2009) which have nice theoretical properties on generalization performance.

For our case of inferring the distribution of latent topic assignments $\mathbf{Z} = \{\mathbf{z}_d\}_{d=1}^D$ and the classification model $\boldsymbol{\eta}$, the expected margin loss is defined as follows. If we have drawn a sample of the topic assignments $\mathbf{Z}$ and the prediction model $\boldsymbol{\eta}$ from a posterior distribution $q(\boldsymbol{\eta}, \mathbf{Z})$, we can define the linear discriminant function

$$F(\boldsymbol{\eta}, \mathbf{z}; \mathbf{w}) = \boldsymbol{\eta}^\top \bar{\mathbf{z}}$$

as before and make prediction using the *latent prediction rule*

$$\hat{y}(\boldsymbol{\eta}, \mathbf{z}) = \text{sign } F(\boldsymbol{\eta}, \mathbf{z}; \mathbf{w}). \tag{7}$$

Note that the prediction is a function of the configuration $(\boldsymbol{\eta}, \mathbf{z})$. Let $\zeta_d = \ell - y_d \boldsymbol{\eta}^\top \bar{\mathbf{z}}_d$, where ℓ is a cost parameter as defined before. The hinge loss of the stochastic classifier is

$$\mathcal{R}(\boldsymbol{\eta}, \mathbf{Z}) = \sum_{d=1}^{D} \max(0, \zeta_d),$$

a function of the latent variables $(\boldsymbol{\eta}, \mathbf{Z})$, and the expected hinge loss is

$$\mathcal{R}'(q) = \mathbb{E}_q[\mathcal{R}(\boldsymbol{\eta}, \mathbf{Z})] = \sum_{d=1}^{D} \mathbb{E}_q \left[\max(0, \zeta_d)\right],$$

a functional of the posterior distribution $q(\boldsymbol{\eta}, \mathbf{Z})$. Since for any $(\boldsymbol{\eta}, \mathbf{Z})$, the hinge loss $\mathcal{R}(\boldsymbol{\eta}, \mathbf{Z})$ is an upper bound of the training error of the latent Gibbs classifier (7), that is,

$$\mathcal{R}(\boldsymbol{\eta}, \mathbf{Z}) \geq \sum_{d=1}^{D} \mathbb{I}\left(y_d \neq \hat{y}_d(\boldsymbol{\eta}, \mathbf{z}_d)\right),$$

we have

$$\mathcal{R}'(q) \geq \sum_{d=1}^{D} \mathbb{E}_q \left[\mathbb{I}(y_d \neq \hat{y}_d(\boldsymbol{\eta}, \mathbf{z}_d))\right],$$

where $\mathbb{I}(\cdot)$ is an indicator function that equals to 1 if the predicate holds otherwise 0. In other words, the expected hinge loss $\mathcal{R}'(q)$ is an upper bound of the expected training error of the Gibbs classifier (7). Thus, it is a good surrogate loss for learning a posterior distribution which could lead to a low training error in expectation.

Then, with the same goal as MedLDA of finding a posterior distribution $q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})$ that on one hand describes the observed data and on the other hand predicts as well as possible on training data, we define Gibbs MedLDA as solving the new regularized Bayesian inference problem

$$\min_{q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})} \mathcal{L}\left(q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right) + 2c\mathcal{R}'\left(q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right). \tag{8}$$

Note that we have written the expected margin loss $\mathcal{R}'$ as a function of the complete distribution $q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})$. This does not conflict with our definition of $\mathcal{R}'$ as a function of the marginal distribution $q(\boldsymbol{\eta}, \mathbf{Z})$ because the other irrelevant variables (i.e., $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$) are integrated out when we compute the expectation.

Comparing to MedLDA in problem (4), we have the following lemma by applying Jensen's inequality.

Lemma 1 *The expected hinge loss $\mathcal{R}'$ is an upper bound of the hinge loss of the expected classifier (2):*

$$\mathcal{R}'(q) \geq \mathcal{R}(q) = \sum_{d=1}^{D} \max\left(0, \mathbb{E}_q[\zeta_d]\right);$$

and thus the objective in (8) is an upper bound of that in (4) when c values are the same.

4.2 Formulation with Data Augmentation

If we directly solve problem (8), the expected hinge loss $\mathcal{R}'$ is hard to deal with because of the non-differentiable max function. Fortunately, we can develop a simple collapsed Gibbs sampling algorithm with analytical forms of local conditional distributions, based on a data augmentation formulation of the expected hinge-loss.

Let $\phi(y_d|\mathbf{z}_d, \boldsymbol{\eta}) = \exp\{-2c\max(0, \zeta_d)\}$ be the unnormalized likelihood of the response variable for document d. Then, problem (8) can be written as

$$\min_{q(\boldsymbol{\eta},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi})} \mathcal{L}\left(q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right) - \mathbb{E}_q\left[\log \phi(\mathbf{y}|\mathbf{Z}, \boldsymbol{\eta})\right], \tag{9}$$

where $\phi(\mathbf{y}|\mathbf{Z}, \boldsymbol{\eta}) = \prod_{d=1}^{D} \phi(y_d|\mathbf{z}_d, \boldsymbol{\eta})$. Solving problem (9) with the constraint that $q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) \in \mathcal{P}$, we can get the normalized posterior distribution

$$q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) = \frac{p_0(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) p(\mathbf{W}|\mathbf{Z}, \boldsymbol{\Phi}) \phi(\mathbf{y}|\mathbf{Z}, \boldsymbol{\eta})}{\psi(\mathbf{y}, \mathbf{W})},$$

where $\psi(\mathbf{y}, \mathbf{W})$ is the normalization constant that depends on the observed data only. Due to the complicated form of ϕ, it will not have simple conditional distributions if we want to derive a Gibbs sampling algorithm for $q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})$ directly. This motivates our exploration of data augmentation techniques. Specifically, using the ideas of data augmentation (Tanner and Wong, 1987; Polson and Scott, 2011), we have Lemma 2.

Lemma 2 (Scale Mixture Representation) *The unnormalized likelihood can be expressed as*

$$\phi(y_d|\mathbf{z}_d, \boldsymbol{\eta}) = \int_0^\infty \frac{1}{\sqrt{2\pi\lambda_d}} \exp\left(-\frac{(\lambda_d + c\zeta_d)^2}{2\lambda_d}\right) \mathrm{d}\lambda_d.$$

Proof Due to the fact that $a\max(0, x) = \max(0, ax)$ if $a \geq 0$, we have $-2c\max(0, \zeta_d) = -2\max(0, c\zeta_d)$. Then, we can follow the proof in Polson and Scott (2011) to get the results. ■

Lemma 2 indicates that the posterior distribution of Gibbs MedLDA can be expressed as the marginal of a higher-dimensional distribution that includes the augmented variables $\boldsymbol{\lambda} = \{\lambda_d\}_{d=1}^D$, that is,

$$q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) = \int_0^\infty \cdots \int_0^\infty q(\boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) \mathrm{d}\lambda_1 \cdots \mathrm{d}\lambda_D = \int_{\mathbb{R}_+^D} q(\boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) \mathrm{d}\boldsymbol{\lambda},$$

where $\mathbb{R}_+ = \{x : x \in \mathbb{R},\ x > 0\}$ is the set of positive real numbers; the complete posterior distribution is

$$q(\boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) = \frac{p_0(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi}) p(\mathbf{W}|\mathbf{Z}, \boldsymbol{\Phi}) \phi(\mathbf{y}, \boldsymbol{\lambda}|\mathbf{Z}, \boldsymbol{\eta})}{\psi(\mathbf{y}, \mathbf{W})};$$

and the unnormalized joint distribution of $\mathbf{y}$ and $\boldsymbol{\lambda}$ is

$$\phi(\mathbf{y}, \boldsymbol{\lambda}|\mathbf{Z}, \boldsymbol{\eta}) = \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\lambda_d}} \exp\left(-\frac{(\lambda_d + c\zeta_d)^2}{2\lambda_d}\right).$$

In fact, we can show that the complete posterior distribution is the solution of the data augmentation problem of Gibbs MedLDA

$$\min_{q(\boldsymbol{\eta},\boldsymbol{\lambda},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi})} \mathcal{L}\left(q(\boldsymbol{\eta},\boldsymbol{\lambda},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi})\right) - \mathbb{E}_q\left[\log \phi(\mathbf{y},\boldsymbol{\lambda}|\mathbf{Z},\boldsymbol{\eta})\right],$$

which is again subject to the normalization constraint that $q(\boldsymbol{\eta},\boldsymbol{\lambda},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi}) \in \mathcal{P}$. The first term in the objective is $\mathcal{L}(q(\boldsymbol{\eta},\boldsymbol{\lambda},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi})) = \mathrm{KL}(q(\boldsymbol{\eta},\boldsymbol{\lambda},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi})||p_0(\boldsymbol{\eta},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi})p_0(\boldsymbol{\lambda})) - \mathbb{E}_q[\log p(\mathbf{W}|\mathbf{Z},\boldsymbol{\Phi})]$, where a prior distribution is imposed on the augmented variables $\boldsymbol{\lambda}$. One good choice of $p_0(\boldsymbol{\lambda})$ is the noninformative uniform prior.

Remark 3 *The objective of this augmented problem is an upper bound of the objective in (9) (thus, also an upper bound of MedLDA's objective due to Lemma 1). This is because by using the data augmentation we can show that*

$$\begin{aligned}
\mathbb{E}_{q(\mathbf{V})}[\log \phi(\mathbf{y}|\mathbf{Z},\boldsymbol{\eta})] =& \mathbb{E}_{q(\mathbf{V})}\left[\log \int_{\mathbb{R}_+^D} \phi(\mathbf{y},\boldsymbol{\lambda}|\mathbf{Z},\boldsymbol{\eta})\mathrm{d}\boldsymbol{\lambda}\right] \\
=& \mathbb{E}_{q(\mathbf{V})}\left[\log \int_{\mathbb{R}_+^D} \frac{q(\boldsymbol{\lambda}|\mathbf{V})}{q(\boldsymbol{\lambda}|\mathbf{V})}\phi(\mathbf{y},\boldsymbol{\lambda}|\mathbf{Z},\boldsymbol{\eta})\mathrm{d}\boldsymbol{\lambda}\right] \\
\geq& \mathbb{E}_{q(\mathbf{V})}\left[\mathbb{E}_{q(\boldsymbol{\lambda}|\mathbf{V})}\left[\log \phi(\mathbf{y},\boldsymbol{\lambda}|\mathbf{Z},\boldsymbol{\eta})\right] - \mathbb{E}_{q(\boldsymbol{\lambda}|\mathbf{V})}\left[\log q(\boldsymbol{\lambda}|\mathbf{V})\right]\right] \\
=& \mathbb{E}_{q(\mathbf{V},\boldsymbol{\lambda})}\left[\log \phi(\mathbf{y},\boldsymbol{\lambda}|\mathbf{Z},\boldsymbol{\eta})\right] - \mathbb{E}_{q(\mathbf{V},\boldsymbol{\lambda})}\left[\log q(\boldsymbol{\lambda}|\mathbf{V})\right],
\end{aligned}$$

where $\mathbf{V} = \{\boldsymbol{\eta},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi}\}$ *denotes all the random variables in MedLDA. Therefore, we have*

$$\begin{aligned}
\mathcal{L}(q(\mathbf{V})) - \mathbb{E}_{q(\mathbf{V})}[\log \phi(\mathbf{y}|\mathbf{Z},\boldsymbol{\eta})] \leq& \mathcal{L}(q(\mathbf{V})) - \mathbb{E}_{q(\mathbf{V},\boldsymbol{\lambda})}\left[\log \phi(\mathbf{y},\boldsymbol{\lambda}|\mathbf{Z},\boldsymbol{\eta})\right] + \mathbb{E}_{q(\mathbf{V},\boldsymbol{\lambda})}\left[\log q(\boldsymbol{\lambda}|\mathbf{V})\right] \\
=& \mathcal{L}(q(\mathbf{V},\boldsymbol{\lambda})) - \mathbb{E}_q[\log \phi(\mathbf{y},\boldsymbol{\lambda}|\mathbf{Z},\boldsymbol{\eta})].
\end{aligned}$$

4.3 Inference with Collapsed Gibbs Sampling

Although with the above data augmentation formulation we can do Gibbs sampling to infer the complete posterior distribution $q(\boldsymbol{\eta},\boldsymbol{\lambda},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi})$ and thus $q(\boldsymbol{\eta},\boldsymbol{\Theta},\mathbf{Z},\boldsymbol{\Phi})$ by ignoring $\boldsymbol{\lambda}$, the mixing speed would be slow because of the large sample space of the latent variables. One way to effectively reduce the sample space and improve mixing rates is to integrate out the intermediate Dirichlet variables $(\boldsymbol{\Theta},\boldsymbol{\Phi})$ and build a Markov chain whose equilibrium distribution is the resulting marginal distribution $q(\boldsymbol{\eta},\boldsymbol{\lambda},\mathbf{Z})$. We propose to use collapsed Gibbs sampling, which has been successfully used in LDA (Griffiths and Steyvers, 2004). With the data augmentation representation, this leads to our "augment-and-collapse" sampling algorithm for Gibbs MedLDA, as detailed below.

For the data augmented formulation of Gibbs MedLDA, by integrating out the Dirichlet variables $(\boldsymbol{\Theta},\boldsymbol{\Phi})$, we get the collapsed posterior distribution:

$$\begin{aligned}
q(\boldsymbol{\eta},\boldsymbol{\lambda},\mathbf{Z}) \propto& p_0(\boldsymbol{\eta})p(\mathbf{W},\mathbf{Z}|\boldsymbol{\alpha},\boldsymbol{\beta})\phi(\mathbf{y},\boldsymbol{\lambda}|\mathbf{Z},\boldsymbol{\eta}) \\
=& p_0(\boldsymbol{\eta})\left[\prod_{d=1}^{D}\frac{\delta(\mathbf{C}_d+\boldsymbol{\alpha})}{\delta(\boldsymbol{\alpha})}\right]\prod_{k=1}^{K}\frac{\delta(\mathbf{C}_k+\boldsymbol{\beta})}{\delta(\boldsymbol{\beta})}\prod_{d=1}^{D}\frac{1}{\sqrt{2\pi\lambda_d}}\exp\left(-\frac{(\lambda_d+c\zeta_d)^2}{2\lambda_d}\right),
\end{aligned}$$

where

$$\delta(\mathbf{x}) = \frac{\prod_{i=1}^{\dim(\mathbf{x})}\Gamma(x_i)}{\Gamma(\sum_{i=1}^{\dim(\mathbf{x})}x_i)};$$

$\Gamma(\cdot)$ is the Gamma function; C_k^t is the number of times that the term t is assigned to topic k over the whole corpus; $\mathbf{C}_k = \{C_k^t\}_{t=1}^V$ is the set of word counts associated with topic k; C_d^k is the number of times that terms are associated with topic k within the d-th document; and $\mathbf{C}_d = \{C_d^k\}_{k=1}^K$ is the set of topic counts for document d. Then, the conditional distributions used in collapsed Gibbs sampling are as follows.

For $\boldsymbol{\eta}$: Let us assume its prior is the commonly used isotropic Gaussian distribution $p_0(\boldsymbol{\eta}) = \prod_{k=1}^K \mathcal{N}(\eta_k; 0, \nu^2)$, where ν is a non-zero parameter. Then, we have the conditional distribution of $\boldsymbol{\eta}$ given the other variables:

$$\begin{aligned} q(\boldsymbol{\eta}|\mathbf{Z}, \boldsymbol{\lambda}) &\propto p_0(\boldsymbol{\eta}) \prod_{d=1}^D \exp\left(-\frac{(\lambda_d + c\zeta_d)^2}{2\lambda_d}\right) \\ &\propto \exp\left(-\sum_{k=1}^K \frac{\eta_k^2}{2\nu^2} - \sum_{d=1}^D \frac{(\lambda_d + c\zeta_d)^2}{2\lambda_d}\right) \\ &= \exp\left(-\frac{1}{2}\boldsymbol{\eta}^\top \left(\frac{1}{\nu^2} I + c^2 \sum_{d=1}^D \frac{\bar{\mathbf{z}}_d \bar{\mathbf{z}}_d^\top}{\lambda_d}\right)\boldsymbol{\eta} + \left(c\sum_{d=1}^D y_d \frac{\lambda_d + c\ell}{\lambda_d}\bar{\mathbf{z}}_d\right)^\top \boldsymbol{\eta}\right) \\ &= \mathcal{N}(\boldsymbol{\eta}; \boldsymbol{\mu}, \boldsymbol{\Sigma}), \end{aligned} \tag{10}$$

a K-dimensional Gaussian distribution, where the posterior mean and the covariance matrix are

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\left(c\sum_{d=1}^D y_d \frac{\lambda_d + c\ell}{\lambda_d}\bar{\mathbf{z}}_d\right), \text{ and } \boldsymbol{\Sigma} = \left(\frac{1}{\nu^2} I + c^2 \sum_{d=1}^D \frac{\bar{\mathbf{z}}_d \bar{\mathbf{z}}_d^\top}{\lambda_d}\right)^{-1}.$$

Therefore, we can easily draw a sample from this multivariate Gaussian distribution. The inverse can be robustly done using Cholesky decomposition, an $O(K^3)$ procedure. Since K is normally not large, the inversion can be done efficiently, especially in applications where the number of documents is much larger than the number of topics.

For $\mathbf{Z}$: The conditional distribution of $\mathbf{Z}$ given the other variables is

$$q(\mathbf{Z}|\boldsymbol{\eta}, \boldsymbol{\lambda}) \propto \prod_{d=1}^D \frac{\delta(\mathbf{C}_d + \boldsymbol{\alpha})}{\delta(\boldsymbol{\alpha})} \exp\left(-\frac{(\lambda_d + c\zeta_d)^2}{2\lambda_d}\right) \prod_{k=1}^K \frac{\delta(\mathbf{C}_k + \boldsymbol{\beta})}{\delta(\boldsymbol{\beta})}.$$

By canceling common factors, we can derive the conditional distribution of one variable z_{dn} given others $\mathbf{Z}_\neg$ as:

$$\begin{aligned} q(z_{dn}^k = 1|\mathbf{Z}_\neg, \boldsymbol{\eta}, \boldsymbol{\lambda}, w_{dn} = t) \propto \frac{(C_{k,\neg n}^t + \beta_t)(C_{d,\neg n}^k + \alpha_k)}{\sum_{t=1}^V C_{k,\neg n}^t + \sum_{t=1}^V \beta_t} \exp\Big(&\frac{\gamma y_d (c\ell + \lambda_d)\eta_k}{\lambda_d} \\ &- c^2 \frac{\gamma^2 \eta_k^2 + 2\gamma(1-\gamma)\eta_k \Lambda_{dn}^k}{2\lambda_d}\Big), \end{aligned} \tag{11}$$

where $C_{\cdot,\neg n}^{\cdot}$ indicates that term n is excluded from the corresponding document or topic; $\gamma = \frac{1}{N_d}$; and

$$\Lambda_{dn}^k = \frac{1}{N_d - 1}\sum_{k'=1}^K \eta_{k'} C_{d,\neg n}^{k'}$$

Algorithm 1 collapsed Gibbs sampling algorithm for Gibbs MedLDA classification models

Initialization: set $\boldsymbol{\lambda} = 1$ and randomly draw z_{dk} from a uniform distribution.
for $m = 1$ **to** M **do**
  draw the classifier from the normal distribution (10)
  for $d = 1$ **to** D **do**
    for each word n in document d **do**
      draw a topic from the multinomial distribution (11)
    end for
    draw λ_d^{-1} (and thus λ_d) from the inverse Gaussian distribution (12).
  end for
end for

is the discriminant function value without word n. We can see that the first term is from the LDA model for observed word counts and the second term is from the supervised signal $\mathbf{y}$.

For $\boldsymbol{\lambda}$: Finally, the conditional distribution of the augmented variables $\boldsymbol{\lambda}$ given the other variables is factorized and we can derive the conditional distribution for each λ_d as

$$\begin{aligned} q(\lambda_d|\mathbf{Z}, \boldsymbol{\eta}) &\propto \frac{1}{\sqrt{2\pi\lambda_d}} \exp\left(-\frac{(\lambda_d + c\zeta_d)^2}{2\lambda_d}\right) \\ &\propto \frac{1}{\sqrt{2\pi\lambda_d}} \exp\left(-\frac{c^2\zeta_d^2}{2\lambda_d} - \frac{\lambda_d}{2}\right) \\ &= \mathcal{GIG}\left(\lambda_d; \frac{1}{2}, 1, c^2\zeta_d^2\right), \end{aligned}$$

where

$$\mathcal{GIG}(x; p, a, b) = C(p, a, b)x^{p-1} \exp\left(-\frac{1}{2}\left(\frac{b}{x} + ax\right)\right)$$

is a generalized inverse Gaussian distribution (Devroye, 1986) and $C(p, a, b)$ is a normalization constant. Therefore, we can derive that λ_d^{-1} follows an inverse Gaussian distribution

$$p(\lambda_d^{-1}|\mathbf{Z}, \boldsymbol{\eta}) = \mathcal{IG}\left(\lambda_d^{-1}; \frac{1}{c|\zeta_d|}, 1\right), \tag{12}$$

where

$$\mathcal{IG}(x; a, b) = \sqrt{\frac{b}{2\pi x^3}} \exp\left(-\frac{b(x-a)^2}{2a^2x}\right)$$

for $a > 0$ and $b > 0$.

With the above conditional distributions, we can construct a Markov chain which iteratively draws samples of the classifier weights $\boldsymbol{\eta}$ using Equation (10), the topic assignments $\mathbf{Z}$ using Equation (11) and the augmented variables $\boldsymbol{\lambda}$ using Equation (12), with an initial condition. To sample from an inverse Gaussian distribution, we apply the transformation method with multiple roots (Michael et al., 1976) which is very efficient with a constant time complexity. Overall, the per-iteration time complexity is $\mathcal{O}(K^3 + N_{total}K)$, where $N_{total} = \sum_{d=1}^{D} N_d$ is the total number of words in all documents. If K is not very large

(e.g., $K \ll \sqrt{N_{total}}$), which is the common case in practice as N_{total} is often very large, the per-iteration time complexity is $\mathcal{O}(N_{total}K)$; if K is large (e.g., $K \gg \sqrt{N_{total}}$), which is not common in practice, drawing the global classifier weights will dominate and the per-iteration time complexity is $\mathcal{O}(K^3)$. In our experiments, we initially set $\lambda_d = 1,\ \forall d$ and randomly draw $\mathbf{Z}$ from a uniform distribution. In training, we run this Markov chain to finish the burn-in stage with M iterations, as outlined in Algorithm 1. Then, we draw a sample $\hat{\boldsymbol{\eta}}$ as the Gibbs classifier to make predictions on testing data.

In general, there is no theoretical guarantee that a Markov chain constructed using data augmentation can converge to the target distribution; see Hobert (2011) for a failure example. However, for our algorithms, we can justify that the Markov transition distribution of the chain satisfies the condition $\mathcal{K}$ from Hobert (2011), that is, the transition probability from one state to any other state is larger than 0. Condition $\mathcal{K}$ implies that the Markov chain is Harris ergodic (Tan, 2009, Lemma 1). Therefore, no matter how the chain is started, our sampling algorithms can be employed to effectively explore the intractable posterior distribution. In practice, the sampling algorithm as well as the ones to be presented require only a few iterations to get stable prediction performance, as we shall see in Section 6.5.1. More theoretical analysis such as convergence rates requires a good bit of technical Markov chain theory and is our future work.

4.4 Prediction

To apply the Gibbs classifier $\hat{\boldsymbol{\eta}}$, we need to infer the topic assignments for testing document, denoted by $\mathbf{w}$. A fully Bayesian treatment needs to compute an integral in order to get the posterior distribution of the topic assignment given the training data $\mathcal{D}$ and the testing document content $\mathbf{w}$:

$$p(\mathbf{z}|\mathbf{w}, \mathcal{D}) \propto \int_{\mathcal{P}_V^K} p(\mathbf{z}, \mathbf{w}, \Phi|\mathcal{D})\mathrm{d}\Phi = \int_{\mathcal{P}_V^K} p(\mathbf{z}, \mathbf{w}|\Phi)p(\Phi|\mathcal{D})\mathrm{d}\Phi,$$

where $\mathcal{P}_V$ is the $V-1$ dimensional simplex; and the second equality holds due to the conditional independence assumption of the documents given the topics. Various approximation methods can be applied to compute the integral. Here, we take the approach applied in Zhu et al. (2012), which uses a point estimate of topics $\boldsymbol{\Phi}$ from training data and makes predictions based on them. Specifically, we use a point estimate $\hat{\boldsymbol{\Phi}}$ (a Dirac measure) to approximate the probability distribution $p(\boldsymbol{\Phi}|\mathcal{D})$. For the collapsed Gibbs sampler, an estimate of $\hat{\boldsymbol{\Phi}}$ using the samples is the posterior mean

$$\hat{\phi}_{kt} \propto C_k^t + \beta_t.$$

Then, given a testing document $\mathbf{w}$, we infer its latent components $\mathbf{z}$ using $\hat{\boldsymbol{\Phi}}$ by drawing samples from the local conditional distribution

$$p(z_n^k = 1|\mathbf{z}_{\neg n}, \mathbf{w}, \mathcal{D}) \propto \hat{\phi}_{kw_n}\left(C_{\neg n}^k + \alpha_k\right), \tag{13}$$

where $C_{\neg n}^k$ is the number of times that the terms in this document $\mathbf{w}$ assigned to topic k with the n-th term excluded. To start the sampler, we randomly set each word to one topic. Then, we run the Gibbs sampler for a few iterations until some stop criterion is satisfied,

for example, after a few burn-in steps or the relative change of data likelihood is lower than some threshold. Here, we adopt the latter, the same as in Jiang et al. (2012). After this burn-in stage, we keep one sample of $\mathbf{z}$ for prediction using the stochastic classifier. Empirically, using the average of a few (e.g., 10) samples of $\mathbf{z}$ could lead to slightly more robust predictions, as we shall see in Section 6.5.4.

5. Extensions to Regression and Multi-task Learning

The above ideas can be naturally generalized to develop Gibbs max-margin supervised topic models for various prediction tasks. In this section, we present two examples for regression and multi-task learning, respectively.

5.1 Gibbs MedLDA Regression Model

We first discuss how to generalize the above ideas to develop a regression model, where the response variable Y takes real values. Formally, the Gibbs MedLDA regression model also has two components—an LDA model to describe input bag-of-words documents and a Gibbs regression model for the response variables. Since the LDA component is the same as in the classification model, we focus on presenting the Gibbs regression model.

5.1.1 The Models with Data Augmentation

If a sample of the topic assignments $\mathbf{Z}$ and the prediction model $\boldsymbol{\eta}$ are drawn from the posterior distribution $q(\boldsymbol{\eta}, \mathbf{Z})$, we define the latent regression rule as

$$\hat{y}(\boldsymbol{\eta}, \mathbf{z}) = \boldsymbol{\eta}^\top \bar{\mathbf{z}}. \tag{14}$$

To measure the goodness of the prediction rule (14), we adopt the widely used ϵ-insensitive loss

$$\mathcal{R}_\epsilon(\boldsymbol{\eta}, \mathbf{Z}) = \sum_{d=1}^{D} \max\left(0, |\Delta_d| - \epsilon\right),$$

where $\Delta_d = y_d - \boldsymbol{\eta}^\top \bar{\mathbf{z}}_d$ is the margin between the true score and the predicted score. The ϵ-insensitive loss has been successfully used in learning fully observed support vector regression (Smola and Scholkopf, 2003). In our case, the loss is a function of predictive model $\boldsymbol{\eta}$ as well as the topic assignments $\mathbf{Z}$ which are hidden from the input data. To resolve this uncertainty, we define the expected ϵ-insensitive loss

$$\mathcal{R}_\epsilon(q) = \mathbb{E}_q\left[\mathcal{R}_\epsilon(\boldsymbol{\eta}, \mathbf{Z})\right] = \sum_{d=1}^{D} \mathbb{E}_q\left[\max(0, |\Delta_d| - \epsilon)\right],$$

a function of the desired posterior distribution $q(\boldsymbol{\eta}, \mathbf{Z})$.

With the above definitions, we can follow the same principle as Gibbs MedLDA to define the Gibbs MedLDA regression model as solving the regularized Bayesian inference problem

$$\min_{q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})} \mathcal{L}\left(q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right) + 2c\mathcal{R}_\epsilon\left(q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right). \tag{15}$$

Note that as in the classification model, we have put the complete distribution $q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})$ as the argument of the expected loss $\mathcal{R}_\epsilon$, which only depends on the marginal distribution $q(\boldsymbol{\eta}, \mathbf{Z})$. This does not affect the results because we are taking the expectation to compute $\mathcal{R}_\epsilon$ and any irrelevant variables will be marginalized out.

As in the Gibbs MedLDA classification model, we can show that $\mathcal{R}_\epsilon$ is an upper bound of the ϵ-insensitive loss of MedLDA's expected prediction rule, by applying Jensen's inequality to the convex function $h(x) = \max(0, |x| - \epsilon)$.

Lemma 4 *We have* $\mathcal{R}_\epsilon \geq \sum_{d=1}^{D} \max(0, |\mathbb{E}_q[\Delta_d]| - \epsilon)$.

We can reformulate problem (15) in the same form as problem (9), with the unnormalized likelihood

$$\phi(y_d|\boldsymbol{\eta}, \mathbf{z}_d) = \exp\left(-2c \max(0, |\Delta_d| - \epsilon)\right).$$

Then, we have the dual scale of mixture representation, by noting that

$$\max(0, |x| - \epsilon) = \max(0, x - \epsilon) + \max(0, -x - \epsilon). \tag{16}$$

Lemma 5 (Dual Scale Mixture Representation) *For regression, the unnormalized likelihood can be expressed as*

$$\begin{aligned}\phi(y_d|\boldsymbol{\eta}, \mathbf{z}_d) = &\int_0^\infty \frac{1}{\sqrt{2\pi\lambda_d}} \exp\left(-\frac{(\lambda_d + c(\Delta_d - \epsilon))^2}{2\lambda_d}\right) \mathrm{d}\lambda_d \\ &\times \int_0^\infty \frac{1}{\sqrt{2\pi\omega_d}} \exp\left(-\frac{(\omega_d - c(\Delta_d + \epsilon))^2}{2\omega_d}\right) \mathrm{d}\omega_d.\end{aligned}$$

Proof By the equality (16), we have $\phi(y_d|\boldsymbol{\eta}, \mathbf{z}_d) = \exp\{-2c\max(0, \Delta_d - \epsilon)\} \exp\{-2c\max(0, -\Delta_d - \epsilon)\}$. Each of the exponential terms can be formulated as a scale mixture of Gaussians due to Lemma 2. ■

Then, the data augmented learning problem of the Gibbs MedLDA regression model is

$$\min_{q(\boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\omega}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})} \mathcal{L}\left(q(\boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\omega}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right) - \mathbb{E}_q\left[\log \phi(\mathbf{y}, \boldsymbol{\lambda}, \boldsymbol{\omega}|\mathbf{Z}, \boldsymbol{\eta})\right],$$

where $\phi(\mathbf{y}, \boldsymbol{\lambda}, \boldsymbol{\omega}|\mathbf{Z}, \boldsymbol{\eta}) = \prod_{d=1}^{D} \phi(y_d, \lambda_d, \omega_d|\mathbf{Z}, \boldsymbol{\omega})$ and

$$\phi(y_d, \lambda_d, \omega_d|\mathbf{Z}, \boldsymbol{\eta}) = \frac{1}{\sqrt{2\pi\lambda_d}} \exp\left(-\frac{(\lambda_d + c(\Delta_d - \epsilon))^2}{2\lambda_d}\right) \frac{1}{\sqrt{2\pi\omega_d}} \exp\left(-\frac{(\omega_d - c(\Delta_d + \epsilon))^2}{2\omega_d}\right).$$

Solving the augmented problem and integrating out $(\boldsymbol{\Theta}, \boldsymbol{\Phi})$, we can get the collapsed posterior distribution

$$q(\boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\omega}, \mathbf{Z}) \propto p_0(\boldsymbol{\eta}) p(\mathbf{W}, \mathbf{Z}|\boldsymbol{\alpha}, \boldsymbol{\beta}) \phi(\mathbf{y}, \boldsymbol{\lambda}, \boldsymbol{\omega}|\mathbf{Z}, \boldsymbol{\eta}).$$

Algorithm 2 collapsed Gibbs sampling algorithm for Gibbs MedLDA regression models

Initialization: set $\boldsymbol{\lambda} = 1$ and randomly draw z_{dk} from a uniform distribution.
for $m = 1$ **to** M **do**
  draw the classifier from the normal distribution (17)
  for $d = 1$ **to** D **do**
    for each word n in document d **do**
      draw a topic from the multinomial distribution (18)
    end for
    draw λ_d^{-1} (and thus λ_d) from the inverse Gaussian distribution (19).
    draw ω_d^{-1} (and thus ω_d) from the inverse Gaussian distribution (20).
  end for
end for

5.1.2 A Collapsed Gibbs Sampling Algorithm

Following similar derivations as in the classification model, the Gibbs sampling algorithm to infer the posterior has the following conditional distributions, with an outline in Algorithm 2.

For $\boldsymbol{\eta}$: Again, with the isotropic Gaussian prior $p_0(\boldsymbol{\eta}) = \prod_{k=1}^K \mathcal{N}(\eta_k; 0, \nu^2)$, we have

$$\begin{aligned} q(\boldsymbol{\eta}|\mathbf{Z}, \boldsymbol{\lambda}, \boldsymbol{\omega}) &\propto p_0(\boldsymbol{\eta}) \prod_{d=1}^D \exp\left(-\frac{(\lambda_d + c(\Delta_d - \epsilon))^2}{2\lambda_d}\right) \exp\left(-\frac{(\omega_d - c(\Delta_d + \epsilon))^2}{2\omega_d}\right) \\ &\propto \exp\left(-\sum_{k=1}^K \frac{\eta_k^2}{2\nu^2} - \sum_{d=1}^D \left(\frac{(\lambda_d + c(\Delta_d - \epsilon))^2}{2\lambda_d} + \frac{(\omega_d - c(\Delta_d + \epsilon))^2}{2\omega_d}\right)\right) \\ &= \exp\left(-\frac{1}{2}\boldsymbol{\eta}^\top \left(\frac{1}{\nu^2} I + c^2 \sum_{d=1}^D \rho_d \bar{\mathbf{z}}_d \bar{\mathbf{z}}_d^\top\right) \boldsymbol{\eta} + c \left(\sum_{d=1}^D \psi_d \bar{\mathbf{z}}_d\right)^\top \boldsymbol{\eta}\right) \\ &= \mathcal{N}(\boldsymbol{\eta}; \boldsymbol{\mu}, \boldsymbol{\Sigma}), \end{aligned} \tag{17}$$

where the posterior covariance matrix and the posterior mean are

$$\boldsymbol{\Sigma} = \left(\frac{1}{\nu^2} I + c^2 \sum_{d=1}^D \rho_d \bar{\mathbf{z}}_d \bar{\mathbf{z}}_d^\top\right)^{-1}, \quad \boldsymbol{\mu} = c\boldsymbol{\Sigma} \left(\sum_{d=1}^D \psi_d \bar{\mathbf{z}}_d\right),$$

and $\rho_d = \frac{1}{\lambda_d} + \frac{1}{\omega_d}$ and $\psi_d = \frac{y_d - \epsilon}{\lambda_d} + \frac{y_d + \epsilon}{\omega_d}$ are two parameters. We can easily draw a sample from a K-dimensional multivariate Gaussian distribution. The inverse can be robustly done using Cholesky decomposition.

For $\mathbf{Z}$: We can derive the conditional distribution of one variable z_{dn} given others $\mathbf{Z}_\neg$ as:

$$\begin{aligned} q(z_{dn}^k = 1|\mathbf{Z}_\neg, \boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\omega}, w_{dn} = t) \propto \frac{(C_{k,\neg n}^t + \beta_t)(C_{d,\neg n}^k + \alpha_k)}{\sum_{t=1}^V C_{k,\neg n}^t + \sum_{t=1}^V \beta_t} \exp\Big(c\gamma\psi_d\eta_k \\ - c^2\big(\frac{\gamma^2\rho_d\eta_k^2}{2} + \gamma(1-\gamma)\rho_d\eta_k\Upsilon_{dn}^k\big)\Big), \end{aligned} \tag{18}$$

where $\gamma = \frac{1}{N_d}$; and $\Upsilon^k_{dn} = \frac{1}{N_d - 1} \sum_{k'=1}^{K} \eta_{k'} C^{k'}_{d,\neg n}$ is the discriminant function value without word n. The first term is from the LDA model for observed word counts. The second term is from the supervised signal $\mathbf{y}$.

For $\boldsymbol{\lambda}$ and $\boldsymbol{\omega}$: Finally, we can derive that λ_d^{-1} and ω_d^{-1} follow the inverse Gaussian distributions:

$$q(\lambda_d^{-1}|\mathbf{Z}, \boldsymbol{\eta}, \boldsymbol{\omega}) = \mathcal{IG}\left(\lambda_d^{-1}; \frac{1}{c|\Delta_d - \epsilon|}, 1\right), \tag{19}$$

$$q(\omega_d^{-1}|\mathbf{Z}, \boldsymbol{\eta}, \boldsymbol{\lambda}) = \mathcal{IG}\left(\omega_d^{-1}; \frac{1}{c|\Delta_d + \epsilon|}, 1\right). \tag{20}$$

The per-iteration time complexity of this algorithm is similar to that of the binary Gibbs MedLDA model, that is, linear to the number of documents and the number of topics if K is not too large.

5.2 Multi-task Gibbs MedLDA

The second extension is a multi-task Gibbs MedLDA. Multi-task learning is a scenario where multiple potentially related tasks are learned jointly with the hope that their performance can be boosted by sharing some statistic information among these tasks, and it has attracted a lot of research attention. In particular, learning a common latent representation shared by all the related tasks has proven to be an effective way to capture task relationships (Ando and Zhang, 2005; Argyriou et al., 2007; Zhu et al., 2014). Here, we take the similar approach to learning multiple predictive models which share the common latent topic representations. As we shall see in Section 6.3.2, one natural application of our approach is to do multi-label classification (Tsoumakas et al., 2010), where each document can belong to multiple categories, by defining each task as a binary classifier to determine whether a data point belongs to a particular category; and it can also be applied to multi-class classification, where each document belongs to only one of the many categories, by defining a single output prediction rule (See Section 6.3.2 for details).

5.2.1 The Model with Data Augmentation

We consider L binary classification tasks and each task i is associated with a classifier with weights $\boldsymbol{\eta}_i$. We assume that all the tasks work on the same set of input data $\mathbf{W} = \{\mathbf{w}_d\}_{d=1}^D$, but each data d has different binary labels $\{y_d^i\}_{i=1}^L$ in different tasks. A multi-task Gibbs MedLDA model has two components—an LDA model to describe input words (the same as in Gibbs MedLDA); and multiple Gibbs classifiers sharing the same topic representations. When we have the classifier weights $\boldsymbol{\eta}$ and the topic assignments $\mathbf{Z}$, drawn from a posterior distribution $q(\boldsymbol{\eta}, \mathbf{Z})$, we follow the same principle as in Gibbs MedLDA and define the latent Gibbs rule for each task as

$$\forall i = 1, \dots L: \quad \hat{y}^i(\boldsymbol{\eta}_i, \mathbf{z}) = \text{sign } F(\boldsymbol{\eta}_i, \mathbf{z}; \mathbf{w}) = \text{sign}(\boldsymbol{\eta}_i^\top \bar{\mathbf{z}}). \tag{21}$$

Let $\zeta_d^i = \ell - y_d^i \boldsymbol{\eta}_i^\top \bar{\mathbf{z}}_d$. The hinge loss of the stochastic classifier i is

$$\mathcal{R}_i(\boldsymbol{\eta}_i, \mathbf{Z}) = \sum_{d=1}^{D} \max(0, \zeta_d^i)$$

and the expected hinge loss is

$$\mathcal{R}'_i(q) = \mathbb{E}_q[\mathcal{R}_i(\boldsymbol{\eta}_i, \mathbf{Z})] = \sum_{d=1}^{D} \mathbb{E}_q \left[\max(0, \zeta_d^i)\right].$$

For each task i, we can follow the argument as in Gibbs MedLDA to show that the expected loss $\mathcal{R}'_i(q)$ is an upper bound of the expected training error $\sum_{d=1}^{D} \mathbb{E}_q[\mathbb{I}(y_d^i \neq \hat{y}_d^i(\boldsymbol{\eta}_i, \mathbf{z}_d))]$ of the Gibbs classifier (21). Thus, it is a good surrogate loss for learning a posterior distribution which could lead to a low expected training error.

Then, following a similar procedure of defining the binary GibbsMedLDA classifier, we define the multi-task GibbsMedLDA model as solving the following RegBayes problem:

$$\min_{q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})} \mathcal{L}\left(q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right) + 2c\mathcal{R}'_{MT}\left(q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right),$$

where the multi-task expected hinge loss is defined as a summation of the expected hinge loss of all the tasks:

$$\mathcal{R}'_{MT}\left(q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right) = \sum_{i=1}^{L} \mathcal{R}'_i\left(q(\boldsymbol{\eta}, \boldsymbol{\Theta}, \mathbf{Z}, \boldsymbol{\Phi})\right).$$

Due to the separability of the multi-task expected hinge loss, we can apply Lemma 2 to reformulate each task-specific expected hinge loss $\mathcal{R}'_i$ as a scale mixture by introducing a set of augmented variables $\{\lambda_d^i\}_{d=1}^D$. More specifically, let $\phi_i(y_d^i|\mathbf{z}_d, \boldsymbol{\eta}) = \exp\{-2c\max(0, \zeta_d^i)\}$ be the unnormalized likelihood of the response variable for document d in task i. Then, we have

$$\phi_i(y_d^i|\mathbf{z}_d, \boldsymbol{\eta}) = \int_0^\infty \frac{1}{\sqrt{2\pi\lambda_d^i}} \exp\left(-\frac{(\lambda_d^i + c\zeta_d^i)^2}{2\lambda_d^i}\right) \mathrm{d}\lambda_d^i.$$

5.2.2 A Collapsed Gibbs Sampling Algorithm

Similar to the binary Gibbs MedLDA classification model, we can derive the collapsed Gibbs sampling algorithm, as outlined in Algorithm 3. Specifically, let

$$\phi_i(\mathbf{y}^i, \boldsymbol{\lambda}^i|\mathbf{Z}, \boldsymbol{\eta}) = \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\lambda_d^i}} \exp\left(-\frac{(\lambda_d^i + c\zeta_d^i)^2}{2\lambda_d^i}\right)$$

be the joint unnormalized likelihood of the class labels $\mathbf{y}^i = \{y_d^i\}_{d=1}^D$ and the augmentation variables $\boldsymbol{\lambda}^i = \{\lambda_d^i\}_{d=1}^D$. Then, for the multi-task Gibbs MedLDA, we can integrate out the Dirichlet variables ($\boldsymbol{\Theta}$, $\boldsymbol{\Phi}$) and get the collapsed posterior distribution

$$\begin{aligned} q(\boldsymbol{\eta}, \boldsymbol{\lambda}, \mathbf{Z}) &\propto p_0(\boldsymbol{\eta}) p(\mathbf{W}, \mathbf{Z}|\boldsymbol{\alpha}, \boldsymbol{\beta}) \prod_{i=1}^{L} \phi_i(\mathbf{y}^i, \boldsymbol{\lambda}^i|\mathbf{Z}, \boldsymbol{\eta}) \\ &= p_0(\boldsymbol{\eta}) \left[\prod_{d=1}^{D} \frac{\delta(\mathbf{C}_d + \boldsymbol{\alpha})}{\delta(\boldsymbol{\alpha})}\right] \prod_{k=1}^{K} \frac{\delta(\mathbf{C}_k + \boldsymbol{\beta})}{\delta(\boldsymbol{\beta})} \prod_{i=1}^{L} \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\lambda_d^i}} \exp\left(-\frac{(\lambda_d^i + c\zeta_d^i)^2}{2\lambda_d^i}\right). \end{aligned}$$

Algorithm 3 collapsed Gibbs sampling algorithm for multi-task Gibbs MedLDA

Initialization: set $\boldsymbol{\lambda} = 1$ and randomly draw z_{dk} from a uniform distribution.
for $m = 1$ **to** M **do**
  for $i = 1$ **to** L **do**
    draw the classifier $\boldsymbol{\eta}_i$ from the normal distribution (22)
  end for
  for $d = 1$ **to** D **do**
    for each word n in document d **do**
      draw a topic from the multinomial distribution (23)
    end for
    for $i = 1$ **to** L **do**
      draw $(\lambda_d^i)^{-1}$ (and thus λ_d^i) from the inverse Gaussian distribution (24).
    end for
  end for
end for

Then, we can derive the conditional distributions used in collapsed Gibbs sampling as follows.

For $\boldsymbol{\eta}$: We also assume its prior is an isotropic Gaussian $p_0(\boldsymbol{\eta}) = \prod_{i=1}^{L}\prod_{k=1}^{K}\mathcal{N}(\eta_{ik}; 0, \nu^2)$. Then, we have the factorized form $q(\boldsymbol{\eta}|\mathbf{Z}, \boldsymbol{\lambda}) = \prod_{i=1}^{L} q(\boldsymbol{\eta}_i|\mathbf{Z}, \boldsymbol{\lambda})$, and each individual distribution is

$$q(\boldsymbol{\eta}_i|\mathbf{Z}, \boldsymbol{\lambda}) \propto p_0(\boldsymbol{\eta}_i)\prod_{d=1}^{D}\exp\left(-\frac{(\lambda_d^i + c\zeta_d^i)^2}{2\lambda_d^i}\right) = \mathcal{N}(\boldsymbol{\eta}_i; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \tag{22}$$

where the posterior covariance matrix and posterior mean are

$$\boldsymbol{\Sigma}_i = \left(\frac{1}{\nu^2}I + c^2\sum_{d=1}^{D}\frac{\bar{\mathbf{z}}_d\bar{\mathbf{z}}_d^\top}{\lambda_d^i}\right)^{-1}, \text{ and } \boldsymbol{\mu}_i = \boldsymbol{\Sigma}_i\left(c\sum_{d=1}^{D}y_d^i\frac{\lambda_d^i + c\ell}{\lambda_d^i}\bar{\mathbf{z}}_d\right).$$

Similarly, the inverse can be robustly done using Cholesky decomposition, an $O(K^3)$ procedure. Since K is normally not large, the inversion can be done efficiently.

For $\mathbf{Z}$: The conditional distribution of $\mathbf{Z}$ is

$$q(\mathbf{Z}|\boldsymbol{\eta}, \boldsymbol{\lambda}) \propto \prod_{d=1}^{D}\frac{\delta(\mathbf{C}_d + \boldsymbol{\alpha})}{\delta(\boldsymbol{\alpha})}\left[\prod_{i=1}^{L}\exp\left(-\frac{(\lambda_d^i + c\zeta_d^i)^2}{2\lambda_d^i}\right)\right]\prod_{k=1}^{K}\frac{\delta(\mathbf{C}_k + \boldsymbol{\beta})}{\delta(\boldsymbol{\beta})}.$$

By canceling common factors, we can derive the conditional distribution of one variable z_{dn} given others $\mathbf{Z}_\neg$ as:

$$q(z_{dn}^k = 1|\mathbf{Z}_\neg, \boldsymbol{\eta}, \boldsymbol{\lambda}, w_{dn} = t) \propto \frac{(C_{k,\neg n}^t + \beta_t)(C_{d,\neg n}^k + \alpha_k)}{\sum_{t=1}^{V}C_{k,\neg n}^t + \sum_{t=1}^{V}\beta_t}\prod_{i=1}^{L}\exp\Big(\frac{\gamma y_d^i(c\ell + \lambda_d^i)\eta_{ik}}{\lambda_d^i} - c^2\frac{\gamma^2\eta_{ik}^2 + 2\gamma(1-\gamma)\eta_{ik}\Lambda_{dn}^i}{2\lambda_d^i}\Big), \tag{23}$$

where $\Lambda^i_{dn} = \frac{1}{N_d - 1} \sum_{k'=1}^{K} \eta_{ik'} C^{k'}_{d,\neg n}$ is the discriminant function value without word n. We can see that the first term is from the LDA model for observed word counts and the second term is from the supervised signal $\{y^i_d\}$ from all the multiple tasks.

For $\boldsymbol{\lambda}$: Finally, the conditional distribution of the augmented variables $\boldsymbol{\lambda}$ is fully factorized, $q(\boldsymbol{\lambda}|\mathbf{Z}, \boldsymbol{\eta}) = \prod_{i=1}^{L} \prod_{d=1}^{D} q(\lambda^i_d|\mathbf{Z}, \boldsymbol{\eta})$, and each variable follows a generalized inverse Gaussian distribution

$$q(\lambda^i_d|\mathbf{Z}, \boldsymbol{\eta}) \propto \frac{1}{\sqrt{2\pi\lambda^i_d}} \exp\left(-\frac{(\lambda^i_d + c\zeta^i_d)^2}{2\lambda^i_d}\right) = \mathcal{GIG}\left(\lambda^i_d; \frac{1}{2}, 1, c^2(\zeta^i_d)^2\right).$$

Therefore, we can derive that $(\lambda^i_d)^{-1}$ follows an inverse Gaussian distribution

$$p((\lambda^i_d)^{-1}|\mathbf{Z}, \boldsymbol{\eta}) = \mathcal{IG}\left((\lambda^i_d)^{-1}; \frac{1}{c|\zeta^i_d|}, 1\right), \tag{24}$$

from which a sample can be efficiently drawn with a constant time complexity.

The per-iteration time complexity of the algorithm is $\mathcal{O}(LK^3 + N_{total}K + DL)$. For common large-scale applications where K and L are not too large while D (thus N_{total}) is very large, the step of sampling latent topic assignments takes most of the time. If L is very large, for example, in the PASCAL large-scale text/image categorization challenge tasks which have tens of thousands of categories,[4] the step of drawing global classifier weights may dominate. A nice property of the algorithm is that we can easily parallelize this step since there is no coupling among these classifiers once the topic assignments are given. A preliminary investigation of the parallel algorithm is presented in Zhu et al. (2013b).

6. Experiments

We present empirical results to demonstrate the efficiency and prediction performance of Gibbs MedLDA (denoted by GibbsMedLDA) on the 20Newsgroups data set for classification, a hotel review data set for regression, and a Wikipedia data set with more than 1 million documents for multi-label classification. We also analyze its sensitivity to key parameters and examine the learned latent topic representations qualitatively. The 20Newsgroups data set contains about 20K postings within 20 groups. We follow the same setting as in Zhu et al. (2012) and remove a standard list of stop words for both binary and multi-class classification. For all the experiments, we use the standard normal prior $p_0(\boldsymbol{\eta})$ (i.e., $\nu^2 = 1$) and the symmetric Dirichlet priors $\boldsymbol{\alpha} = \frac{\alpha}{K}\mathbf{1}$, $\boldsymbol{\beta} = 0.01 \times \mathbf{1}$, where $\mathbf{1}$ is a vector with all entries being 1. For each setting, we report the average performance and standard deviation with five randomly initialized runs. All the experiments, except the those on the large Wikipedia data set, are done on a standard desktop computer.

6.1 Binary Classification

The binary classification task is to distinguish postings of the newsgroup *alt.atheism* and postings of the newsgroup *talk.religion.misc*. The training set contains 856 documents, and

4. See the websites: `http://lshtc.iit.demokritos.gr/`; and `http://www.image-net.org/challenges/LSVRC/2012/index`.

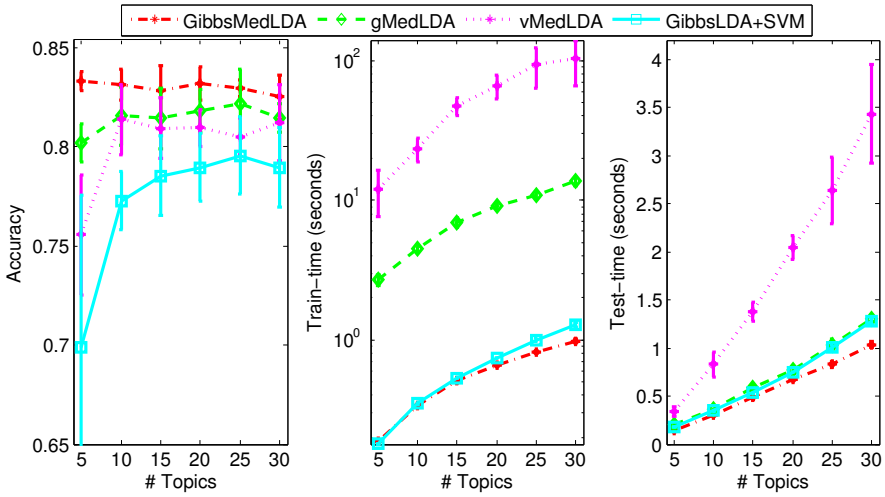


Figure 1: Classification accuracy, training time (in log-scale) and testing time (in linear scale) on the 20Newsgroups binary classification data set.

the test set contains 569 documents. We compare Gibbs MedLDA with the MedLDA model that uses variational methods (denoted by vMedLDA) (Zhu et al., 2012) and the MedLDA that uses collapsed Gibbs sampling algorithms (denoted by gMedLDA) (Jiang et al., 2012). We also include unsupervised LDA using collapsed Gibbs sampling as a baseline, denoted by GibbsLDA. For GibbsLDA, we learn a binary linear SVM on its topic representations using SVMLight (Joachims, 1999). The results of other supervised topic models, such as sLDA and DiscLDA (Lacoste-Jullien et al., 2009), were reported in Zhu et al. (2012). For Gibbs MedLDA, we set $\alpha = 1$, $\ell = 164$ and $M = 10$. As we shall see in Section 6.5, Gibbs MedLDA is insensitive to α, ℓ and M in a wide range. Although tuning c (e.g., via cross-validation) can produce slightly better results, we fix $c = 1$ for simplicity.

Figure 1 shows the accuracy, training time, and testing time of different methods with various numbers of topics. We can see that by minimizing an expected hinge-loss and making no restrictive assumptions on the posterior distributions, GibbsMedLDA achieves higher accuracy than other max-margin topic models, which make some restrictive mean-field assumptions. Similarly, as gMedLDA makes a weaker mean-field assumption, it achieves slightly higher accuracy than vMedLDA, which assumes that the posterior distribution is fully factorized. For the training time, GibbsMedLDA is about two orders of magnitudes faster than vMedLDA, and about one order of magnitude faster than gMedLDA. This is partly because both vMedLDA and gMedLDA need to solve multiple SVM problems. For the testing time, GibbsMedLDA is comparable with gMedLDA and the unsupervised GibbsLDA, but faster than the variational algorithm used by vMedLDA, especially when the number of topics K is large. There are several possible reasons for the faster testing than vMedLDA, though they use the same stopping criterion. For example, vMedLDA performs mean-field inference in a full space which leads to a low convergence speed, while GibbsMedLDA carries out Gibbs sampling in a collapsed space. Also, the sparsity of the

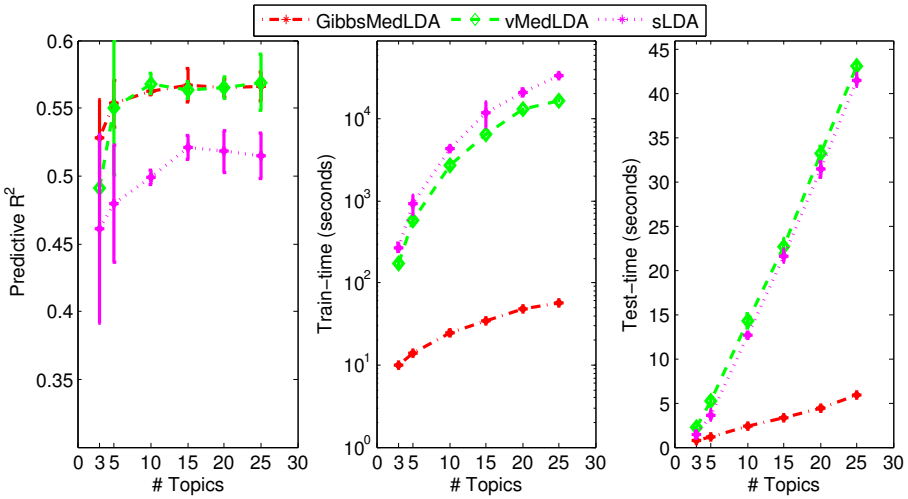


Figure 2: Predictive R2, training time and testing time on the hotel review data set.

sampled topics in GibbsMedLDA could save time, while vMedLDA needs to carry out computation for each dimension of the variational parameters.

6.2 Regression

We use the hotel review data set (Zhu and Xing, 2010) built by randomly crawling hotel reviews from the TripAdvisor website,[5] where each review is associated with a global rating score ranging from 1 to 5. In these experiments, we focus on predicting the global rating scores for reviews using the bag-of-words features only, with a vocabulary of 12,000 terms, though the other manually extracted features (e.g., part-of-speech tags) are provided. All the reviews have character lengths between 1,500 and 6,000. The data set consists of 5,000 reviews, with 1,000 reviews per rating. The data set is uniformly partitioned into training and testing sets. We compare the Gibbs MedLDA regression model with the MedLDA regression model that uses variational inference and supervised LDA (sLDA) which also uses variational inference. For Gibbs MedLDA and vMedLDA, the precision is set at $\epsilon = 1e^{-3}$ and c is selected via 5 fold cross-validation during training. Again, we set the Dirichlet parameter $\alpha = 1$ and the number of burn-in $M = 10$.

Figure 2 shows the predictive R^2 (Blei and McAuliffe, 2007) of different methods. We can see that GibbsMedLDA achieves comparable prediction performance with vMedLDA, which is better than sLDA. Note that vMedLDA uses a full likelihood model for both input words and response variables, while GibbsMedLDA uses a simpler likelihood model for words only.[6] For training time, GibbsMedLDA is about two orders of magnitudes faster than vMedLDA (as well as sLDA), again due to the fact that GibbsMedLDA does not need to solve multiple SVM problems. For testing time, GibbsMedLDA is also much faster than vMedLDA and sLDA, especially when the number of topics is large, due to the same reasons as stated in Section 6.1.

5. See the website: `http://www.tripadvisor.com/`.
6. The MedLDA with a simple likelihood on words only doesn't perform well for regression.

6.3 Multi-class Classification

We perform multi-class classification on 20Newsgroups with all 20 categories. The data set has a balanced distribution over the categories. The test set consists of 7,505 documents, in which the smallest category has 251 documents and the largest category has 399 documents. The training set consists of 11,269 documents, in which the smallest and the largest categories contain 376 and 599 documents, respectively. We consider two approaches to doing multi-class classification—one is to build multiple independent binary Gibbs MedLDA models, one for each category, and the other one is to build multiple dependent binary Gibbs MedLDA models under the framework of multi-task learning, as presented in Section 5.2.

6.3.1 Multiple One-vs-All Classifiers

Various methods exist to apply binary classifiers to do multi-class classification, including the popular "one-vs-all" and "one-vs-one" strategies. Here we choose the "one-vs-all" strategy, which has shown effective (Rifkin and Klautau, 2004), to provide some preliminary analysis. Let $\hat{\boldsymbol{\eta}}_i$ be the sampled classifier weights of the 20 "one-vs-all" binary classifiers after the burn-in stage. For a test document $\mathbf{w}$, we need to infer the latent topic assignments $\mathbf{z}_i$ under each "one-vs-all" binary classifier using a Gibbs sampler with the conditional distribution (13). Then, we predict the document as belonging to the single category which has the largest discriminant function value, that is,

$$\hat{y} = \underset{i=1,\ldots,L}{\operatorname{argmax}} (\hat{\boldsymbol{\eta}}_i^\top \bar{\mathbf{z}}_i),$$

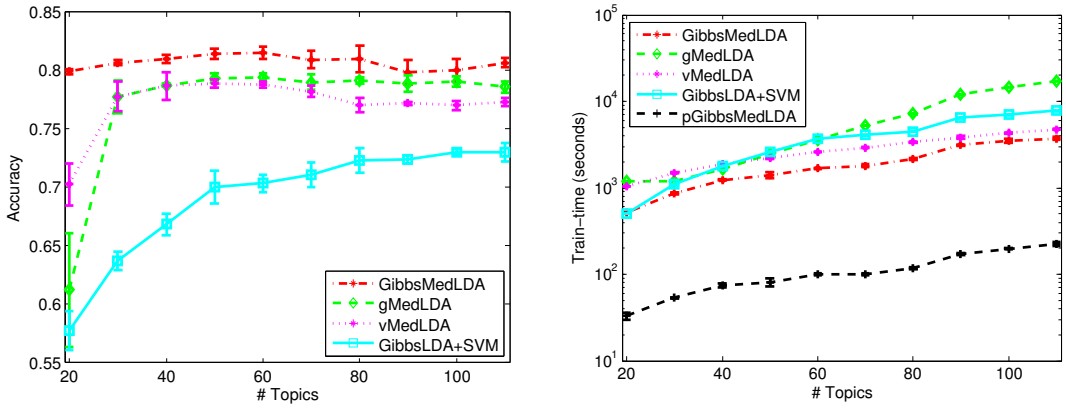
where L is the number of categories (i.e., 20 in this experiment). Again, since GibbsMedLDA is insensitive to α and ℓ, we set $\alpha = 1$ and $\ell = 64$. We also fix $c = 1$ for simplicity. The number of burn-in iterations is set as $M = 20$, which is sufficiently large as will be shown in Figure 7.

Figure 3 shows the classification accuracy and training time, where GibbsMedLDA builds 20 binary Gibbs MedLDA classifiers. Note that for GibbsMedLDA the horizontal axis denotes the number of topics used by each single binary classifier. Since there is no coupling among these 20 binary classifiers, we can learn them in parallel, which we denote by pGibbsMedLDA. We can see that GibbsMedLDA clearly improves over other competitors on the classification accuracy, which may be due to the different strategies on building the multi-class classifiers.[7] However, given the performance gain on the binary classification task, we believe that the Gibbs sampling algorithm without any restrictive factorization assumptions is another factor leading to the improved performance. For training time, GibbsMedLDA takes slightly less time than the variational MedLDA as well as gMedLDA. But if we train the 20 binary GibbsMedLDA classifiers in parallel, we can save a lot of training time. These results are promising since it is now not uncommon to have a desktop computer with multiple processors or a cluster with tens or hundreds of computing nodes.

6.3.2 Multi-class Classification as a Multi-task Learning Problem

The second approach to performing multi-class classification is to formulate it as a multiple task learning problem, with a single output prediction rule. Specifically, let the label space

7. MedLDA learns multi-class SVM (Zhu et al., 2012).



(a) (b)

Figure 3: (a) classification accuracy and (b) training time of the one-vs-all Gibbs MedLDA classifiers for multi-class classification on the whole 20Newsgroups data set.

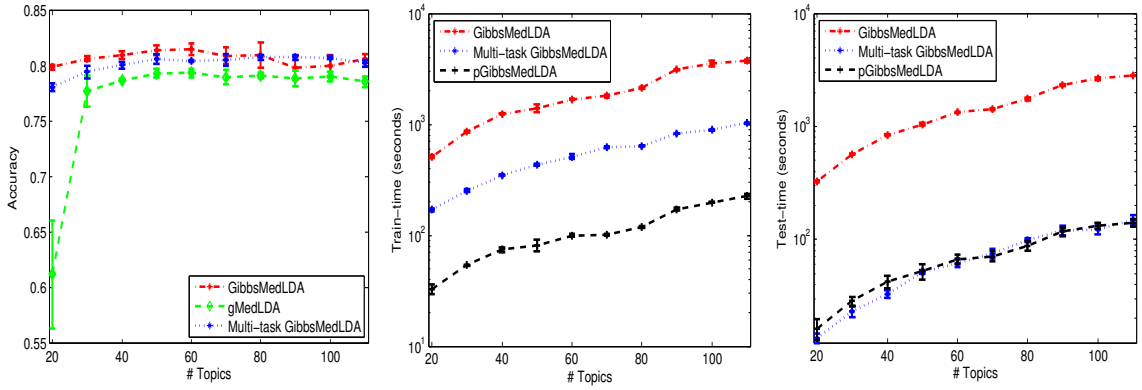
be $\mathcal{Y} = \{1, \ldots, L\}$. We can define one binary classification task for each category i and the task is to distinguish whether a data example belongs to the class i (with binary label $+1$) or not (with binary label -1). All the binary tasks share the same topic representations. To apply the model as we have presented in Section 5.2, we need to determine the true binary label of each document in a task. Given the multi-class label y_d of document d, this can be easily done by defining

$$\forall i = 1, \ldots, L: \; y_d^i = \begin{cases} +1 \text{ if } y_d = i \\ -1 \text{ otherwise} \end{cases}.$$

Then, we can learn a multi-task Gibbs MedLDA model using the data with transferred multiple labels. Let $\hat{\boldsymbol{\eta}}_i$ be the sampled classifier weights of task i after the burn-in stage. For a test document $\mathbf{w}$, once we have inferred the latent topic assignments $\mathbf{z}$ using a Gibbs sampler with the conditional distribution (13), we compute the discriminant function value $\hat{\boldsymbol{\eta}}_i^\top \bar{\mathbf{z}}$ for each task i, and predict the document as belonging to the single category which has the largest discriminant function value, that is,

$$\hat{y} = \underset{i=1,\ldots,L}{\text{argmax}} \, (\hat{\boldsymbol{\eta}}_i^\top \bar{\mathbf{z}}).$$

Figure 4 shows the performance of the multi-task Gibbs MedLDA with comparison to the high-performance methods of the one-vs-all GibbsMedLDA and gMedLDA. Note again that for the one-vs-all GibbsMedLDA the horizontal axis denotes the number of topics used by each single binary classifier. We can see that although the multi-task GibbsMedLDA uses 20 times fewer topics than the one-vs-all GibbsMedLDA, their prediction accuracy scores are comparable when the multi-task GibbsMedLDA uses a reasonable number of topics (e.g., larger than 40). Both implementations of Gibbs MedLDA yield higher performance than gMedLDA. Looking at training time, when there is only a single processor core available, the multi-task GibbsMedLDA is about 3 times faster than the one-vs-all GibbsMedLDA. When there are multiple processor cores available, the naive parallel one-vs-all Gibbs MedLDA is



(a) (b) (c)

Figure 4: (a) classification accuracy; (b) training time; and (c) testing time of the multi-task Gibbs MedLDA classifiers for multi-class classification on the whole 20Newsgroups data set.

faster. In this case, using 20 processor cores, the parallel one-vs-all GibbsMedLDA is about 7 times faster than the multi-task GibbsMedLDA. In some scenarios, the testing time is significant. We can see that using a single core, the multi-task GibbsMedLDA is about 20 times faster than the one-vs-all GibbsMedLDA. Again however, in the presence of multiple processor cores, in this case 20, the parallel one-vs-all GibbsMedLDA tests at least as fast, at the expense of using more processor resources. So, depending on the processor cores available, both the parallel one-vs-all GibbsMedLDA and the multi-task GibbsMedLDA can be excellent choices. Where high efficiency single-core processing is key, then the multi-task GibbsMedLDA is a great choice. When there are many processor cores available, then the parallel one-vs-all GibbsMedLDA might be an appropriate choice.

6.4 Multi-label Classification

We also present some results on a multi-label classification task. We use the Wiki data set which is built from the large Wikipedia set used in the PASCAL LSHC challenge 2012, and where each document has multiple labels. The original data set is extremely imbalanced.[8] We built our data set by selecting the 20 categories that have the largest numbers of documents and keeping all the documents that are labeled by at least one of these 20 categories. The training set consists of 1.1 millions of documents and the testing set consists of 5,000 documents. The vocabulary has 917,683 terms in total. To examine the effectiveness of Gibbs MedLDA, which performs topic discovery and classifier learning jointly, we compare it with a linear SVM classifier built on the raw bag-of-words features and a two-step approach denoted by GibbsLDA+SVM. The GibbsLDA+SVM method first uses LDA with collapsed Gibbs sampling to discover latent topic representations for all the documents and then builds 20 separate binary SVM classifiers using the training documents with their

8. The data set is available at: `http://lshtc.iit.demokritos.gr/`.

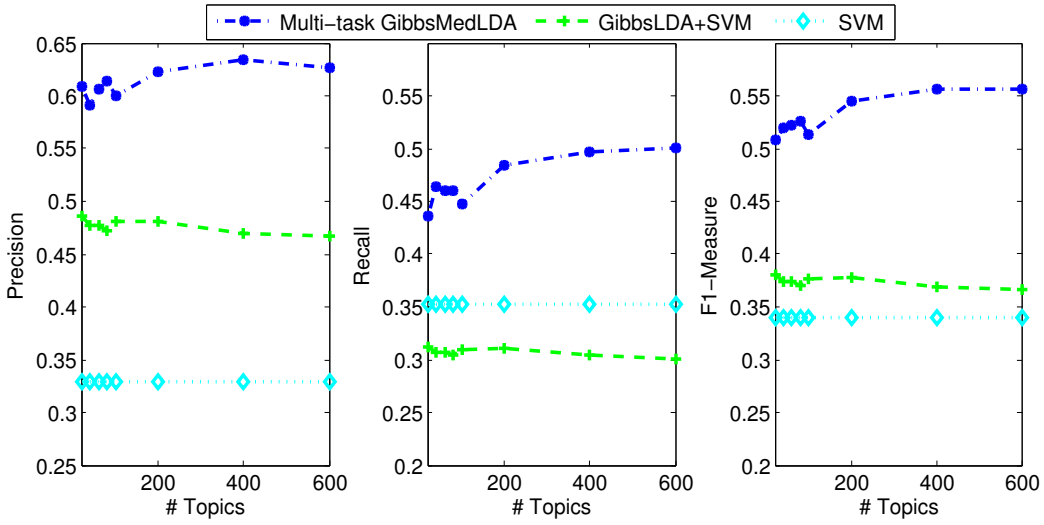


Figure 5: Precision, recall and F1-measure of the multi-label classification using different models on the Wiki data set.

discovered topic representations. For multi-task Gibbs MedLDA, we use 40 burn-in steps, which is sufficiently large. The model is insensitive to other parameters, similar to the multi-class classification task.

Figure 5 shows the precision, recall and F1 measure (i.e., the harmonic mean of precision and recall) of various models running on a distributed cluster with 20 nodes (each node is equipped with two 6-core CPUs).[9] We can see that the multi-task Gibbs MedLDA performs much better than other competitors. There are several reasons for the improvements. Since the vocabulary has about 1 million terms, the raw features are in a high-dimensional space and each document gives rise to a sparse feature vector, that is, only a few elements are nonzero. Thus, learning SVM classifiers on the raw data leads not just to over-fitting but a wider failure to generalize. For example, two documents from the same category might contain non-intersecting sets of words, yet contain similar latent topics. Using LDA to discover latent topic representations can produce dense features. Building SVM classifiers using the latent topic features improves the overall F1 measure, by improving the ability to generalize, and reducing overfitting. But, due to its two-step procedure, the discovered topic representations may not be very predictive. By doing max-margin learning and topic discovery jointly, the multi-task GibbsMedLDA can discover more discriminative topic features, thus improving significantly over the two-step GibbsLDA+SVM algorithm.

6.5 Sensitivity Analysis

We now provide a more careful analysis of the various Gibbs MedLDA models on their sensitivity to some key parameters in the classification tasks. Specifically, we will look at

9. For GibbsLDA, we use the parallel implementation in Yahoo-LDA, which is publicly available at: `https://github.com/shravanmn/Yahoo_LDA`. For Gibbs MedLDA, the parallel implementation of our Gibbs sampler is presented in Zhu et al. (2013b).

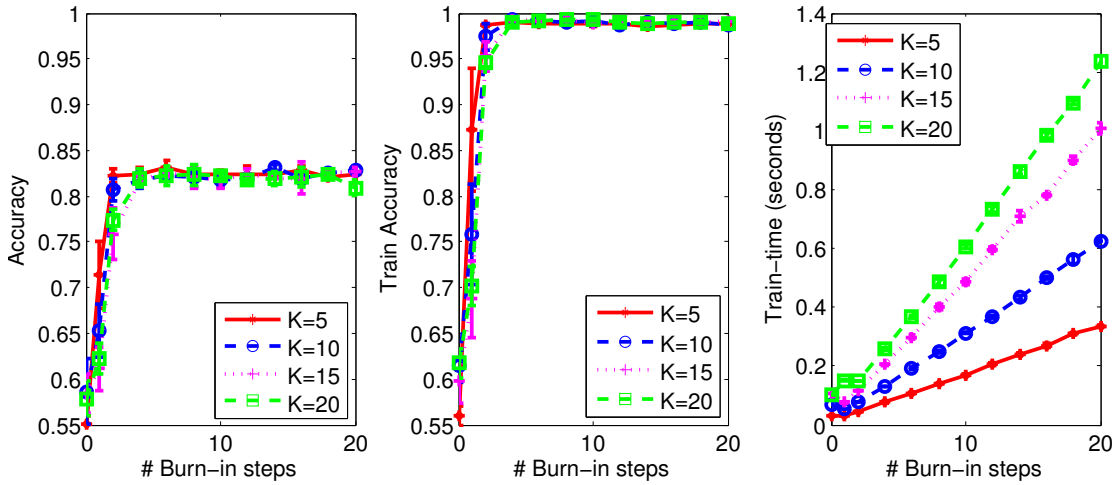


Figure 6: (Left) testing accuracy, (Middle) training accuracy, and (Right) training time of GibbsMedLDA with different numbers of burn-in steps for binary classification.

the effects of the number of burn-in steps, the Dirichlet prior α, the loss penalty ℓ, and the number of testing samples.

6.5.1 Burn-in Steps

Figure 6 shows the classification accuracy, training accuracy and training time of GibbsMedLDA with different numbers of burn-in samples in the binary classification task. When $M = 0$, the model is essentially random, for which we draw a classifier with the randomly initialized topic assignments for training data. We can see that both the training accuracy and testing accuracy increase very quickly and converge to their stable values with 5 to 10 burn-in steps. As expected, the training time increases about linearly in general when using more burn-in steps. Moreover, the training time increases linearly as K increases. In the previous experiments, we have chosen $M = 10$, which is sufficiently large.

Figure 7 shows the performance of GibbsMedLDA for multi-class classification with different numbers of burn-in steps when using the one-vs-all strategy. We show the total training time as well as the training time of the naive parallel implementation of pGibbsMedLDA. We can see that when the number of burn-in steps is larger than 20, the performance is quite stable, especially when K is large. Again, the training time grows about linearly as the number of burn-in steps increases. Even if we use 40 or 60 steps of burn-in, the training time is still competitive, compared with the variational MedLDA, especially considering that GibbsMedLDA can be naively parallelized by learning different binary classifiers simultaneously.

Figure 8 shows the testing classification accuracy, training accuracy and training time of the multi-task Gibbs MedLDA for multi-class classification with different numbers of burn-in steps. We can see that again both the training accuracy and testing accuracy increase fast and converge to their stable scores after about 30 burn-in steps. Also, the training time increases about linearly as the number of burn-in steps increases.

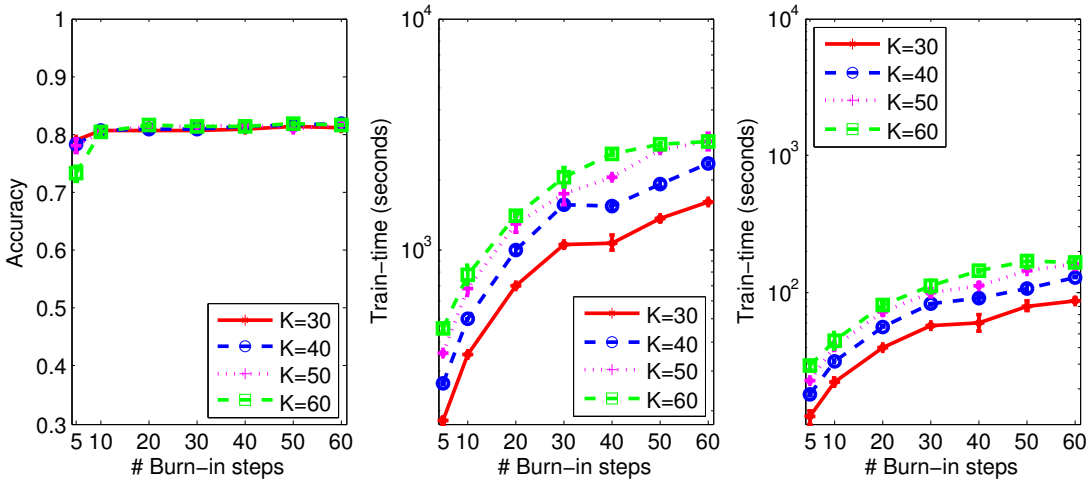


Figure 7: (Left) classification accuracy of GibbsMedLDA, (Middle) training time of GibbsMedLDA and (Right) training time of the parallel pGibbsMedLDA with different numbers of burn-in steps for multi-class classification.

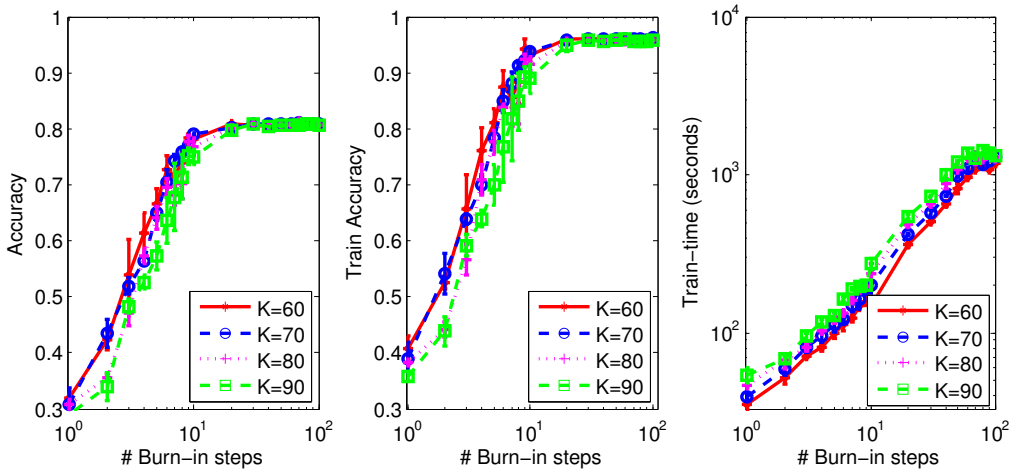


Figure 8: (Left) test accuracy, (Middle) training accuracy, and (Right) training time of the multi-task GibbsMedLDA with different numbers of burn-in steps for multi-class classification.

6.5.2 Dirichlet Prior

For topic models with a Dirichlet prior, the Dirichlet hyper-parameter can be automatically estimated, such as using the Newton-Raphson method (Blei et al., 2003). Here, we analyze its effects on the performance by setting different values. Figure 9 shows the classification performance of GibbsMedLDA on the binary task with different α values for the symmetric Dirichlet prior $\boldsymbol{\alpha} = \frac{\alpha}{K}\mathbf{1}$. For the three different topic numbers, we can see that the performance is quite stable in a wide range of α values, for example, from 0.1 to 10. We can also

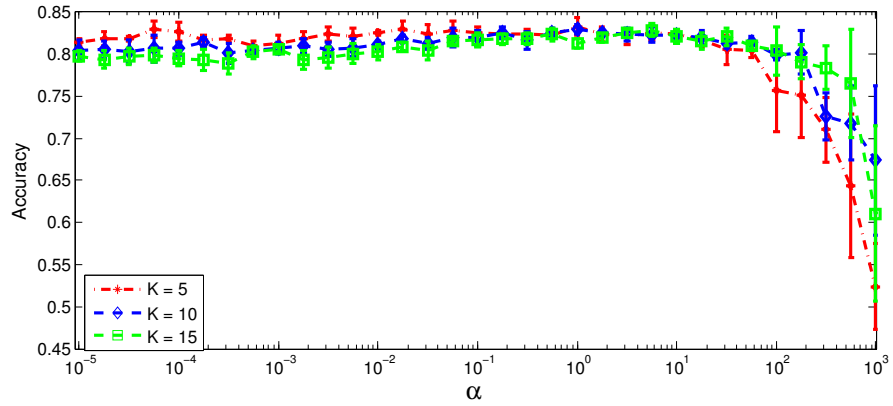


Figure 9: Classification accuracy of GibbsMedLDA on the binary classification data set with different α values.

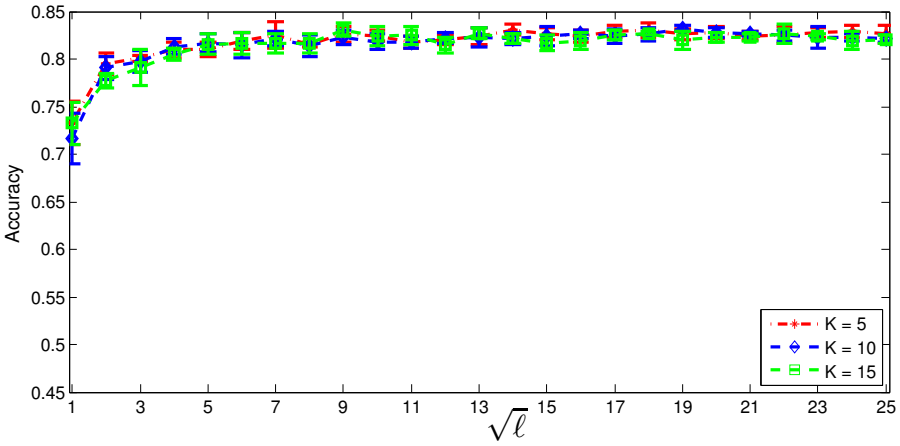


Figure 10: Classification accuracy of GibbsMedLDA on the binary classification data set with different ℓ values.

see that it generally needs a larger α in order to get the best results when K becomes larger. For example, when $\alpha < 0.1$, using fewer topics results in slightly higher performance. This is mainly because a large K tends to produce sparse topic representations and an appropriately large α is needed to smooth the representations, as the effective Dirichlet prior is $\alpha_k = \alpha/K$.

6.5.3 Loss Penalty ℓ

Figure 10 shows the classification performance of GibbsMedLDA on the binary classification task with different ℓ values. Again, we can see that in a wide range, for example, from 25 to 625, the performance is quite stable for all the three different K values. In the

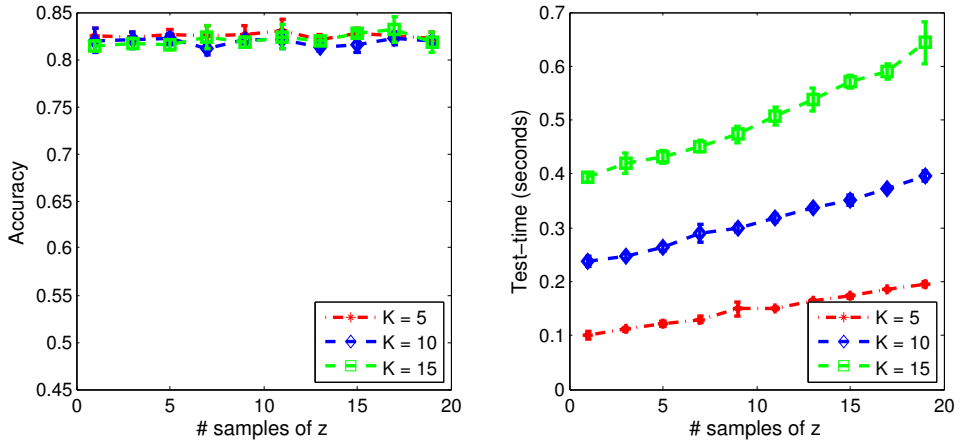


Figure 11: (Left) classification accuracy and (Right) testing time of GibbsMedLDA on the binary classification data set with different numbers of $\mathbf{z}$ samples in making predictions.

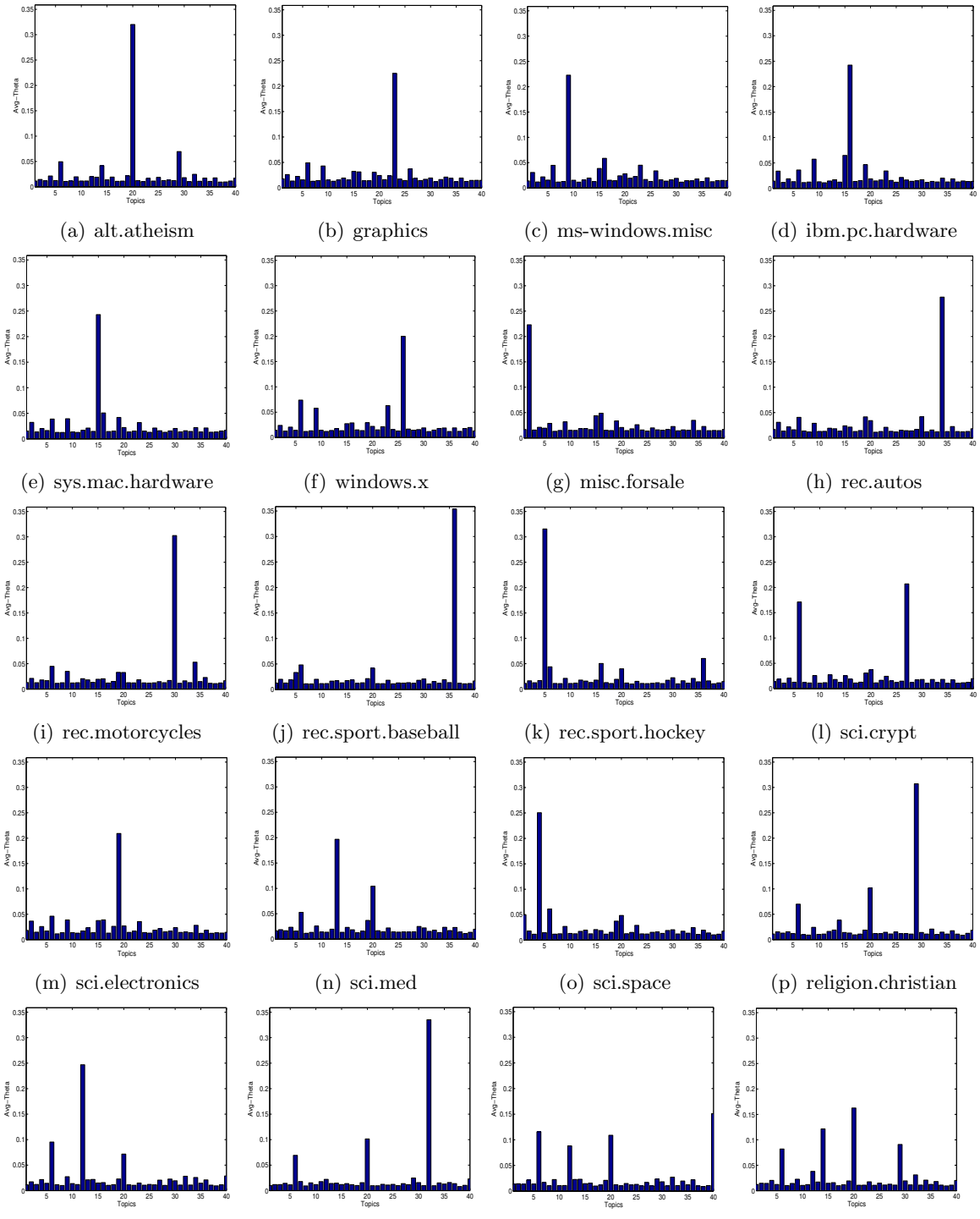
above experiments, we set $\ell = 164$. For the multi-class classification task, we have similar observations, and we set $\ell = 64$ in the previous experiments.

6.5.4 The Number of Testing Samples

Figure 11 shows the classification performance and testing time of GibbsMedLDA in the binary classification task with different numbers of $\mathbf{z}$ samples when making predictions, as stated in Section 4.4. We can see that in a wide range, for example, from 1 to 19, the classification performance is quite stable for all the three different K values we have tested; and the testing time increases about linearly as the number of $\mathbf{z}$ samples increases. For the multi-class classification task, we have similar observations.

6.6 Topic Representations

Finally, we also visualize the discovered latent topic representations of Gibbs MedLDA on the 20Newsgroup data set. We choose the multi-task Gibbs MedLDA, since it learns a single common topic space shared by multiple classifiers. We set the number of topics at 40. Figure 12 shows the average topic representations of the documents from each category, and Table 1 presents the 10 most probable words in each topic. We can see that for different categories, the average representations are quite different, indicating that the topic representations are good at distinguishing documents from different classes. We can also see that on average the documents in each category have very few salient topics, that is, the topics with a high probability of describing the documents. For example, the first two most salient topics for describing the documents in the category *alt.atheism* are topic 20 and topic 29, whose top-ranked words (see Table 1) reflect the semantic meaning of the category. For *graphics* category, the documents have the most salient topic 23, which



(a) alt.atheism (b) graphics (c) ms-windows.misc (d) ibm.pc.hardware

(e) sys.mac.hardware (f) windows.x (g) misc.forsale (h) rec.autos

(i) rec.motorcycles (j) rec.sport.baseball (k) rec.sport.hockey (l) sci.crypt

(m) sci.electronics (n) sci.med (o) sci.space (p) religion.christian

(q) politics.guns (r) politics.mideast (s) politics.misc (t) religion.misc

Figure 12: (a-t) per-class average topic representations on the 20Newsgroups data set.

has topic words *image*, *graphics*, *file*, *jpeg*, and etc., all of which are closely related to the semantic of graphics. For other categories, we have similar observations.

Topic 1	**Topic 2**	**Topic 3**	**Topic 4**	**Topic 5**	**Topic 6**	**Topic 7**	**Topic 8**	**Topic 9**	**Topic 10**
data	sale	woman	space	team	writes	mr	db	windows	file
mission	offer	told	nasa	game	don	president	cs	writes	congress
center	shipping	afraid	launch	hockey	time	stephanopoulos	mov	article	january
sci	dos	building	writes	play	article	jobs	bh	file	centers
jpl	mail	couldn	earth	season	information	russian	si	files	bill
planetary	price	floor	article	nhl	number	administration	al	problem	quotes
mass	condition	beat	orbit	ca	people	meeting	byte	dos	hr
probe	good	standing	moon	games	make	george	theory	don	states
ames	interested	immediately	shuttle	players	work	russia	bits	run	march
atmosphere	sell	crowd	gov	year	part	working	larson	win	included
Topic 11	**Topic 12**	**Topic 13**	**Topic 14**	**Topic 15**	**Topic 16**	**Topic 17**	**Topic 18**	**Topic 19**	**Topic 20**
organizations	gun	msg	jesus	mac	drive	ma	wiring	writes	writes
began	people	health	god	apple	scsi	nazis	supply	power	people
security	guns	medical	people	writes	mb	mu	boxes	article	article
terrible	weapons	food	christian	drive	card	conflict	bnr	don	don
association	firearms	article	bible	problem	system	ql	plants	ground	god
sy	writes	disease	sandvik	mb	controller	te	reduce	good	life
publication	article	patients	christians	article	bus	ne	corp	current	things
helped	government	writes	objective	system	hard	eu	relay	work	apr
organized	fire	doctor	ra	bit	ide	cx	onur	circuit	evidence
bullock	law	research	christ	don	disk	qu	damage	ve	time
Topic 21	**Topic 22**	**Topic 23**	**Topic 24**	**Topic 25**	**Topic 26**	**Topic 27**	**Topic 28**	**Topic 29**	**Topic 30**
ms	pub	image	internet	fallacy	window	key	unit	god	bike
myers	ftp	graphics	anonymous	conclusion	server	encryption	heads	jesus	dod
os	mail	file	privacy	rule	motif	chip	master	people	writes
vote	anonymous	files	posting	innocent	widget	clipper	multi	church	article
votes	archive	software	email	perfect	sun	government	led	christians	ride
santa	electronic	jpeg	anonymity	assertion	display	keys	vpic	christ	don
fee	server	images	users	true	application	security	dual	christian	apr
impression	faq	version	postings	ad	mit	escrow	ut	bible	ca
issued	eff	program	service	consistent	file	secure	ratio	faith	motorcycle
voting	directory	data	usenet	perspective	xterm	nsa	protected	truth	good
Topic 31	**Topic 32**	**Topic 33**	**Topic 34**	**Topic 35**	**Topic 36**	**Topic 37**	**Topic 38**	**Topic 39**	**Topic 40**
matthew	israel	courtesy	car	didn	year	south	entry	open	people
wire	people	announced	writes	apartment	writes	war	output	return	government
neutral	turkish	sequence	cars	sumgait	article	tb	file	filename	article
judas	armenian	length	article	started	game	nuclear	program	read	tax
reported	jews	rates	don	mamma	team	rockefeller	build	input	make
acts	israeli	molecular	good	father	baseball	georgia	um	char	health
island	armenians	sets	engine	karina	good	military	entries	write	state
safety	article	pm	speed	ll	don	bay	ei	enter	don
hanging	writes	automatic	apr	guy	games	acquired	eof	judges	money
outlets	turkey	barbara	oil	knew	runs	ships	printf	year	cramer

Table 1: The ten most probable words in the topics discovered by Multi-task Gibbs MedLDA ($K = 40$) on the 20Newsgroups data set.

7. Conclusions and Discussions

We have presented Gibbs MedLDA, an alternative approach to learning max-margin supervised topic models by minimizing an expected margin loss. We have applied Gibbs MedLDA to various tasks including text categorization, regression, and multi-task learning. By using data augmentation techniques, we have presented simple and highly efficient "augment-and-collapse" Gibbs sampling algorithms, without making any restricting assumptions on posterior distributions. Empirical results on the medium-sized 20Newsgroups and hotel re-

view data sets and a large-scale Wikipedia data set demonstrate significant improvements on time efficiency and classification accuracy over existing max-margin topic models. Our approaches are applicable to building other max-margin latent variable models, such as the max-margin nonparametric latent feature models for link prediction (Zhu, 2012) and matrix factorization (Xu et al., 2012). Finally, we release the code for public use.[10]

The new data augmentation formulation without any need to solve constrained subproblems has shown great promise on improving the time efficiency of max-margin topic models. For future work, we are interested in developing highly scalable sampling algorithms (e.g., using a distributed architecture) (Newman et al., 2009; Smola and Narayanamurthy, 2010; Ahmed et al., 2012) to deal with large scale data sets. One nice property of the sampling algorithms is that the augmented variables are local to each document. Therefore, they can be effectively handled in a distributed architecture. But, the global prediction model weights bring in new challenges. Some preliminary work has been investigated in Zhu et al. (2013b). Another interesting topic is to apply the data augmentation technique to deal with the multiclass max-margin formulation, which was proposed by Crammer and Singer (2001) and used in MedLDA for learning multi-class max-margin topic models. Intuitively, it can be solved following an iterative procedure that infers the classifier weights associated with each category by fixing the others, similar as in polychomotous logistic regression (Holmes and Held, 2006), in which each substep may involve solving a binary hinge loss and thus our data augmentation techniques can be applied. A systematical investigation is our future work.

Acknowledgments

The work was supported by the National Basic Research Program (973 Program) of China (Nos. 2013CB329403, 2012CB316301), National Natural Science Foundation of China (Nos. 61322308, 61332007, 61305066), Tsinghua University Initiative Scientific Research Program (No. 20121088071), a Microsoft Research Asia Research Fund (No. 20123000007), and a China Postdoctoral Science Foundation Grant (No. 2013T60117) to NC.

References

Amr Ahmed, Mohamed Aly, Joseph Gonzalez, Shravan Narayanamurthy, and Alex Smola. Scalable inference in latent variable models. In *International Conference on Web Search and Data Mining (WSDM)*, pages 123–132, 2012.

Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research (JMLR)*, 9:1981–2014, 2008.

Rie K. Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research (JMLR)*, (6):1817–1853, 2005.

10. The code is available at: `http://www.ml-thu.net/~jun/gibbs-medlda.shtml`.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 41–48, 2007.

David M. Blei and John D. McAuliffe. Supervised topic models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 121–128, 2007.

David M. Blei, Andrew Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022, 2003.

Olivier Catoni. PAC-Bayesian supervised classification: The thermodynamics of statistical learning. *Monograph Series of the Institute of Mathematical Statistics*, 2007.

Jonathan Chang and David Blei. Relational topic models for document networks. In *Artificial Intelligence and Statistics (AISTATS)*, pages 81–88, 2009.

Ning Chen, Jun Zhu, Fuchun Sun, and Eric P. Xing. Large-margin predictive latent subspace learning for multiview data analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(12):2365–2378, 2012.

Ning Chen, Jun Zhu, Fei Xia, and Bo Zhang. Generalized relational topic models with data augmentation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1273–1279, 2013.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research (JMLR)*, (2):265–292, 2001.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Ser. B*, (39):1–38, 1977.

Luc Devroye. *Non-uniform Random Variate Generation*. Springer-Verlag, 1986.

Li Fei-Fei and Pietro Perona. A Bayesian hierarchical model for learning natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 524–531, 2005.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research (JMLR)*, 11:2001–2049, 2010.

Pascal Germain, Alexandre Lacasse, Francois Laviolette, and Mario Marchand. PAC-Bayesian learning of linear classifiers. In *International Conference on Machine Learning (ICML)*, pages 353–360, 2009.

Thomas Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of National Academy of Science (PNAS)*, pages 5228–5235, 2004.

James P. Hobert. *The Data Augmentation Algorithm: Theory and Methodology*. In *Handbook of Markov Chain Monte Carlo* (S. Brooks, A. Gelman, G. Jones and X.-L. Meng, eds.). Chapman & Hall/CRC Press, Boca Raton, FL., 2011.

Chris C. Holmes and Leonhard Held. Bayesian auxiliary variable models for binary and multinomial regression. *Bayesian Analysis*, 1(1):145–168, 2006.

Tommi Jaakkola, Marina Meila, and Tony Jebara. Maximum entropy discrimination. In *Advances in Neural Information Processing Systems (NIPS)*, 1999.

Tony Jebara. *Discriminative, Generative and Imitative Learning*. PhD thesis, Media Laboratory, MIT, Dec 2001.

Qixia Jiang, Jun Zhu, Maosong Sun, and Eric P. Xing. Monte Carlo methods for maximum margin supervised topic models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1601–1609, 2012.

Thorsten Joachims. *Making Large-scale SVM Learning Practical*. MIT press, 1999.

Simons Lacoste-Jullien, Fei Sha, and Michael I. Jordan. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 897–904, 2009.

David McAllester. PAC-Bayesian stochastic model selection. *Machine Learning*, 51:5–21, 2003.

Xiao-Li Meng and David A. van Dyk. The EM algorithm – an old folk song sung to a fast new tune. *Journal of the Royal Statistical Society, Ser. B*, (59):511–567, 1997.

John R. Michael, William R. Schucany, and Roy W. Haas. Generating random variates using transformations with multiple roots. *The American Statistician*, 30(2):88–90, 1976.

Kevin Miller, M. Pawan Kumar, Ben Packer, Danny Goodman, and Daphne Koller. Max-margin min-entropy models. In *Artificial Intelligence and Statistics (AISTATS)*, pages 779–787, 2012.

Radford M. Neal. Markov chain Monte Carlo methods based on 'slicing' the density function. *Technical Report No. 9722, Department of Statistics, University of Toronto*, 1997.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. Distributed algorithms for topic models. *Journal of Machine Learning Research (JMLR)*, (10):1801–1828, 2009.

Nicholas G. Polson and Steven L. Scott. Data augmentation for support vector machines. *Bayesian Analysis*, 6(1):1–24, 2011.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research (JMLR)*, (5):101–141, 2004.

Alex Smola and Shravan Narayanamurthy. An architecture for parallel topic models. *Very Large Data Base (VLDB)*, 3(1-2):703–710, 2010.

Alex Smola and Bernhard Scholkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2003.

Robert H. Swendsen and Jian-Sheng Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters*, (58):86–88, 1987.

Aixin Tan. *Convergence Rates and Regeneration of the Block Gibbs Sampler for Bayesian Random Effects Models*. PhD thesis, Department of Statistics, University of Florida, 2009.

Martin A. Tanner and Wing-Hung Wong. The calculation of posterior distributions by data augmentation. *Journal of the Americal Statistical Association (JASA)*, 82(398):528–540, 1987.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, pages 667–685, 2010.

David van Dyk and Xiao-Li Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics (JCGS)*, 10(1):1–50, 2001.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

Minjie Xu, Jun Zhu, and Bo Zhang. Bayesian nonparametric maximum margin matrix factorization for collaborative prediction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 64–72, 2012.

Minjie Xu, Jun Zhu, and Bo Zhang. Fast max-margin matrix factorization with data augmentation. In *International Conference on Machine Learning (ICML)*, pages 978–986, 2013.

Shuanghong Yang, Jiang Bian, and Hongyuan Zha. Hybrid generative/discriminative learning for automatic image annotation. In *Uncertainty in Artificial Intelligence (UAI)*, pages 683–690, 2010.

Chun-Nam Yu and Thorsten Joachims. Learning structural SVMs with latent variables. In *International Conference on Machine Learning (ICML)*, pages 1169–1176, 2009.

Jun Zhu. Max-margin nonparametric latent feature models for link prediction. In *International Conference on Machine Learning (ICML)*, pages 719–726, 2012.

Jun Zhu and Eric P. Xing. Conditional topic random fields. In *International Conference on Machine Learning (ICML)*, pages 1239–1246, 2010.

Jun Zhu, Ning Chen, and Eric P. Xing. Infinite SVM: Dirichlet process mixtures of large-margin kernel machines. In *International Conference on Machine Learning (ICML)*, pages 617–624, 2011.

Jun Zhu, Amr Ahmed, and Eric. P. Xing. MedLDA: maximum margin supervised topic models. *Journal of Machine Learning Research (JMLR)*, (13):2237–2278, 2012.

Jun Zhu, Ning Chen, Hugh Perkins, and Bo Zhang. Gibbs max-margin topic models with fast inference algorithms. In *International Conference on Machine Learning (ICML)*, pages 124–132, 2013a.

Jun Zhu, Xun Zheng, Li Zhou, and Bo Zhang. Scalable inference in max-margin topic models. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 964–972, 2013b.

Jun Zhu, Ning Chen, and Eric P. Xing. Bayesian inference with posterior regularization and applications to infinite latent SVMs. *Journal of Machine Learning Research (JMLR, in press) (Technical Report, arXiv:1210.1766v3)*, 2014.

Journal of Machine Learning Research 15 (2014) 1111-1133 Submitted 7/12; Revised 12/12; Published 3/14

A Reliable Effective Terascale Linear Learning System

Alekh Agarwal* ALEKHA@MICROSOFT.COM
Microsoft Research
641 Avenue of the Americas
New York, NY 10011

Olivier Chapelle OLIVIER@CHAPELLE.CC
Criteo
411 High Street
Palo Alto, CA 94301

Miroslav Dudík MDUDIK@MICROSOFT.COM
John Langford JCL@MICROSOFT.COM
Microsoft Research
641 Avenue of the Americas
New York, NY 10011

Editor: Corinna Cortes

Abstract

We present a system and a set of techniques for learning linear predictors with convex losses on terascale data sets, with trillions of features,[1] billions of training examples and millions of parameters in an hour using a cluster of 1000 machines. Individually none of the component techniques are new, but the careful synthesis required to obtain an efficient implementation is. The result is, up to our knowledge, the most scalable and efficient linear learning system reported in the literature.[2] We describe and thoroughly evaluate the components of the system, showing the importance of the various design choices.

Keywords: distributed machine learning, Hadoop, AllReduce, repeated online averaging, distributed L-BFGS

1. Introduction

Distributed machine learning is a research area that has seen a growing body of literature in recent years. Much work focuses on problems of the form

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i;\, y_i) + \lambda R(\mathbf{w}), \tag{1}$$

where $\mathbf{x}_i$ is the feature vector of the i-th example, y_i is the label, $\mathbf{w}$ is the linear predictor, ℓ is a loss function and R is a regularizer. Most distributed methods for optimizing the objective (1) exploit its natural decomposability over examples, partitioning the examples over different nodes in a distributed environment such as a cluster.

∗. This work was done while all authors were part of Yahoo! Research.
1. The number of features here refers to the number of non-zero entries in the data matrix.
2. All the empirical evaluation reported in this work was carried out between May-Oct 2011.

Perhaps the simplest strategy when the number of examples n is too large for a given learning algorithm is to reduce the data set size by subsampling. However, this strategy only works if the problem is simple enough or the number of parameters is very small. The setting of interest here is when a large number of examples is really needed to learn a good model. Distributed algorithms are a natural choice for such scenarios.

It might be argued that even for these large problems, it is more desirable to explore multicore solutions developed for single machines with large amounts of fast storage and memory, rather than a fully distributed algorithm which brings additional complexities due to the need for communication over a network. Yet, we claim that there are natural reasons for studying distributed machine learning on a cluster. In many industry-scale applications, the data sets themselves are collected and stored in a decentralized fashion over a cluster, typical examples being logs of user clicks or search queries. When the data storage is distributed, it is much more desirable to also process it in a distributed fashion to avoid the bottleneck of data transfer to a single powerful server. Second, it is often relatively easy to get access to a distributed computing platform such as Amazon EC2, as opposed to procuring a sufficiently powerful server. Finally, the largest problem solvable by a single machine will always be constrained by the rate at which the hardware improves, which has been steadily dwarfed by the rate at which our data sizes have been increasing over the past decade. Overall, we think that there are several very strong reasons to explore the questions of large-scale learning in cluster environments.

Previous literature on cluster learning is broad. Several authors (Mangasarian, 1995; McDonald et al., 2010; Zinkevich et al., 2010) have studied approaches that first solve the learning problem independently on each machine using the portion of the data stored on that machine, and then average the independent local solutions to obtain the global solution. Duchi et al. (2012) propose gossip-style message passing algorithms extending the existing literature on distributed convex optimization (Bertsekas and Tsitsiklis, 1989). Langford et al. (2009) analyze a delayed version of distributed online learning. Dekel et al. (2012) consider mini-batch versions of distributed online algorithms which are extended to delay-based updates in Agarwal and Duchi (2011). A recent article of Boyd et al. (2011) describes an application of the ADMM technique for distributed learning problems. GraphLab (Low et al., 2010) is a parallel computation framework on graphs. The closest to our work are optimization approaches based on centralized algorithms with parallelized gradient computation (Nash and Sofer, 1989; Teo et al., 2007). To our knowledge, all previous versions of algorithms based on parallelized gradient computation rely on MPI implementations.[3] Finally, the large-scale learning system Sibyl (currently unpublished, but see the talks Chandra et al., 2010; Canini et al., 2012) implements a distributed boosting approach. It can be used to solve the problems of form (1) at the scales similar to those reported in this paper, but it runs on a proprietary architecture and many implementation details are missing, so doing a fair comparison is currently not possible. We attempt to compare the performance of our algorithm with the published Sibyl performance in Section 3.2.

All of the aforementioned approaches (perhaps with the exception of Sibyl) seem to leave something to be desired empirically when deployed on large clusters. In particular, their *learning throughput*—measured as the input size divided by the wall-clock running

3. See, for example, `http://www.mcs.anl.gov/research/projects/mpi/`.

time of the entire learning algorithm—is smaller than the I/O interface of a single machine for almost all parallel learning algorithms (Bekkerman et al., 2011, Part III, page 8). The I/O interface is an upper bound on the speed of the fastest single-machine algorithm since all single-machine algorithms are limited by the network interface in acquiring data. In contrast, we were able to achieve a learning throughput of 500M features/s, which is about a factor of 5 faster than the 1Gb/s network interface of any one node. This learning throughput was achieved on a cluster of 1000 nodes. Each node accessed its local examples 10 times during the course of the algorithm, so the per-node processing speeds were 5M features/s. We discuss our throughput results in more detail in Section 3.2, and contrast them with results reported for Sibyl.

Two difficulties bedevil easy parallel machine learning:

1. Efficient large-scale parallel learning algorithms must occur on a data-centric computing platform (such as Hadoop) to prevent data transfer overheads. These platforms typically do *not* support the full generality of MPI operations.

2. Existing data-centric platforms often lack efficient mechanisms for state synchronization and force both refactoring and rewriting of existing learning algorithms.

We effectively deal with both of these issues. Our system is compatible with MapReduce clusters such as Hadoop (unlike MPI-based systems) and minimal additional programming effort is required to parallelize existing learning algorithms (unlike MapReduce approaches). In essence, an existing implementation of a learning algorithm need only insert a few strategic library calls to switch from learning on one machine to learning on a thousand machines.

One of the key components in our system is a communication infrastructure that efficiently accumulates and broadcasts values across all nodes of a computation. It is functionally similar to MPI AllReduce (hence we use the name), but it takes advantage of and is compatible with Hadoop so that programs are easily moved to data, automatic restarts on failure provide robustness, and speculative execution speeds up completion. Our optimization algorithm is a hybrid online+batch algorithm with rapid convergence and only small synchronization overhead, which makes it a particularly good fit for the distributed environment.

In Section 2 we describe our approach and our communication infrastructure in more detail. The core of the paper is Section 3 where we conduct many experiments evaluating our design choices and comparing our approach with existing algorithms. In Section 4 we provide some theoretical intuition for our design, and contrast our approach with previous work. We conclude with a discussion in Section 5.

2. Computation and Communication Framework

MapReduce (Dean and Ghemawat, 2008) and its open source implementation Hadoop[4] have become the overwhelmingly favorite platforms for distributed data processing. However, the abstraction is rather ill-suited for machine learning algorithms as several researchers in the field have observed (Low et al., 2010; Zaharia et al., 2011), because it does not easily allow iterative algorithms, such as typical optimization algorithms used to solve the problem (1).

4. See, for example, `http://hadoop.apache.org/`.

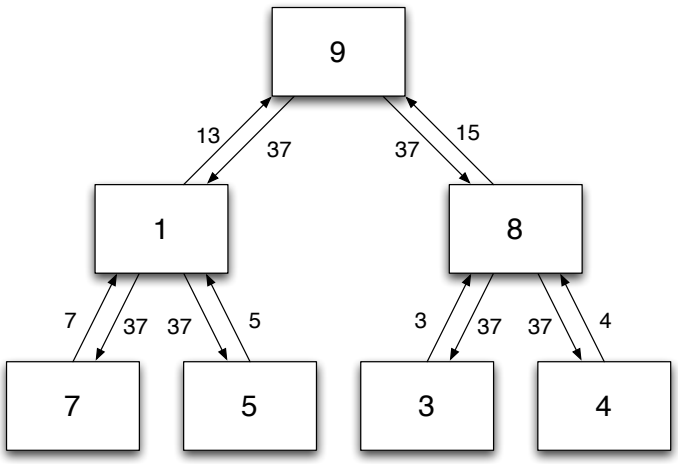


Figure 1: AllReduce operation. Initially, each node holds its own value. Values are passed up the tree and summed, until the global sum is obtained in the root node (reduce phase). The global sum is then passed back down to all other nodes (broadcast phase). At the end, each node contains the global sum.

2.1 Hadoop-compatible AllReduce

AllReduce is a more suitable abstraction for machine learning algorithms. AllReduce is an operation where every node starts with a number and ends up with the sum of the numbers across all the nodes (hence the name). A typical implementation imposes a tree structure on the communicating nodes and proceeds in two phases: numbers are first summed up the tree (the *reduce* phase) and then broadcast down to all the nodes (the *broadcast* phase), see Figure 1 for a graphical illustration. When doing summing or averaging of a long vector, such as the weight vector $\mathbf{w}$ in the optimization (1), the reduce and broadcast operations can be pipelined over the vector entries and hence the latency of going up and down the tree becomes negligible on a typical Hadoop cluster. This is the main optimization we do within the AllReduce architecture. While other (potentially more efficient or simpler) architectures for AllReduce are possible, in our experiments in Section 3 we will see that the time spent in AllReduce operation is negligible compared with the computation time and stalling time while waiting for other nodes. Therefore, we do not attempt to optimize the architecture further.

For problems of the form (1), AllReduce provides straightforward parallelization of gradient-based optimization algorithms such as gradient descent or L-BFGS—gradients are accumulated locally, and the global gradient is obtained by AllReduce. In general, any statistical query algorithm (Kearns, 1993) can be parallelized with AllReduce with only a handful of additional lines of code. This approach also easily implements averaging parameters of online learning algorithms.

An implementation of AllReduce is available in the MPI package. However, it is not easy to run MPI on top of existing Hadoop clusters (Ye et al., 2009). Moreover, MPI implements little fault tolerance, with the bulk of robustness left to the programmer.

To address the reliability issues better, we developed an implementation of AllReduce that is compatible with Hadoop. Our implementation works as follows. We initialize a spanning tree server on the gateway node to the Hadoop cluster. We then launch a map-only (alternatively reduce-only) job where each mapper processes a subset of the data. Each mapper is supplied with the IP address of the gateway node, to which it connects as the first step. Once all the mappers are launched and connected to the spanning tree server, it creates a (nearly balanced) binary tree on these nodes. Each node is given the IP addresses of its parent and child nodes in the tree, allowing it to establish TCP connections with them. All the nodes are now ready to pass messages up and down the tree. The actual communication between the nodes is all implemented directly using C++ sockets and does not rely on any Hadoop services. Implementation of AllReduce using a single tree is clearly less desirable than MapReduce in terms of reliability, because if any individual node fails, the entire computation fails. To deal with this problem, we use a simple trick described below which makes AllReduce reliable enough to use in practice for computations up to 10K node hours.

2.2 Proposed Algorithm

Our main algorithm is a hybrid online+batch approach. Pure online and pure batch learning algorithms have some desirable features, on which we build, and some drawbacks, which we overcome. For instance, an attractive feature of online learning algorithms is that they optimize the objective to a rough precision quite fast, in just a handful of passes over the data. The inherent sequential nature of these algorithms, however, makes them tricky to parallelize and we discuss the drawbacks of some of the attempts at doing so in Section 4. Batch learning algorithms such as Newton and quasi-Newton methods (e.g., L-BFGS), on the other hand, are great at optimizing the objective to a high accuracy, once they are in a *good neighborhood* of the optimal solution. But the algorithms can be quite slow in reaching this good neighborhood. Generalization of these approaches to distributed setups is rather straightforward, only requiring aggregation across nodes after every iteration, as has been noted in previous research (Teo et al., 2007).

We attempt to reap the benefits and avoid the drawbacks of both above approaches through our hybrid method. We start with each node making one online pass over its local data according to adaptive gradient updates (Duchi et al., 2010; McMahan and Streeter, 2010) modified for loss non-linearity (Karampatziakis and Langford, 2011). We notice that each online pass happens completely *asynchronously* without any communication between the nodes, and we can afford to do so since we are only seeking to get into a good neighborhood of the optimal solution rather than recovering it to a high precision at this first stage. AllReduce is used to average these weights non-uniformly according to locally accumulated gradient squares. Concretely, node k maintains a local weight vector $\mathbf{w}^k$ and a diagonal matrix $\mathbf{G}^k$ based on the gradient squares in the adaptive gradient update rule (see Algorithm 1). We compute the following weighted average over all m nodes

$$\bar{\mathbf{w}} = \left(\sum_{k=1}^{m} \mathbf{G}^k\right)^{-1} \left(\sum_{k=1}^{m} \mathbf{G}^k \mathbf{w}^k\right). \tag{2}$$

This has the effect of weighting each dimension according to how "confident" each node is in its weight (i.e., more weight is assigned to a given parameter of a given node if that node has seen more examples with the corresponding feature). We note that this averaging can indeed be implemented using AllReduce by two calls to the routine since the matrices $\mathbf{G}^k$ are only diagonal.

This solution $\bar{\mathbf{w}}$ is used to initialize L-BFGS (Nocedal, 1980) with the standard Jacobi preconditioner, with the expectation that the online stage gives us a good *warmstart* for L-BFGS. At each iteration, global gradients are obtained by summing up local gradients via AllReduce, while all the other operations can be done locally at each node. The algorithm benefits from the fast initial reduction of error provided by an online algorithm, and rapid convergence in a good neighborhood guaranteed by quasi-Newton algorithms. We again point out that the number of communication operations is relatively small throughout this process.

In addition to hybrid strategy, we also evaluate repeated online learning with averaging using the adaptive updates. In this setting, each node performs an online pass over its data and then we average the weights according to Equation 2. We average the scaling matrices similarly

$$\bar{\mathbf{G}} = \left(\sum_{k=1}^{m} \mathbf{G}^k\right)^{-1} \left(\sum_{k=1}^{m} (\mathbf{G}^k)^2\right)$$

and use this averaged state to start a new online pass over the data. This strategy is similar to those proposed by McDonald et al. (2010) and Hall et al. (2010) for different online learning algorithms. We will see in the next section that this strategy can be very effective at getting a moderately small test error very fast, but its convergence slows down and it might be too slow at reaching the optimal test error.

All strategies described above share the same processing structure. They carry out several iterations, each of which can be broken into three phases: (1) Pass through the entire local portion of the data set and accumulate the result as a vector of size d (i.e., the same size as the parameter vector). (2) Carry out AllReduce operation on a vector of size d. (3) Do some additional processing and updating of the parameter vector.

The key point to notice is that in typical applications the local data set will be orders of magnitude larger than the parameter vector, hence the communication after each pass is much more compact than transmitting the entire local data set. The second point is that each iteration is naturally a MapReduce operation. The main reason that we expect to benefit by AllReduce is because of the iterative nature of these algorithms and the shared state between iterations.

Our implementation is available as part of the open source project Vowpal Wabbit (Langford et al., 2007) and is summarized in Algorithm 2. It makes use of stochastic gradient descent (Algorithm 1) for the initial pass.

2.3 Speculative Execution

Large clusters of machines are typically busy with many jobs which use the cluster unevenly, resulting in one of a thousand nodes being very slow. To avoid this, Hadoop can speculatively execute a job on identical data, using the first job to finish and killing the other one. In our framework, it can be tricky to handle duplicates once a spanning tree topology

Algorithm 1 Stochastic gradient descent algorithm on a single node using adaptive gradient update (Duchi et al., 2010; McMahan and Streeter, 2010).

Require: Invariance update function s
(see Karampatziakis and Langford, 2011)
$\mathbf{w} = \mathbf{0}$, $\mathbf{G} = \mathbf{I}$
for all $(\mathbf{x}, y)$ in training set **do**
$\mathbf{g} \leftarrow \nabla_{\mathbf{w}} \ell(\mathbf{w}^\top \mathbf{x}; y)$
$\mathbf{w} \leftarrow \mathbf{w} - s(\mathbf{w}, \mathbf{x}, y)\mathbf{G}^{-1/2}\mathbf{g}$
$G_{jj} \leftarrow G_{jj} + g_j^2$ for all $j = 1, \ldots, d$
end for

Algorithm 2 Sketch of the proposed learning architecture

Require: Data split across nodes
for all nodes k **do**
$\mathbf{w}^k$ = result of stochastic gradient descent on the data of node k using Algorithm 1.
Compute the weighted average $\bar{\mathbf{w}}$ as in (2) using AllReduce.
Start a preconditioned L-BFGS optimization from $\bar{\mathbf{w}}$.
for $t = 1, \ldots, T$ **do**
Compute $\mathbf{g}^k$ the (local batch) gradient of examples on node k.
Compute $\mathbf{g} = \sum_{k=1}^m \mathbf{g}^k$ using AllReduce.
Add the regularization part in the gradient.
Take an L-BFGS step.
end for
end for

is created for AllReduce. For this reason, we delay the initialization of the spanning tree until each node completes the first pass over the data, building the spanning tree on only the speculative execution survivors. The net effect of this speculative execution trick is perhaps another order of magnitude of scalability and reliability in practice. Indeed, we found the system reliable enough for up to 1000 nodes running failure-free for hundreds of trials (of typical length up to 2 hours). This level of fault tolerance highlights the benefits of a Hadoop-compatible implementation of AllReduce. We will show the substantial gains from speculative execution in mitigating the "slow node" problem in the experiments.

3. Experiments

In this section we will present the empirical evaluation of the system described so far. We begin by describing the datasets used, before evaluating the various properties of our framework both from a systems as well as a machine learning perspective. A more theoretical evaluation of our approach can be found in the next section.

3.1 Data Sets

We evaluated our framework on two datasets. The first dataset is a computational advertising task, with proprietary data from Yahoo! Inc. The second dataset comes from the task of recognizing a human acceptor splice site. Both the datasets are large enough to necessitate

distributed machine learning, and simple approaches such as subsampling do not work to obtain a good model with reasonable predictive accuracy as we describe in detail next.

3.1.1 Display Advertising

In online advertising, given a user visiting a publisher page, the problem is to select the best advertisement for that user. A key element in this matching problem is the click-through rate (CTR) estimation: what is the probability that a given ad will be clicked on, given some context (user, page visited)? Indeed, in a cost-per-click (CPC) campaign, the advertiser only pays when the ad gets clicked, so even modest improvements in predictive accuracy directly affect revenue.

Training data contains user visits, which either resulted in a click on the ad (positive examples with $y_i = 1$), or did not result in a click (negative examples with $y_i = 0$). We estimate the click probabilities by logistic regression with L_2 regularization. The regularization coefficient is chosen from a small set to obtain the best test performance. The user visit is represented by binary indicator features encoding the user, page, ad, as well as conjunctions of these features. Some of the features include identifiers of the ad, advertiser, publisher and visited page. These features are hashed (Weinberger et al., 2009) and each training example ends up being represented as a sparse binary vector of dimension 2^{24} with around 125 non-zero elements. Let us illustrate the construction of a conjunction feature with an example. Imagine that an ad from `etrade` was placed on `finance.yahoo.com`. Let h be a 24 bit hash of the string "`publisher=finance.yahoo.com and advertiser=etrade`". Then the (publisher, advertiser) conjunction is encoded by setting to 1 the h-th entry of the feature vector for that example.

Since the data is unbalanced (low CTR) and because of the large number of examples, we subsample the negative examples resulting in a class ratio of about 2 negatives for 1 positive, and use a large test set drawn from days later than the training set. There are 2.3B examples in the training set. More characteristics of this data set and modeling details can be found in Chapelle et al. (2013).

3.1.2 Splice Site Recognition

The problem consists of recognizing a human acceptor splice site (Sonnenburg and Franc, 2010). We consider this learning task because this is, as far as we know, the largest public data set for which subsampling is not an effective learning strategy. Sonnenburg et al. (2007) introduced the *weighted degree kernel* to learn over DNA sequences. They also proposed an SVM training algorithm for that kernel for which learning over 10M sequences took 24 days. Sonnenburg and Franc (2010) proposed an improved training algorithm, in which the weight vector—in the feature space induced by the kernel—is learned, but the feature vectors are never explicitly computed. This resulted in a faster training: 3 days with 50M sequences.

We solve this problem by L_2-regularized logistic regression. Again, the regularization coefficient is chosen from a small set to optimize test set performance. We follow the same experimental protocol as in Sonnenburg and Franc (2010): we use the same training and test sets of respectively 50M and 4.6M samples. We also consider the same kernel of degree $d = 20$ and hash size $\gamma = 12$. The feature space induced by this kernel has

	1%	10%	100%
auROC	0.8178	0.8301	0.8344
auPRC	0.4505	0.4753	0.4856
NLL	0.2654	0.2582	0.2554

Table 1: Test performance on the display advertising problem as a function of the subsampling rate, according to three metrics: area under ROC curve (auROC), area under precision/recall curve (auPRC), and negative log likelihood (NLL).

dimensionality 11,725,480. The number of non-zero features per sequence is about 3,300. Unlike Sonnenburg and Franc (2010), we explicitly compute the feature space representation of the examples, yielding about 3TB of data. This explicit representation is a *disadvantage* we impose on our method to simplify implementation.

3.2 Results

We now present a detailed evaluation of our system on both the above datasets. We begin by evaluating the performance of simpler heuristics such as subsampling the data for faster running time, before examining the computational and communication aspects of our system in detail.

3.2.1 Effect of Subsampling

The easiest way to deal with a very large training set is to reduce it by subsampling as discussed in the introduction. Sometimes similar test errors can be achieved with smaller training sets and there is no need for large-scale learning. For splice site recognition, Table 2 of Sonnenburg and Franc (2010) shows that smaller training sets do hurt the area under the precision/recall curve on the test set.

For display advertising, we subsampled the data at 1% and 10%. The results in Table 1 show that there is a noticeable drop in accuracy after subsampling. Note that even if the drop does not appear large at a first sight, it can cause a substantial loss of revenue. Thus, for both data sets, the entire training data set is needed to achieve optimal performances.

The three metrics reported in Table 1 are area under the ROC curve (auROC), area under the precision/recall curve (auPRC) and negative log-likelihood (NLL). Since auPRC is the most sensitive metric, we report test results using that metric in the rest of the paper. This is also the metric used in Sonnenburg and Franc (2010).

3.2.2 Running Time

We ran 5 iterations of L-BFGS on the splice site data with 1000 nodes. On each node, we recorded for every iteration the time spent in AllReduce and the computing time—defined as the time not spent in AllReduce. The time spent in AllReduce can further be divided into stall time—waiting for the other nodes to finish their computation—and communication time. The communication time can be estimated by taking the minimum value of the AllReduce times across nodes.

	5%	50%	95%	Max	Comm. time
Without spec. execution	29	34	60	758	26
With spec. execution	29	33	49	63	10

Table 2: Distribution of computing times (in seconds) over 1000 nodes with and without speculative execution. First three columns are quantiles. Times are average per iteration (excluding the first one) for the splice site recognition problem.

Nodes	100	200	500	1000
Comm time / pass	5	12	9	16
Median comp time / pass	167	105	43	34
Max comp time / pass	462	271	172	95
Wall clock time	3677	2120	938	813

Table 3: Computing times to obtain a fixed test error on the splice site recognition data, using different numbers of nodes. The first 3 rows are averages per iteration (excluding the first pass over the data).

The distribution of the computing times is of particular interest because the speed of our algorithm depends on the slowest node. Statistics are shown in Table 2. It appears that computing times are concentrated around the median, but there are a few outliers. Without speculative execution, one single node was about 10 times slower than the other nodes; this has the catastrophic consequence of slowing down the entire process by a factor 10. The table shows that the use of speculative execution successfully mitigates this issue.

We now study the running time as a function of the number of nodes. For the display advertising problem, we varied the number of nodes from 10 to 100 and computed the speed-up factor relative to the run with 10 nodes. In each case, we measured the amount of time needed to get to a fixed test error. Since there can be significant variations from one run to the other—mostly because of the cluster utilization—each run was repeated 10 times. Results are reported in Figure 2. We note that speculative execution was not turned on in this experiment, and we expect better speed-ups with speculative execution. In particular, we expect that the main reason for the departure from the ideal speed-up curve is the "slow node" problem (as opposed to the aspects of the AllReduce communication implementation), which is highlighted also in the next experiment.

Table 3 shows the running times for attaining a fixed test error as a function of the number of nodes on the splice site recognition problem. Unlike Figure 2, these timing experiments have not been repeated and thus there is a relatively large uncertainty in their expected values. It can be seen from Tables 2 and 3 that even with as many as 1000 nodes, communication is not the bottleneck. One of the main challenges instead is the "slow node" issue. This is mitigated to some degree by speculative execution, but as the number of nodes increases, so does the likelihood of hitting slow nodes.

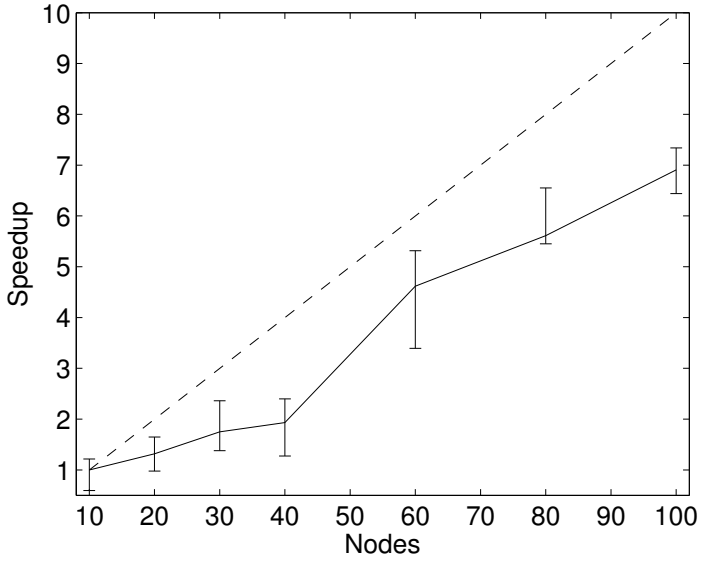


Figure 2: Speed-up for obtaining a fixed test error, on the display advertising problem, relative to the run with 10 nodes, as a function of the number of nodes. The dashed line corresponds to the ideal speed-up, the solid line is the average speed-up over 10 repetitions, and the bars indicate maximum and minimum values.

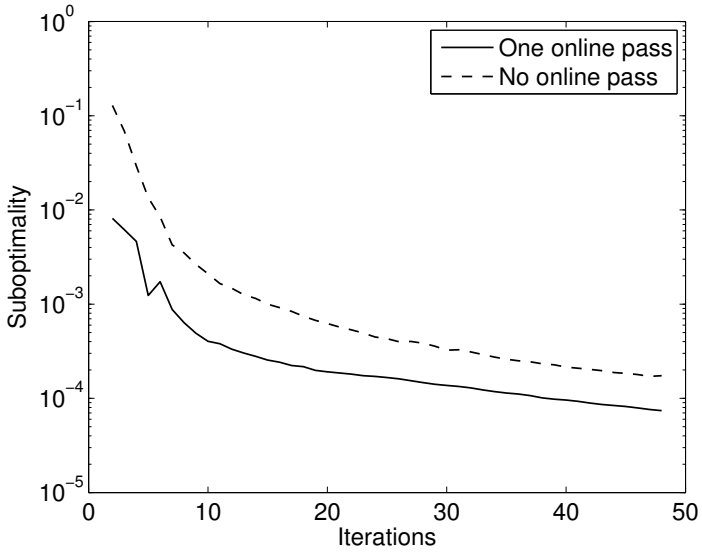


Figure 3: Effect of initializing the L-BFGS optimization by an average solution from online runs on individual nodes. Suboptimality is the difference between the objective value on the training data and the optimal value obtained by running the algorithm to convergence.

3.2.3 Large Experiment and Comparison with Sibyl

We also experimented with an 8 times larger version of the display advertising data (16B examples). Using 1000 nodes and 10 passes over the data, the training took only 70 minutes.[5] Since each example is described by 125 non-zero features, the average processing speed was

$$16\text{B} \times 10 \times 125 \text{ features/1000 nodes/70 minutes } = \text{ 4.7 M features/node/s } .$$

The overall learning throughput was

$$16\text{B} \times 125 \text{ features/70 minutes } = \text{ 470 M features/s } .$$

We briefly compare this with a result reported for the distributed boosting system Sibyl for a run on 970 cores (Canini et al., 2012, slide 24). The run was done over 129.1B examples, with 54.61 non-zero features per example. The reported processing speed was 2.3M features/core/s (which is a factor of two slower than our achieved processing speed). The reported number of iterations was 10–50, which would lead to the final learning throughput in the range 45–223 M features/s, that is, the result appears to be slower by a factor of 2–10.

3.2.4 Online and Batch Learning

We now investigate the speed of convergence of three different learning strategies: batch, online and hybrid. We are interested in how fast the algorithms minimize the training objective as well as the test error.

Figure 3 compares how fast the two learning strategies—batch with and without an initial online pass—optimize the training objective. It plots the optimality gap, defined as the difference between the current objective function and the optimal one (i.e., the minimum value of the objective (1)), as a function of the number iterations. From this figure we can see that the initial online pass results in a saving of about 10–15 iterations.

Figure 4 shows the test auPRC on both data sets as a function of the number of iterations for 4 different strategies: only online learning, only L-BFGS learning, and 2 hybrid methods consisting of 1 or 5 passes of online learning followed by L-BFGS optimization. L-BFGS with one online pass appears to be the most effective strategy.

For the splice site recognition problem, an initial online pass and 14 L-BFGS iterations yield an auPRC of 0.581, which is just a bit higher than results of Sonnenburg and Franc (2010). This was achieved in 1960 seconds using 500 machines, resulting in a 68 speed-up factor (132,581 seconds on a single machine reported in Table 2 of Sonnenburg and Franc, 2010). This seems rather poor compared with the ideal 500 speed-up factor, but recall that we used explicit feature representation which creates a significant overhead.

3.3 Comparison with Previous Approaches

In order to better assess the overall system, we next compare its performance with that of some other published baselines. We start by demonstrating the efficacy of AllReduce

5. As mentioned before, there can be substantial variations in timing between different runs; this one was done when the cluster was not too occupied.

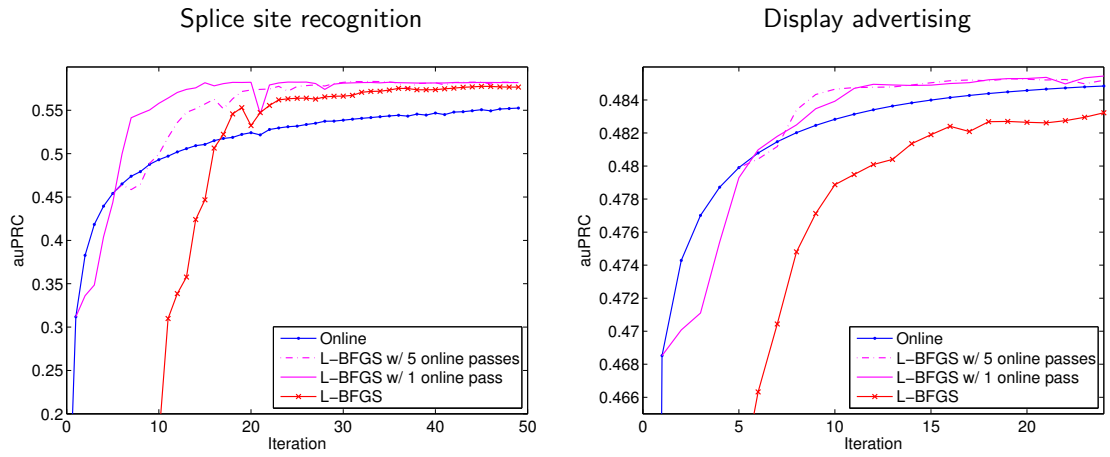


Figure 4: Test auPRC for 4 different learning strategies. Note that the online and hybrid curves overlap during the warmstart phase (of either 1 or 5 online passes).

	Full size	10% sample
MapReduce	1690	1322
AllReduce	670	59

Table 4: Average training time per iteration of an internal logistic regression implementation using either MapReduce or AllReduce for gradients aggregation. The data set is the display advertising one and a subset of it.

compared with MapReduce, before comparing to some other distributed machine learning algorithms.

3.3.1 AllReduce vs. MapReduce

The standard way of using MapReduce for iterative machine learning algorithms is the following (Chu et al., 2007): every iteration is a MapReduce job where the mappers compute some local statistics (such as a gradient) and the reducers sum them up. This is ineffective because each iteration has large overheads (job scheduling, data transfer, data parsing, etc.). We have an internal implementation of such a MapReduce algorithm. We updated this code to use AllReduce instead and compared both versions of the code in Table 4. This table confirms that Hadoop MapReduce has substantial overheads since the training time is not much affected by the data set size. The speed-up factor of AllReduce over Hadoop's MapReduce can become extremely large for smaller data sets, and remains noticeable even for the largest data sets. It is also noteworthy that *all* algorithms described in Chu et al. (2007) can be parallelized with AllReduce, plus further algorithms such as parameter averaging approaches.

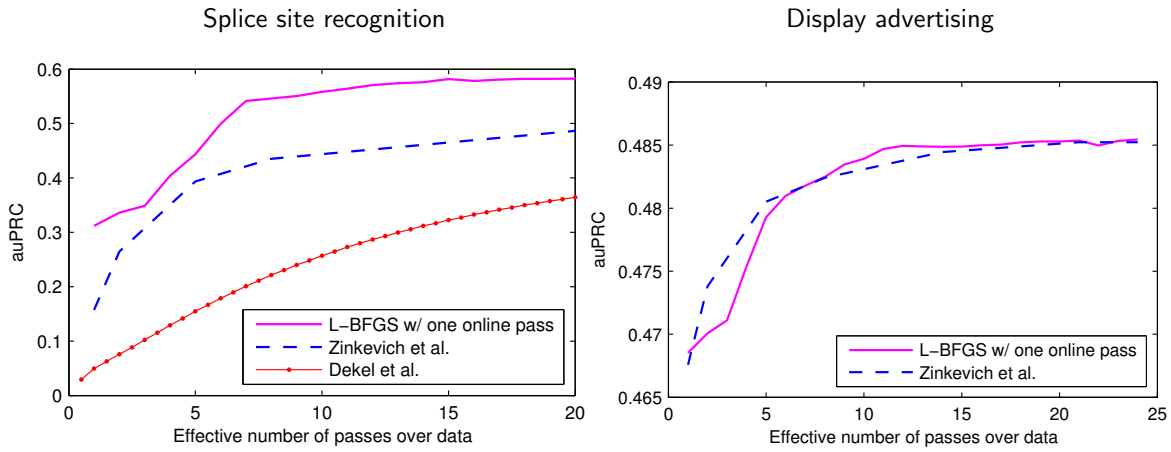


Figure 5: Test auPRC for different learning strategies as a function of the effective number of passes over data. In L-BFGS, it corresponds to iterations of the optimization. In overcomplete SGD with averaging (Zinkevich et al.), it corresponds to the replication coefficient.

3.3.2 Overcomplete Average

We implemented oversampled stochastic gradient with final averaging (Zinkevich et al., 2010), and compared its performance to our algorithm. We used stochastic gradient descent with the learning rate in the t-th iteration as

$$\eta_t = \frac{1}{L + \gamma\sqrt{t}} \ .$$

We tuned γ and L on a small subset of the data set.

In Figure 5, we see that the oversampled SGD is competitive with our approach on the display advertising data, but its convergence is much slower on splice site data.

3.3.3 Parallel Online Mini-batch

Dekel et al. (2012) propose to perform online convex optimization using stochastic gradients accumulated in small mini-batches across all nodes. We implemented a version of their algorithm using AllReduce. They suggest global minibatch sizes of no more than $b \propto \sqrt{n}$. On m nodes, each node accumulates gradients from b/m examples, then an AllReduce operation is carried out, yielding the mini-batch gradient, and each node performs a stochastic gradient update with the learning rate of the form

$$\eta_t = \frac{1}{L + \gamma\sqrt{t/m}} \ .$$

We tuned L and γ on a smaller data set. In Figure 5, we report the results on splice site data set, using 500 nodes, and mini-batch size $b = 100$k. Twenty passes over the data thus corresponded to 10k updates. Due to the overwhelming communication overhead

associated with the updates, the overall running time was 40 hours. In contrast, L-BFGS took less than an hour to finish 20 passes while obtaining much superior performance. The difference in the running time between 1h and 40h is solely due to communication. Thus, in this instance, we can conservatively conclude that the communication overhead of 10k mini-batch updates is 39 hours.

We should point out that it is definitely possible that the mini-batched SGD would reach similar accuracy with much smaller mini-batch sizes (for 10k updates theory suggests we should use mini-batch sizes of at most 10k), however, the 39 hour communication overhead would remain. Using larger mini-batches, we do expect that the time to reach 20 passes over data would be smaller (roughly proportional to the number of mini-batch updates), but according to theory (as well as our preliminary experiments on smaller subsets of splice site data), we would have inferior accuracy. Because of the prohibitive running time, we were not able to tune and evaluate this algorithm on display advertising data set.

4. Communication and Computation Complexity

The two key performance characteristics of any distributed algorithm are its communication and computation complexity. The aim of this section is to discuss the complexity of our approach and to compare it with previous solutions. We hope to clarify the reasons underlying our design choices and explain the scalability of our system. We start with a discussion of computational considerations.

4.1 Computational Complexity of the Hybrid Approach

In this section, we explain the convergence properties of the hybrid approach and compare it with other optimization strategies. In order to have a clean discussion, we make some simplifying assumptions. We consider the case of only one online pass at each node. Furthermore, we restrict ourselves to the case of uniform averaging of weights. Similar analysis does extend to the non-uniform weighting scheme that we use, but the details are technical and provide no additional intuition. Before we embark on any details, it should be clear that the hybrid approach is always convergent, owing to the convergence of L-BFGS. All the online learning step initially does is to provide a good warmstart to L-BFGS. This section aims to provide theoretical intuition why the gains of such a warmstart can be substantial in certain problem regimes.

Let $\tilde{\ell}(\mathbf{w};\, \mathbf{x},\, y) = \ell(\mathbf{w}^\top \mathbf{x};\, y) + \lambda R(\mathbf{w})/n$ be the regularized loss function. We analyze a scaled version of the objective (1):

$$f(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i;\, y_i) + \frac{\lambda}{n} R(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \tilde{\ell}(\mathbf{w};\, \mathbf{x}_i,\, y_i) \ .$$

We assume that the cluster is comprised of m nodes, with a total of n data examples distributed uniformly at random across these nodes. Let us denote the local objective function at each node as f^k:

$$f^k(\mathbf{w}) = \frac{m}{n} \sum_{i \in S_k} \tilde{\ell}(\mathbf{w};\, \mathbf{x}_i,\, y_i)$$

where S_k is the set of n/m examples at node k. Note that the global objective $f = (\sum_{k=1}^m f^k)/m$ is the average of the local objectives. We observe that owing to our random data split, we are guaranteed that

$$\mathbb{E}[f^k(\mathbf{w})] = \mathbb{E}[f(\mathbf{w})] = \mathbb{E}_{\mathbf{x},y}\Big[\tilde{\ell}(\mathbf{w};\, \mathbf{x},\, y)\Big]$$

for each k, where the expectation is taken over the distribution from which our examples are drawn. In order to discuss the convergence properties, we need to make a couple of standard assumptions regarding the functions f^k. First, we assume that the functions f^k are differentiable, with Lipschitz-continuous gradients. We also assume that each f^k is strongly convex, at least in a local neighborhood around the optimum. We note that these assumptions are unavoidable for the convergence of quasi-Newton methods such as L-BFGS.

To understand how many passes over the data are needed for the hybrid approach to minimize f to a precision ϵ, we first analyze the online learning pass at each node. In this pass, we compute a weight vector $\mathbf{w}^k$ by performing n/m steps of stochastic gradient descent or some variant thereof (Duchi et al., 2010; Karampatziakis and Langford, 2011). Since we performed only one pass at each node, the resulting $\mathbf{w}^k$ at each node approximately minimizes $\mathbb{E}[f^k] = \mathbb{E}_{\mathbf{x},y}[\tilde{\ell}]$ to a precision ϵ^k (for the methods we use, we expect $\epsilon^k = \mathcal{O}(\sqrt{m/n})$). Let us now denote the uniform average $\bar{\mathbf{w}} = \sum_{k=1}^m \mathbf{w}^k/m$. For this approach, a direct application of Jensen's inequality yields

$$\begin{aligned}\mathbb{E}_{\mathbf{x},y}\Big[\tilde{\ell}(\bar{\mathbf{w}};\, \mathbf{x}, y)\Big] &= \mathbb{E}_{\mathbf{x},y}\left[\tilde{\ell}\left(\frac{\sum_{i=1}^m \mathbf{w}^k}{m};\, \mathbf{x}, y\right)\right] \leq \frac{1}{m}\sum_{k=1}^m \mathbb{E}_{\mathbf{x},y}\Big[\tilde{\ell}(\mathbf{w}^k;\, \mathbf{x}, y)\Big] \qquad (3)\\ &\leq \min_{\mathbf{w}} \mathbb{E}_{\mathbf{x},y}\Big[\tilde{\ell}(\mathbf{w};\, \mathbf{x}, y)\Big] + \frac{1}{m}\sum_{k=1}^m \epsilon^k = \min_{\mathbf{w}} \mathbb{E}_{\mathbf{x},y}\Big[\tilde{\ell}(\mathbf{w};\, \mathbf{x}, y)\Big] + \mathcal{O}\left(\sqrt{\frac{m}{n}}\right) \ .\end{aligned}$$

Furthermore, standard sample complexity arguments (see, e.g., Bartlett and Mendelson, 2002; Devroye et al., 1996) allow us to bound the function value $f(\mathbf{w})$ for an arbitrary $\mathbf{w}$ as

$$\Big|f(\mathbf{w}) - \mathbb{E}_{\mathbf{x},y}\Big[\tilde{\ell}(\mathbf{w};\, \mathbf{x},\, y)\Big]\Big| \leq \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) \ . \qquad (4)$$

Let $\mathbf{w}^*$ be the minimizer of the training loss function f. Then we can combine the above inequalities as

$$\begin{aligned} f(\bar{\mathbf{w}}) &\leq \mathbb{E}_{\mathbf{x},y}\Big[\tilde{\ell}(\bar{\mathbf{w}};\, \mathbf{x},\, y)\Big] + \mathcal{O}(1/\sqrt{n})\\ &\leq \min_{\mathbf{w}} \mathbb{E}_{\mathbf{x},y}\Big[\tilde{\ell}(\mathbf{w};\, \mathbf{x},\, y)\Big] + \mathcal{O}(\sqrt{m/n})\\ &\leq \mathbb{E}_{\mathbf{x},y}\Big[\tilde{\ell}(\mathbf{w}^*;\, \mathbf{x},\, y)\Big] + \mathcal{O}(\sqrt{m/n})\\ &\leq f(\mathbf{w}^*) + \mathcal{O}(\sqrt{m/n})\end{aligned}$$

where the first inequality follows by (4), the second by (3), and the fourth by (4). For the remainder of the discussion, we denote the overall suboptimality of $\bar{\mathbf{w}}$ relative to $\mathbf{w}^*$ by $\epsilon_0 = \mathcal{O}(\sqrt{m/n})$.

Switching over to the L-BFGS phase, we assume that we are in the linear convergence regime of L-BFGS (Liu and Nocedal, 1989). We denote the contraction factor by κ, so that the number of additional L-BFGS passes over data needed to minimize f to a precision ϵ is at most

$$\kappa \log \frac{\epsilon_0}{\epsilon} \ .$$

Compared to initializing L-BFGS without any warmstart, our hybrid strategy amounts to overall savings of

$$\kappa \log \frac{1}{\epsilon} - \left(1 + \kappa \log \frac{\epsilon_0}{\epsilon}\right) = \kappa \log \frac{1}{\epsilon_0} - 1 = \mathcal{O}\left(\frac{\kappa}{2} \log \frac{n}{m}\right) - 1$$

passes over data. In typical applications, we expect $n \gg m$ to ensure that computation amortize the cost of communication. As a result, the improvement due to the warmstart can be quite substantial just like we observed in our experiments. Furthermore, this part of our argument is in no way specific to the use of L-BFGS as the batch optimization algorithm. Similar reasoning holds for any reasonable (quasi)-Newton method.

We could also consider the alternative choice of just using parallel online learning without ever switching to a batch optimization method. The theoretical results in this area, however, are relatively harder to compare with the hybrid approach. For general online learning algorithms, previous works study just one local pass of online learning followed by averaging (McDonald et al., 2010), which typically cannot guarantee an error smaller than ϵ_0 in our earlier notation. The repeated averaging approach, discussed and analyze for the specific case of perceptron algorithm in earlier work (McDonald et al., 2010), works well in our experiments on the computational advertising task but does not have easily available convergence rates beyond the special case of separable data and the perceptron algorithm. Nevertheless, one appeal of the hybrid approach is that it is guaranteed to be competitive with such online approaches, by the mere virtue of the first online phase.

Overall, we see that the hybrid approach will generally be competitive with purely online or batch approaches in terms of the computational complexity. As a final point, we discuss two extreme regimes where it can and does offer substantial gains. The first regime is when the data has a significant noise level. In such a scenario, the level ϵ of optimization accuracy desired is typically not too large (intuitive statistical arguments show no reduction in generalization error for $\epsilon \ll 1/n$). Setting $\epsilon = 1/n$ for a clean comparison, we observe that the total number of passes for the hybrid method is at most

$$1 + \frac{\kappa}{2}(\log(m) + \log(n)),$$

as opposed to $\kappa \log(n)$ for just pure batch optimization. When $m \ll n$, this shows that the online warmstart can cut down the number of passes almost by a factor of 2. We do note that in such high noise regimes, pure online approaches can often succeed, as we observed with our advertising data.

The second extreme is when our data is essentially noiseless, so that the desired accuracy ϵ is extremely small. In this case, the relative impact of the online warmstart can be less pronounced (it is certainly strictly better still) over an arbitrary initialization of L-BFGS. However, as we saw on our splice site recognition data, on this extreme, the online learning methods will typically struggle since they are usually quite effective in fitting the data to

Algorithm	Per-node communication cost
Bundle method (Teo et al., 2007)	$O(dT_{\text{bundle}})$
Online with averaging (McDonald et al., 2010; Hall et al., 2010)	$O(dT_{\text{online}})$
Parallel online (Hsu et al., 2011)	$O(ns/m + nT_{\text{online}'})$
Overcomplete online with averaging (Zinkevich et al., 2010)	$O(ns + d)$
Distr. minibatch (dense) (Dekel et al., 2012; Agarwal and Duchi, 2011)	$O(dT_{\text{mini}}n/b) = O(dT_{\text{mini}}\sqrt{n})$
Distr. minibatch (sparse) (Dekel et al., 2012; Agarwal and Duchi, 2011)	$O(bsT_{\text{mini}}n/b) = O(nsT_{\text{mini}})$
Hybrid online+batch	$O(dT_{\text{hybrid}})$

Table 5: Communication cost of various learning algorithms. Here n is the number of examples, s is the number of nonzero features per example, d is the number of dimensions, T is the number of times the algorithm examines each example, and b is the minibatch size (in minibatch algorithms).

moderate but not high accuracies (as evident from their $1/\epsilon$ or $1/\epsilon^2$ convergence rates). Overall, we find that even on these two extremes, the hybrid approach is competitive with the better of its two components.

4.2 Communication Cost Comparison with Previous Approaches

In the previous section we discussed the computational complexity of several techniques with an identical communication pattern: communication of the entire weight vector once per pass. In this section we contrast our approach with techniques that use other communication patterns. We focus mainly on communication cost since the computational cost is typically the same as for our algorithm, or the communication dominates the computation.

Since modern network switches are quite good at isolating communicating nodes, the most relevant communication cost is the maximum (over nodes) of the communication cost of a single node.

Several variables (some of them recalled from the previous section) are important:

1. m the number of nodes.
2. n the total number of examples across all nodes.
3. s the number of nonzero features per example.
4. d the parameter dimension.
5. T the number of passes over the examples.

In the large-scale applications that are subject of this paper, we typically have $s \ll d \ll n$, where both d and n are large (see Section 3.1).

The way that data is dispersed across a cluster is relevant in much of this discussion since an algorithm not using the starting format must pay the communication cost of redistributing the data. We assume the data is distributed across the nodes uniformly according to an example partition, as is common.

The per-node communication cost of the hybrid algorithm is $\Theta(dT_{\text{hybrid}})$ where T_{hybrid} is typically about 15 to maximize test accuracy in our experiments. Note that the minimum

possible communication cost is $\Theta(d)$ if we save the model on a single machine. There is no communication involved in getting data to workers based on the data format assumed above. An important point here is that every node has a communication cost functionally smaller than the size of the data set, because there is no dependence on ns.

Similar to our approach, Teo et al. (2007) propose a parallel batch optimization algorithm (specifically, a bundle method) using the MPI implementation of AllReduce. This approach arrives at an accurate solution with $O(dT_{\text{bundle}})$ communication per node. Our approach improves over this in several respects. First, as Figure 4 demonstrates, we obtain a substantial boost thanks to our warmstarting strategy, hence in practice we expect $T_{\text{bundle}} > T_{\text{hybrid}}$. The second distinction is in the AllReduce implementation. Our implementation is well aligned with Hadoop and takes advantage of speculative execution to mitigate the slow node problem. On the other hand, MPI assumes full control over the cluster, which needs to be carefully aligned with Hadoop's MapReduce scheduling decisions, and by itself, MPI does not provide robustness to slow nodes.

Batch learning can also be implemented using MapReduce on a Hadoop cluster (Chu et al., 2007), for example in the Mahout project.[6] Elsewhere it has been noted that MapReduce is not well suited for iterative machine learning algorithms (Low et al., 2010; Zaharia et al., 2011). Evidence of this is provided by the Mahout project itself, as their implementation of logistic regression is not parallelized. Indeed, we observe substantial speed-ups from a straightforward substitution of MapReduce by AllReduce on Hadoop. It is also notably easier to program with AllReduce, as code does not require refactoring.

The remaining approaches are based on online convex optimization. McDonald et al. (2010) and Hall et al. (2010) study the approach when each node runs an online learning algorithm on its examples and the results from the individual nodes are averaged. This simple method is empirically rather effective at creating a decent solution. The communication cost is structurally similar to our algorithm $\Theta(dT_{\text{online}})$ when T_{online} passes are done. However, as we saw empirically in Figure 4 and also briefly argued theoretically in Section 4.1, $T_{\text{online}} > T_{\text{hybrid}}$.

Similarly to these, Zinkevich et al. (2010) create an overcomplete partition of the data and carry out separate online optimization on each node followed by global averaging. Our experiments show that this algorithm can have competitive convergence (e.g., on display advertising data), but on more difficult optimization problems it can be much slower than the hybrid algorithm we use here (e.g., on splice site recognition data). This approach also involves deep replication of the data—for example, it may require having 1/4 of all the examples on each of 100 nodes. This is generally undesirable with large data sets. The per-node communication cost is $\Theta(nsT_{\text{rep}}/m + d)$ where T_{rep} is the level of replication and m is the number of nodes. Here, the first term comes from the data transfer required for creating the overcomplete partition whereas the second term comes from parameter averaging. Since T_{rep}/m is often a constant near 1 (0.25 was observed by Zinkevich et al., 2010, and the theory predicts only a constant factor improvement), this implies the communication cost is $\Theta(ns)$, the size of the data set.

Other authors have looked into online mini-batch optimization (Dekel et al., 2012; Agarwal and Duchi, 2011). The key problem here is the communication cost. The per-node

6. For Mahout, see `http://mahout.apache.org/`.

communication cost is $\Theta(T_{\text{mini}}dn/b)$ where b is the minibatch size (number of examples per minibatch summed across all nodes), T_{mini} is the number of passes over the data, n/b is the number of minibatch updates per pass and d is the number of parameters. Theory suggests $b \leq \sqrt{n}$, implying communication costs of $\Theta(T_{\text{mini}}d\sqrt{n})$. While for small minibatch sizes T_{mini} can be quite small (plausibly even smaller than 1), when d is sufficiently large, this communication cost is prohibitively large. This is the reason for the slow performance of mini-batched optimization that we observed in our experiments. Reworking these algorithms with sparse parameter updates, the communication cost per update becomes bs yielding an overall communication cost of $\Theta(T_{\text{mini}}ns)$, which is still several multiples of the data set size. Empirically, it has also been noted that after optimizing learning rate parameters, the optimal minibatch size is often 1 (Hsu et al., 2011).

Another category of algorithms is those which use online learning with a *feature based* partition of examples (Hsu et al., 2011; Dean et al., 2012). The advantage of this class of algorithms is that they can scale to a very large number of parameters, more than can be fit in the memory of a single machine. Several families of algorithms have been tested in Hsu et al. (2011) including delayed updates, minibatch, second-order minibatch, independent learning, and backpropagation. The per-node communication costs differ substantially here. Typical communication costs are $\Theta(ns/m + nT_{\text{online'}})$ where the first term is due to shuffling from an example-based format, and the second term is for the run of the actual algorithm. The complexity of our approach is superior to this strategy since $n \gg d$.

5. Discussion

We have shown that a new architecture for parallel learning based on a Hadoop-compatible implementation of AllReduce can yield a combination of accurate prediction and short training time in an easy programming style. The hybrid algorithm we employ allows us to benefit from the rapid initial optimization of online algorithms and the high precision of batch algorithms where the last percent of performance really matters. Our experiments reveal that each component of our system is critical in driving the performance benefits we obtain. Specifically, Table 4 and Figure 3 show the performance gains resulting from our use of AllReduce and the warmstart of the L-BFGS algorithm. The effectiveness of our overall system, as compared to the previous approaches, is confirmed in Figure 5. Two issues we do not discuss in this paper are the overheads of data loading and node scheduling within Hadoop. These issues can indeed affect the performance, but we found that they typically get amortized since they are one-time overheads in the AllReduce approach as opposed to per-iteration overheads in MapReduce. Nonetheless, improvements in the scheduling algorithms can further improve the overall performance of our system.

Our paper carefully considers various design choices affecting the communication and computation speeds of a large-scale linear learning system, drawing from and building upon the available techniques in the literature. The resulting system enables the training of linear predictors on data sets of size unmatched in previous published works. In particular, the results demonstrate the effectiveness of our system compared to other alternatives in the literature. We believe that this provides a very strong and natural baseline which we previously found lacking in the literature on distributed machine learning. The conceptual

simplicity of our framework, and the open-source implementation should further help other researchers in comparing with and building on our system.

References

A. Agarwal and J. Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems 24*. 2011.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

R. Bekkerman, M. Bilenko, and J. Langford. A tutorial on scaling up machine learning. Technical report, KDD, 2011. URL `http://hunch.net/~large_scale_survey/`.

D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., 1989.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 2011.

K. Canini, T. Chandra, E. Ie, J. McFadden, K. Goldman, M. Gunter, J. Harmsen, K. LeFevre, D. Lepikhin, T. L. Llinares, I. Mukherjee, F. Pereira, J. Redstone, T. Shaked, and Y. Singer. Sibyl: A system for large scale supervised machine learning. In *MLSS Santa Cruz*, 2012. URL `http://users.soe.ucsc.edu/~niejiazhong/slides/chandra.pdf`. A short presentation.

T. Chandra, E. Ie, K. Goldman, T. L. Llinares, J. McFadden, F. Pereira, J. Redstone, T. Shaked, and Y. Singer. Sibyl: a system for large scale machine learning. In *LADIS 2010: The 4th ACM SIGOPS/SIGACT Workshop on Large Scale Distributed Systems and Middleware*, 2010. URL `http://www.magicbroom.info/Papers/Ladis10.pdf`. A keynote talk.

O. Chapelle, E. Manavoglu, and R. Rosales. Simple and scalable response prediction for display advertising. *Transactions on Intelligent Systems and Technology*, 2013. In press.

C.T. Chu, S.K. Kim, Y.A. Lin, Y.Y. Yu, G. Bradski, A.Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *Advances in Neural Information Processing Systems 19*, volume 19, page 281, 2007.

J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51:107–113, 2008.

J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. A. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems 25*, pages 1232–1240. 2012.

O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13:165–202, 2012.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2010.

J.C. Duchi, A. Agarwal, and M.J. Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *Automatic Control, IEEE Transactions on*, 57(3):592–606, 2012.

K. Hall, S. Gilpin, and G. Mann. Mapreduce/bigtable for distributed optimization. In *Workshop on Learning on Cores, Clusters, and Clouds*, 2010.

D. Hsu, N. Karampatziakis, J. Langford, and A. Smola. Parallel online learning. In *Scaling Up Machine Learning*, 2011.

N. Karampatziakis and J. Langford. Online importance weight aware updates. In *Uncertainty in Artificial Intelligence*, 2011.

M. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, 1993.

J. Langford, L. Li, and A. Strehl. Vowpal wabbit open source project. Technical report, Yahoo!, 2007.

J. Langford, A. Smola, and M. Zinkevich. Slow learners are fast. In *Advances in Neural Information Processing Systems 22*, 2009.

D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Graphlab: A new framework for parallel machine learning. In *Uncertainty in Artificial Intelligence*, 2010.

O. L. Mangasarian. Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Optimization*, 33:1916–1925, 1995.

R. McDonald, K. Hall, and G. Mann. Distributed training strategies for the structured perceptron. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2010.

H. B. McMahan and M. Streeter. Adaptive bound optimization for online convex optimization. In *Proceedings of the Twenty Third Annual Conference on Computational Learning Theory*, 2010.

S. G. Nash and A. Sofer. Block truncated-newton methods for parallel optimization. *Mathematical Programming*, 45:529–546, 1989.

J. Nocedal. Updating quasi-Newton matrices with limited storage. *Math. Comp.*, 35(151): 773–782, 1980.

S. Sonnenburg and V. Franc. COFFIN: a computational framework for linear SVMs. In *International Conference on Machine Learning*, 2010.

S. Sonnenburg, G. Rätsch, and K. Rieck. Large scale learning with string kernels. In *Large Scale Kernel Machines*, pages 73–103. 2007.

C. Teo, Q. Le, A. Smola, and SVN Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2007.

K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *International Conference on Machine Learning*, 2009.

J. Ye, J.-H. Chow, J. Chen, and Z. Zheng. Stochastic gradient boosted distributed decision trees. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management*, 2009.

M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Technical Report UCB/EECS-2011-82, EECS Department, University of California, Berkeley, 2011.

M. Zinkevich, M. Weimer, A. Smola, and L. Li. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems 23*. 2010.

Journal of Machine Learning Research 15 (2014) 1135-1176 Submitted 10/12; Revised 9/13; Published 3/14

New Learning Methods for Supervised and Unsupervised Preference Aggregation

Maksims N. Volkovs MVOLKOVS@CS.TORONTO.EDU
Richard S. Zemel ZEMEL@CS.TORONTO.EDU
University of Toronto
40 St. George Street
Toronto, ON M5S 2E4

Editor: William Cohen

Abstract

In this paper we present a general treatment of the preference aggregation problem, in which multiple preferences over objects must be combined into a single consensus ranking. We consider two instances of this problem: unsupervised aggregation where no information about a target ranking is available, and supervised aggregation where ground truth preferences are provided. For each problem class we develop novel learning methods that are applicable to a wide range of preference types.[1] Specifically, for unsupervised aggregation we introduce the Multinomial Preference model (MPM) which uses a multinomial generative process to model the observed preferences. For the supervised problem we develop a supervised extension for MPM and then propose two fully supervised models. The first model employs SVD factorization to derive effective item features, transforming the aggregation problems into a learning-to-rank one. The second model aims to eliminate the costly SVD factorization and instantiates a probabilistic CRF framework, deriving unary and pairwise potentials directly from the observed preferences. Using a probabilistic framework allows us to directly optimize the expectation of any target metric, such as NDCG or ERR. All the proposed models operate on pairwise preferences and can thus be applied to a wide range of preference types. We empirically validate the models on rank aggregation and collaborative filtering data sets and demonstrate superior empirical accuracy.

Keywords: preference aggregation, meta-search, learning-to-rank, collaborative filtering

1. Introduction

Many areas of study, such as information retrieval (IR), collaborative filtering, and social web analysis face the preference aggregation problem, in which multiple preferences over objects must be combined into a single consensus ranking. Early developments in preference aggregation and analysis originated in social science (Arrow, 1951) and statistics (Luce, 1959), giving rise to the field of social choice. Research in social choice concentrates on measuring individual interests, values, and/or welfare as an aggregate towards collective decision. Common problems explored in this field include vote aggregation in elections and other domains as well as player/team ranking based on observed game outcomes. Most of these problems are relatively small in size and can be analyzed thoroughly, resulting in models that have well-explored properties and theoretical guarantees. These models

1. The code for all models introduced in this paper is available at `www.cs.toronto.edu/~mvolkovs`.

now decide the outcomes of such crucial events as presidential and government elections as well as legal decisions. However, the theoretical guarantees for many of these models typically come at the expense of complicated inference and/or strong assumptions about the preference data (Chevaleyre et al., 2007; Rossi et al., 2011).

The recent explosion of web technologies has generated immense amounts of new preference data. Several properties of this data make it difficult to apply many of the existing aggregation models. First, the ease with which people can access and generate content on the web has resulted in a drastic increase in the quantity of data. For instance, where before the majority of sports data had on the order of a thousand players that participated in several tournaments per year, now, online gaming has millions of users that participate in tens of millions of games daily. Recent statistics on the popular game Halo indicate that over 2 billion multiplayer games are played annually,[2] with hundreds of thousands of games happening at any given moment. Consequently, while the aggregation problem in online gaming remains similar to the traditional one in sports—combine preference in the form of game outcomes to generate reliable estimates of players' skills—any model developed for this task now has to be able to process large amounts of data quickly and handle diverse evidence types ranging from one-on-one games to elimination team tournaments.

Second, the diversity of online applications has led to many new preference types. In addition to the common direct evidence in the form of votes and ratings we now also have a variety of indirect evidence, including web page clicks, dwell time (time spent on a page), and viewing patterns. While these evidence forms do not directly indicate preference, when aggregated across many users, they have been found to closely correlate with it (Joachims, 2002; Joachims et al., 2007). Methods that mine these preferences are now extensively used in search engine optimization (Agichtein et al., 2006; Joachims et al., 2007; Guo et al., 2009) and other domains. Moreover, for some of the new problems the preferences are no longer generated by people. An example of this is meta-search where an issued query is sent to several search engines and the (often partial) document rankings returned by them are aggregated by the meta-search engine to generate more comprehensive ranking results. In this problem there is no human interaction and all the preferences are generated by the machines. Consequently social theories on user behavior and models based on these theories are less applicable here.

Finally, new types of aggregation problems have also recently emerged. In the past the majority of the aggregation problems were unsupervised, that is, no ground truth preference information about the items was available. For these problems the aim is typically to produce a ranking that satisfies as many of the observed preferences (majority or another related objective) as possible. Due to the popularity of such problems almost all of the existing research in preference aggregation has concentrated on the unsupervised aggregation. However, many of the recent problems are amenable to the supervised setting, as ground truth preference information is available. The meta-search problem mentioned above is one example of supervised preference aggregation. Often, to train/evaluate the aggregating function the documents retrieved by the search engines are given to human annotators who assign relevance labels to each document. The relevance labels provide ground truth preference information about the documents, that is, the documents with higher relevance

2. Full article can be found at `http://www.pcmag.com/article2/0,2817,2402479,00.asp`.

label are to be ranked above those with lower one. Another example is crowdsourcing (e.g., MechanicalTurk), where tasks often involve assigning ratings to objects or pairs of objects ranging from images to text. The ratings from several users are then aggregated to produce a single labeling of the data. To ensure consistency in generated labels a domain expert typically labels a subset of the data shown to the "crowd". The labels are then used to evaluate the quality of annotations submitted by each worker. In these problems the aim is not to satisfy the majority but rather to learn a mapping from the observed preferences to the ground truth ones. Consequently, methods that aim to satisfy the majority often lead to suboptimal results since they lack the "specialization" property: they cannot identify cases where the majority is wrong and only a small subset of the preferences should be used. Such a property is impossible to achieve without referring to the ground truth labels.

The scale and variety of the preference data generated by the social and other web domains discussed above show that the field of preference aggregation is rapidly evolving and expanding. Almost every user-oriented web application ranging from web shops and social networks to web search and gaming is now using preference aggregation techniques, the accuracy of which has a direct and significant impact on the generated revenue and business decisions. There is thus an evident need to develop effective aggregation methods that are able to scale to the large web data sets and handle diverse preference types.

As the field has evolved a new trend has recently emerged where machine learning methods are starting to be used to automatically learn the aggregating models. While these methods typically lack the theoretical support of the social choice models they often show excellent empirical performance and are able to handle large and diverse preference data. These models have now been applied successfully to preference aggregation problems in collaborative filtering (Gleich and Lim, 2011; Jiang et al., 2011; Guiver and Snelson, 2009), information retrieval (Cormack et al., 2009; Liu et al., 2007b; Chen et al., 2011) and online gaming (Dangauthier et al., 2007), as well as others. Inspired by these results the work presented in this paper also takes a machine learning approach and develops new models for both supervised and unsupervised preference aggregation problems. In the following sections we describe existing approaches and open challenges for both problems. We then introduce and empirically validate new models for each problem type.

2. Unsupervised Preference Aggregation

Unsupervised preference aggregation is the problem of combining multiple preferences over objects into a single consensus ranking when no ground truth preference information is available. As mentioned above, the majority of research in preference aggregation has concentrated on this problem and a number of models have been developed. Given the underlying correspondence between ranking and permutation, considerable work on unsupervised preference aggregation has exploited probabilistic models on permutations, many of which originate in statistics and psychology. Mallows (Mallows, 1957) and Plackett-Luce (Plackett, 1975; Luce, 1959) are particularly popular models, each with many extensions (Guiver and Snelson, 2009; Quin et al., 2010; Lu and Boutilier, 2011). However, research has largely concentrated on learning a consensus ranking based on a set of observed full, or partial rankings. These models are thus inadequate for problems where preferences are

expressed in other forms, and where inconsistencies exist in the observed preferences, such as "a beat b", "b beat c", and "c beat a".

In this section we address this problem by developing a flexible probabilistic model over pairwise comparisons. Pairwise comparisons are the building blocks of almost all forms of evidence about preference and subsume the most general models of evidence proposed in literature. Our model can thus be applied to a wide spectrum of preference aggregation problems and does not impose any restrictions on the type of evidence. The score-based approach that we adopt allows for rapid learning and inference, which makes the model applicable to large-scale aggregation problems. We experimentally validate our model on a rank aggregation and collaborative filtering tasks using Microsoft's LETOR4.0 (Liu et al., 2007a) and the MovieLens (Herlocker et al., 1999) data sets.

2.1 Framework

We assume a set of N instances where for every instance n we have a set of M_n items $\mathbf{X}_n = \{x_{n1}, ..., x_{nM_n}\}$ and a set of Ψ experts. Each expert $\psi \in \{1, ..., \Psi\}$ generates a list of preferences for items in $\mathbf{X}_n$. We assume that the same set of experts generate preferences for items in each instance. The preferences can be in the form of full or partial rankings, top-K lists, ratings, relative item comparisons, or combinations of these. All of these forms can be converted to a set of *partial pairwise preferences*, which in most cases will be neither complete nor consistent. We use $\{x_{ni} \succ x_{nj}\}$ to denote the preference of x_{ni} over x_{nj}. We allow the same pairwise preferences to occur multiple times, and use the pairwise count matrix $\mathbf{Y}_n^{\psi}(i,j) : M_n \times M_n$ to count the number of times preference $\{x_{ni} \succ x_{nj}\}$ is produced by the expert ψ, with $\mathbf{Y}_n^{\psi}(i,j) = 0$ if $\{x_{ni} \succ x_{nj}\}$ is not expressed by ψ.

The most straightforward way to convert rankings into pairwise preferences is through binary comparisons. Given two rankings r_{ni}^{ψ} and r_{nj}^{ψ} assigned by ψ to x_{ni} and x_{nj} we set $\mathbf{Y}_n^{\psi}(i,j) = I[r_{ni}^{\psi} < r_{nj}^{\psi}]$ where I is an indicator function, similarly $\mathbf{Y}_n^{\psi}(j,i) = I[r_{nj}^{\psi} < r_{ni}^{\psi}]$. This representation, however, completely ignores the strength of preference expressed by the magnitude of the rankings. For example, the partial ranking $\{1, 200, 300\}$ will have the same count matrix as the ranking $\{1, 2, 3\}$, but the first ranking expresses significantly more confidence about the ordering of the items than the second one. To account for this we instead use $\mathbf{Y}_n^{\psi}(i,j) = (r_{nj}^{\psi} - r_{ni}^{\psi})I[r_{ni}^{\psi} < r_{nj}^{\psi}]$ and $\mathbf{Y}_n^{\psi}(j,i) = (r_{ni}^{\psi} - r_{nj}^{\psi})I[r_{nj}^{\psi} < r_{ni}^{\psi}]$. In this form we assume that ranking $\{r_{ni} = 1, r_{nj} = 200\}$ is equivalent to observing the pairwise preference $\{x_{ni} \succ x_{nj}\}$ 199 times, whereas ranking $\{r_{ni} = 1, r_{nj} = 2\}$ is equivalent to observing $\{x_{ni} \succ x_{nj}\}$ only once. This method of accounting for preference strength is not new and the reader can refer to Gleich and Lim (2011) and Jiang et al. (2011) for more extensive treatment of this and other approaches for converting rankings to pairwise matrices. We summarize these pairwise preference representations below:

1. **Binary Comparison**:
$$\mathbf{Y}_n^{\psi}(i,j) = I[r_{ni}^{\psi} < r_{nj}^{\psi}],$$

2. **Rank Difference**:
$$\mathbf{Y}_n^{\psi}(i,j) = (r_{nj}^{\psi} - r_{ni}^{\psi})I[r_{ni}^{\psi} < r_{nj}^{\psi}].$$

A ranking of items in $\mathbf{X}_n$ can be represented as a permutation of $\mathbf{X}_n$. A permutation π is a bijection $\pi : \{1, ..., M_n\} \rightarrow \{1, ..., M_n\}$ mapping each item x_{ni} to its rank $\pi(i) = j$, and $i = \pi^{-1}(j)$. Given the observed (partial) preference instance n consisting of count matrices $\mathbf{Y}_n = \{\mathbf{Y}_n^1, ..., \mathbf{Y}_n^\Psi\}$ the goal is to come up with a single ranking π of items in $\mathbf{X}_n$ that maximally satisfies this instance.

Most preference aggregation problems fit this framework. For instance in meta-search instances correspond to queries and $\mathbf{X}_n$ is the set of documents retrieved for a given query q_n. Each expert ψ represents a search engine which generates either partial or complete ranking of the documents in $\mathbf{X}_n$. As before we can let $\mathbf{Y}_n^\psi(i,j) = (r_{nj}^\psi - r_{ni}^\psi) I[r_{ni}^\psi < r_{nj}^\psi]$ if documents x_{ni} and x_{nj} are both ranked by the search engine ψ and set $\mathbf{Y}_n^\psi(i,j) = 0$ otherwise. In collaborative filtering $\mathbf{X}$ is the set of movies/songs/books etc., and an instance of the rank aggregation problem aims to infer the consensus ranking of movies given the (partial) ratings of users ψ (Guiver and Snelson, 2009; Gleich and Lim, 2011). The pairwise approach provides a natural way to model this problem. We can define $\mathbf{Y}^\psi(i,j) = (\ell_i^\psi - \ell_j^\psi) I[\ell_i^\psi > \ell_j^\psi]$ where ℓ_i and ℓ_j are the ratings assigned to movies x_{ni} and x_{nj} by user ψ. If ψ did not rate either x_{ni} or x_{nj} we set $\mathbf{Y}^\psi(i,j) = 0$.

2.2 Previous Work

Relevant previous work in this area can be divided into two categories: permutation based and score based. In this section we describe both types of models.

2.2.1 Permutation-Based Models

Permutation based models work directly in the permutation space. The most common and well explored such model is the Mallows model (Mallows, 1957). Mallows defines a distribution over permutations and is typically parametrized by a central permutation σ and a dispersion parameter $\phi \in (0, 1]$; the probability of a permutation π is given by:

$$P(\pi|\phi,\sigma) = \frac{1}{Z(\phi,\sigma)} \phi^{-\mathcal{D}(\pi,\sigma)},$$

where $\mathcal{D}(\pi,\sigma)$ is a distance between π and σ. For rank aggregation problems inference in this model amounts to finding the permutation σ that maximizes the likelihood of the observed rankings. For some distance metrics, such as Kendall's τ and Spearman's rank correlation, the partition function $Z(\phi,\sigma)$ can be found exactly. However, finding the central permutation σ that maximizes the likelihood is typically very difficult and in many cases is intractable (Meila et al., 2007).

Recent work extends the Mallows model to define distributions over partial rankings (Lu and Boutilier, 2011). Under partial rankings the partition function can no longer be computed exactly, so these authors introduced a new sampling approach to estimate it. When the number of items is large, however, this sampling approach is typically very slow, which makes the model impractical for many large scale online problems such as meta-search where aggregation has to be done very quickly. Furthermore, both the proposed pairwise model and the sampling approach rely on the assumption that all pairwise preferences are consistent, which is often violated in real-world preference aggregation problems.

A number of other generalizations of the Mallows model such as the Cranking model (Lebanon and Lafferty, 2002) the Aggregation model (Klementiev et al., 2008), and the CPS model (Quin et al., 2010). The Cranking model extends the Mallows distribution to model several diverse preference profiles. Each preference profile i is modeled by its own central permutation σ_i and "importance" θ_i:

$$P(\pi|\boldsymbol{\sigma},\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\sigma},\boldsymbol{\theta})} e^{-\sum_i \theta_i \mathcal{D}(\pi,\sigma_i)},$$

where $\boldsymbol{\sigma} = \{\sigma_i\}$ and $\boldsymbol{\theta} = \{\theta_i\}$ are the profile parameters. The above model can be viewed as a mixture of Mallows models where each component is parametrized by (σ_i, θ_i) pair. Using this alternative representation the work of Klementiev et al. (2008) further generalizes the Cranking model to partial top-K lists and derives an Expectation Minimization (EM) algorithm to learn the parameters $\boldsymbol{\theta}$.

Another recent extension of the Mallows model, the CPS model, defines a sequential generative process, similar to the Plackett-Luce model described below, which draws the items without replacement to form a permutation; the probability of a given permutation π is:

$$P(\pi|\sigma,\phi) = \prod_{i=1}^{M} \frac{\exp(-\theta \sum_{\pi_{1:i}} \mathcal{D}(\pi_{1:i},\sigma))}{Z(i,\pi)},$$

where the summation in the numerator is over all permutations $\pi_{1:i}$ that have the first i elements fixed to π; $Z(i,\pi)$'s are the normalizing constants that ensure that $\sum_\pi P(\pi|\sigma,\phi) = 1$. For several distance metrics such as Spearman's rank correlation and footrule as well as Kendall's τ, the summation $\sum_{\pi_{1:i}} \mathcal{D}(\pi_{1:i},\sigma)$ over $(M-i)!$ elements, can be found in $O(M^2)$, allowing the normalizing constants $Z(i,\pi)$ to be computed in polynomial time. However, during inference one must still consider nearly all of the $M!$ possible permutations to find an optimal π. A greedy approximation avoids this search, which reduces the complexity to $O(M^2)$, but provides no guarantee with respect to the optimal solution.

In general, due to the extremely large search space (typically $M!$ for M items) and the discontinuity of functions over permutations, exact inference in permutation-based models is often intractable. Thus one must resort to approximate inference methods, such as sampling or greedy approaches, often without guarantees on how close the approximate solution will be to the target optimal one. As the number of items grows, the cost of finding a good approximation increases significantly, which makes the majority of these models impractical for many real world applications where data collections are extremely large. The score-based approach described next avoids this problem by working with real valued scores instead.

2.2.2 Score-Based Models

In score-based approaches the goal is to learn a set of real valued scores (one per item) $\mathbf{S}_n = \{s_{n1}, ..., s_{nM_n}\}$ which are then used to sort the items. Working with scores avoids the discontinuity problems of the permutation space.

Early score based methods for rank aggregation in meta-search are heuristic based. For example, BordaCount (Aslam and Montague, 2001), Condorcet (Montague and Aslam, 2002) and median rank aggregation (Fagin et al., 2003) derive the item scores by averaging

ranks across the experts or counting the number of pairwise wins. In statistics a very popular pairwise score model is the Bradley-Terry (Bradley and Terry, 1952) model:

$$P(\mathbf{Y}_n^{\psi}|\mathbf{S}_n) = \prod_{i \neq j} \left(\frac{\exp(s_{ni})}{\exp(s_{ni}) + \exp(s_{nj})} \right)^{\mathbf{Y}_n^{\psi}(i,j)},$$

where $\frac{\exp(s_{ni})}{\exp(s_{ni})+\exp(s_{nj})}$ can be interpreted as the probability that item x_{ni} beats x_{nj} in the pairwise contest. In logistic form the Bradley-Terry model is very similar to another popular pairwise model, the Thurstone model (Thurstone, 1927). Extensions of these models include the Elo Chess rating system (Elo, 1978), adopted by the World Chess Federation FIDE in 1970, and Microsoft's TrueSkill (Dangauthier et al., 2007) rating system for player matching in online games, used extensively in Halo and other games. Furthermore, the popular supervised learning-to-rank model RankNet (Burges et al., 2005) is also based on this approach.

The key assumption behind the Bradley-Terry model is that the pairwise probabilities are completely independent of the items not included in the pair. A problem that arises from this assumption is that if a given item x_{ni} has won all pairwise contests, the likelihood becomes larger as s_{ni} becomes larger. It follows that a maximum likelihood estimate for s_{ni} is ∞ (Mase, 2003). As a consequence the model will always produce a tie amongst all undefeated items. Often this is an unsatisfactory solution because the contests that the undefeated items participated in, and their opponents' strengths, could be significantly different.

To avoid some of these drawbacks, the Bradley-Terry model was generalized by Plackett and Luce (Plackett, 1975; Luce, 1959) to a model for permutations:

$$P(\pi|\mathbf{S}_n) = \prod_{i=1}^{M_n} \frac{\exp(\mathbf{S}_n(\pi^{-1}(i)))}{\sum_{j=i}^{M_n} \exp(\mathbf{S}_n(\pi^{-1}(j)))},$$

where $\mathbf{S}_n(\pi^{-1}(i))$ is the score of the item in position i in π. The generative process behind the Plackett-Luce model assumes that items are selected sequentially without replacement. Initially item $\pi^{-1}(1)$ is selected from the set of M_n items and placed first, then item $\pi^{-1}(2)$ is selected from the remaining $M_n - 1$ items and placed second and so on until all M_n items are placed. Note that here inference can be done quickly by doing simple gradient descent on scores, which is a clear advantage over most permutation based models. The Plackett-Luce generalization relaxes the independence assumption of the Bradley-Terry model but this model is only applicable to consistent full or partial rankings (or consistent pairwise preferences) which significantly limits its application. Moreover, for 2-item rankings the Plackett-Luce model reduces to the Bradley-Terry model and thus suffers from the same infinite score problem. To overcome this problem a Bayesian framework was also recently introduced for the Plackett-Luce model by placing a Gamma prior on the selection probabilities (Guiver and Snelson, 2009). The authors of that work demonstrated that the Bayesian approach prevented overfitting and produced aggregate rankings that better fitted the observed preference data. This improvement however, comes at the cost of significant computational overhead required during score inference. In this work we show that the

model we develop achieves similar improvement over the Plackett-Luce model without the additional computational overhead and preference type restrictions.

Recently several score based approaches have been developed to model the joint pairwise matrix (Gleich and Lim, 2011; Jiang et al., 2011). In these methods the preferences expressed by each of the experts are combined into a single preference matrix $\mathbf{Y}_n^{tot} = \sum_{\psi=1}^{\Psi} \mathbf{Y}_n^{\psi}$, which is then factorized by a low rank factorization such as:

$$\mathbf{Y}_n^{tot} \approx \mathbf{S}_n \mathbf{e}^T - \mathbf{e}\mathbf{S}_n^T.$$

The resulting scores $\mathbf{S}_n$ are then used to rank the items. The main drawback of this approach is that by combining all preferences into a single $\mathbf{Y}_n^{tot}$ the individual expert information is lost. Consequently outlier experts with preferences substantially deviating from the consensus can significantly influence both $\mathbf{Y}_n^{tot}$ and the resulting scores.

2.3 Multinomial Preference Model (MPM)

In this section we develop a new score based model for pairwise preferences, the Multinomial Preference Model (MPM) (Volkovs and Zemel, 2012). A key motivating idea behind our approach is that when absolute preferences such as rankings are converted into pairwise counts using the rank difference approach described above, we interpret the resulting counts as conveying two forms of information: a binary preference, simply based on which item is ranked higher, and a confidence, based on the magnitude of the rank difference. Consider for example three items x_1, x_2 and x_3 with ranks $r_1 = 1$, $r_2 = 2$ and $r_3 = 3$ respectively. Figure 1(a) shows the resulting count matrix after these ranks are converted to pairwise preferences. Item x_1 is preferred to both x_2 and x_3 with $\mathbf{Y}(1,2) = r_2 - r_1 = 1$ and $\mathbf{Y}(1,3) = r_3 - r_1 = 2$, x_2 is preferred only to x_3 with $\mathbf{Y}(2,3) = r_3 - r_2 = 1$, and x_3 is not preferred to any item. Note that preference $\{x_1 \succ x_3\}$ where both items are at the extremes of the ranking has the largest rank difference and consequently the biggest count.

Now consider the second example with partial ranking $r_1 = 30$, $r_2 = 20$ and $r_3 = 1$ yielding the pairwise count matrix shown in Figure 1(b). Comparing this with the previous example we see that the preference $\{x_3 \succ x_1\}$ with items at the extremes of the ranking also has the highest count, however in this case we are significantly more certain of it. The count $\mathbf{Y}(3,1) = 29$ is considerably higher than the highest count from the previous example, strongly indicating that x_3 should be placed above x_1. The two examples demonstrate how large values of $\mathbf{Y}(i,j)$ may be interpreted as providing more evidence to conclude that $\{x_i \succ x_j\}$ is correct.

In MPM we model the count matrix $\mathbf{Y}_n^{\psi}$ as an outcome of multiple draws from the joint consensus distribution Q_n over pairwise preferences defined by the scores $\mathbf{S}_n$. For instance in the second example above after observing $\mathbf{Y}$ we can infer that $P(x_3 \succ x_1)$ should have the most mass under Q. We use $\mathbf{B}_n$ to denote the random variable distributed as Q_n. A draw from Q_n can be represented as a vector $\mathbf{b}_{ij}$ of length $M_n * (M_n - 1)$ (all possible pairs), with 1 on the entry corresponding to preference $\{x_{ni} \succ x_{nj}\}$ and zeros everywhere else, that is, a one-hot encoding. Given $\mathbf{S}_n$ we define the consensus distribution as follows:

Definition 1 *The consensus distribution* $Q_n = \{P(\mathbf{B}_n = \mathbf{b}_{ij}|\mathbf{S}_n)\}_{i \neq j}$ *is a collection of pairwise probabilities* $P(\mathbf{B}_n = \mathbf{b}_{ij}|\mathbf{S}_n)$*, where* $P(\mathbf{B}_n = \mathbf{b}_{ij}|\mathbf{S}_n) = \frac{\exp(s_{ni} - s_{nj})}{\sum_{k \neq l} \exp(s_{nk} - s_{nl})}$.

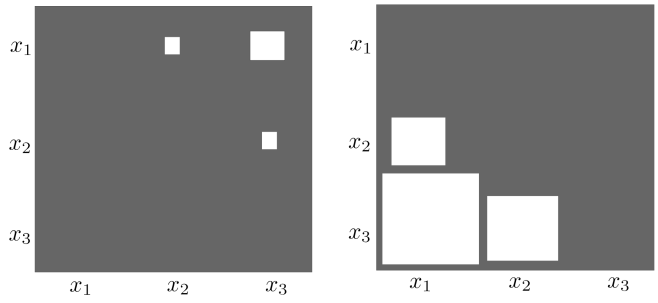


Figure 1: Figure 1(a) displays the count matrix with the contests won by each of the 3 items x_1, x_2 and x_3 after their ranking $\{r_1 = 1, r_2 = 2, r_3 = 3\}$ is converted to pairwise counts using the rank difference method. A count is displayed in each (x_i, x_j) entry if $r_i < r_j$, and the size of the square represents the count magnitude. Figure 1(b) shows the same matrix for the ranking $\{r_1 = 30, r_2 = 20, r_3 = 1\}$.

Q_n defines a multinomial distribution over pairwise preferences. Parametrization through $\mathbf{S}_n$ controls the shape of Q_n, lending considerable flexibility in distributions over preferences, which can be tailored to many different problems. To generate the observed aggregated counts $\mathbf{Y}_n^\psi$ we assume that T_n^ψ independent samples are drawn from Q_n where $T_n^\psi = \sum_{i \neq j} \mathbf{Y}_n^\psi(i,j)$ so that:

$$\mathbf{Y}_n^\psi(i,j) = \sum_{t=1}^{T_n^\psi} I[\mathbf{B}_n = \mathbf{b}_{ij}^{(t)}],$$

where $I[\mathbf{B}_n = \mathbf{b}_{ij}^{(t)}]$ is 1 if preference $\{x_{ni} \succ x_{nj}\}$ was sampled on the t'th draw and 0 otherwise. Under this model the probability of the observed counts is given by:

$$\begin{aligned} P(\mathbf{Y}_n^\psi | \mathbf{S}_n) &= \frac{T_n^\psi!}{\prod_{i \neq j} \mathbf{Y}_n^\psi(i,j)!} \prod_{i \neq j} P(\mathbf{B}_n = \mathbf{b}_{ij} | \mathbf{S}_n)^{\mathbf{Y}_n^\psi(i,j)} \\ &= \frac{T_n^\psi!}{\prod_{i \neq j} \mathbf{Y}_n^\psi(i,j)!} \prod_{i \neq j} \left(\frac{\exp(s_{ni} - s_{nj})}{\sum_{k \neq l} \exp(s_{nk} - s_{nl})} \right)^{\mathbf{Y}_n^\psi(i,j)}. \end{aligned}$$

Note that in MPM the pairwise probabilities depend on the entire item set $\mathbf{X}$ and the observed counts matrix is modeled jointly. The magnitude of the score s_{ni} is directly related to the count $\mathbf{Y}_n^\psi(i,j)$. When the scores are fitted via maximum likelihood the gradient of the log probability with respect to s_{ni} is given by:

$$\frac{\partial \log(P(\mathbf{Y}_n^\psi | \mathbf{S}_n))}{\partial s_{ni}} = \left(\sum_j \mathbf{Y}_n^\psi(i,j) - \sum_j \mathbf{Y}_n^\psi(j,i) \right) - T_n^\psi \left(\frac{\partial \log(\sum_{k \neq l} e^{s_{nk} - s_{nl}})}{\partial s_{ni}} \right). \quad (1)$$

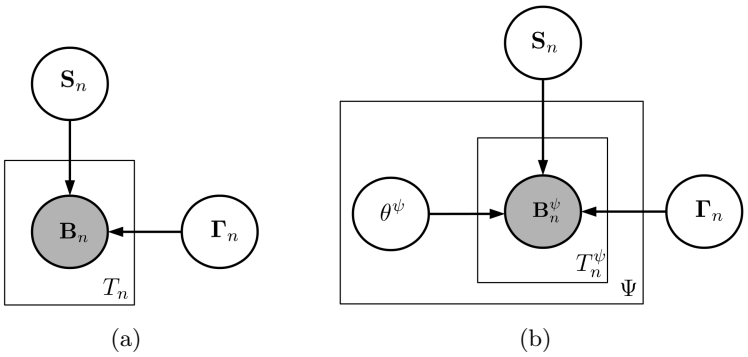


Figure 2: Graphical model representation of the generative process in MPM and its θ extension for a single instance n. Here $T_n^{\psi} = \sum_{i \neq j} \mathbf{Y}_n^{\psi}(i,j)$ is the total number of preferences observed for expert ψ and $T_n = \sum_{\psi=1}^{\Psi} T_n^{\psi}$ is the total number of preferences across all experts for instance n.

Note that when x_{ni} is strongly preferred to other items the first term in Equation 1 will be large leading to an increase in s_{ni}. This will in turn raise the probability of preferences where x_{ni} beats the other items. Raising the probability for some preferences must simultaneously lower it for others since the probabilities always sum to 1. The second term, the derivative of the partition function, accounts for this. The scores thus compete with each other and the ones with the most positive/negative evidence get pushed to the extremes. This is exactly the effect we wanted to achieve because it will allow us to accurately model the count matrices as illustrated by the toy examples above. In contrast with MPM, in the Bradley-Terry model there is no joint interaction amongst scores and pairs are modeled independently so a single preference is sufficient to push the score to infinity.

2.4 Incorporating Prediction Confidence

In the base MPM model it is difficult to judge the model's *confidence* for a given score combination. Aside from the relative score magnitudes, it is hard to measure the uncertainty associated with the score assigned to each item and the aggregate ranking that the scores impose. Such a measure can be very useful during inference and can influence the decision process. For instance, it can be used to further filter and/or reorder the items in the aggregate ranking. Moreover, for problems where the accuracy is extremely important, the recommender system can inform the user if the produced ranking has high/low degree of uncertainty.

To address this problem we introduce a set of *variance* parameters $\mathbf{\Gamma}_n = \{\gamma_{n1}, ..., \gamma_{nM_n}\}$, $\gamma_{ni} > 0 \ \forall i$. Each γ_{ni} models the uncertainty associated with the score s_{ni} inferred for the item x_{ni}. The consensus distribution now becomes:

$$P(\mathbf{B}_n = \mathbf{b}_{ij} | \mathbf{S}_n, \mathbf{\Gamma}_n) = \frac{\exp((s_{ni} - s_{nj})/(\gamma_{ni} + \gamma_{nj}))}{\sum_{k \neq l} \exp((s_{nk} - s_{nl})/(\gamma_{nk} + \gamma_{nl}))}.$$

Note that the probability of x_{ni} beating x_{nj} decreases (increases) if the variance for *either* x_{ni} or x_{nj} increases (decreases). Through $\mathbf{\Gamma}_n$ we can effectively express the variance over the preferences for each item x_{ni} and translate this variance into uncertainty over pairwise probabilities. Moreover, measures such as the average variance, $\bar{\gamma}_n = \frac{1}{M_n}\sum_{i=1}^{M}\gamma_{ni}$, can be used to infer the variance for the entire aggregate ranking produced by the model.

In this setting the γ's can either be learned in combination with scores via maximum likelihood or set using some inference procedure. The generative process for MPM with both $\mathbf{S}_n$ and $\mathbf{\Gamma}_n$ parameters is shown in Figure 2(a).

2.5 Modelling Deviations from the Consensus

The assumption in MPM that the preferences generated by the Ψ experts are independent and identically distributed is likely to be false in many domains. Often one would expect to find preferences which either completely or partially deviate from the general consensus. For example in collaborative filtering most people tend to like popular movies such as *Harry Potter* and *Forrest Gump*, but in almost all cases one can find a number of outlier users who would give these movies low ratings. Assuming that the preferences of the outliers have the same distribution as the consensus, as is done in the base MPM model, can skew the aggregation especially if the outliers are severe.

To introduce the notion of outliers into our model we define an additional set of *adherence* parameters $\mathbf{\Theta} = \{\theta^1, ..., \theta^\Psi\}$, $\theta^\psi \in [0, 1]$. Here we assume that each expert ψ has its own distribution over preferences Q_n^ψ, whose adherence to the global consensus distribution Q_n (see Definition 1) is described by θ^ψ. Associated with each expert ψ is a random variable $\mathbf{B}_n^\psi \sim Q_n^\psi$, where we define Q_n^ψ as:

$$Q_n^\psi = \{P(\mathbf{B}_n^\psi = \mathbf{b}_{ij} | \mathbf{S}_n, \mathbf{\Gamma}_n, \theta^\psi)\}_{i \neq j},$$
$$P(\mathbf{B}_n^\psi = \mathbf{b}_{ij} | \mathbf{S}_n, \mathbf{\Gamma}_n, \theta^\psi) = \frac{\exp(\theta^\psi (s_{ni} - s_{nj})/(\gamma_{ni} + \gamma_{nj}))}{\sum_{k \neq l} \exp(\theta^\psi (s_{ni} - s_{nj})/(\gamma_{ni} + \gamma_{nj}))}.$$

Note that if $\theta^\psi = 0$, Q_n^ψ becomes a uniform distribution indicating that the preferences of the expert ψ deviate completely from the consensus (is an outlier), and will not be modeled by it. Values between 0 and 1 indicate different degrees of agreement, with $\theta^\psi = 1$ indicating complete agreement. Hence, by introducing θ^ψ we make the model robust, allowing it to control the extent to which each expert's preferences are modeled by the scores, effectively eliminating the outliers.

In the generative process we now assume that at each of the $T_n = \sum_\psi T_n^\psi$ draws an expert ψ is picked at random and a preference is generated from Q_n^ψ; Figure 2(b) demonstrates this process. Under this process the probability of the observed instance n with count matrices

$\mathbf{Y}_n = \{\mathbf{Y}_n^1, ..., \mathbf{Y}_n^\Psi\}$ is given by:

$$P(\mathbf{Y}_n|\mathbf{S}_n, \mathbf{\Gamma}_n, \mathbf{\Theta}) = \\ = \prod_{\psi=1}^{\Psi}\left[\frac{T_n^\psi!}{\prod_{i\neq j}\mathbf{Y}_n^\psi(i,j)!}\prod_{i\neq j}P(\mathbf{B}_n^\psi = \mathbf{b}_{ij}|\mathbf{S}_n, \mathbf{\Gamma}, \theta^\psi)^{\mathbf{Y}_n^\psi(i,j)}\right] \\ = \prod_{\psi=1}^{\Psi}\left[\frac{T_n^\psi!}{\prod_{i\neq j}\mathbf{Y}_n^\psi(i,j)!}\prod_{i\neq j}\left(\frac{\exp(\theta^\psi(s_{ni}-s_{nj})/(\gamma_{ni}+\gamma_{nj}))}{\sum_{k\neq l}\exp(\theta^\psi(s_{nk}-s_{nl})/(\gamma_{nk}+\gamma_{nl}))}\right)^{\mathbf{Y}_n^\psi(i,j)}\right].$$

The preferences are modeled by a mixture of Ψ multinomials that share the same score vector $\mathbf{S}_n$ but differ in the adherence parameter θ^ψ. Both $\mathbf{S}_n$ and $\mathbf{\Theta}$ can be efficiently learned by maximizing the log likelihood, and the consensus ranking can then be obtained by sorting the scores.

As noted above, in many preference aggregation problems the input typically consists of several preference instances n, and the goal is to infer a separate set of scores $\mathbf{S}_n$ and variances $\mathbf{\Gamma}_n$ for each instance n. The log likelihood of the entire corpus under the model is given by:

$$\mathcal{L}(\{\mathbf{Y}_n\}|\{\mathbf{S}_n\}, \{\mathbf{\Gamma}_n\}, \mathbf{\Theta}) = \\ = \log\prod_{n=1}^{N}\prod_{\psi=1}^{\Psi}\left[\frac{T_n^\psi!}{\prod_{i\neq j}\mathbf{Y}_n^\psi(i,j)!}\prod_{i\neq j}P(\mathbf{B}_n^\psi = \mathbf{b}_{ij}|\mathbf{S}_n, \mathbf{\Gamma}, \theta^\psi)^{\mathbf{Y}_n^\psi(i,j)}\right].$$

Here $\mathbf{\Theta}$ is shared across the instances and the original MPM model is recovered by setting $\mathbf{\Theta} \equiv \mathbf{1}$. When two of the three parameters $\{\mathbf{S}_n, \mathbf{\Gamma}_n, \mathbf{\Theta}\}$ are fixed it is not difficult to show that $\mathcal{L}$ is concave with respect to the third parameter. Therefore simple gradient descent can be used to efficiently find a globally optimal setting. Furthermore, even though joint optimization is no longer convex, in the experiments we found that by using gradient descent jointly good local optimum solutions can still be found efficiently.

When training examples are available the inference proceeds as follows: first training examples are used to set $\mathbf{\Theta}$; then keeping $\mathbf{\Theta}$ fixed the scores and the variances are optimized on the test examples by maximizing the log likelihood. The advantage of this approach is that it is conceptually simple and can be applied to most extensions/generalizations of the model. The disadvantage is that it requires computing parameter gradients for every test instance which can be computationally intensive. Moreover, due to the non-convexity we do not have any guarantees on the types of solutions found by this approach since gradient optimization can converge to any local optimum. These disadvantages however are shared by most score-based aggregation models. For many of these models (aside from the simple ones) finding the maximum a posteriori score vectors is intractable so one has to resort to approximate variational or gradient-based methods that have similar complexities. We empirically found gradient-based inference to be stable provided that the model parameters are initialized with small values. Throughout all experiments we used samples from a Gaussian with mean 0 and standard deviation of 0.01 to initialize the parameters and found that the difference in results across multiple restarts was negligible.

2.6 Rank Aggregation Experiments

For rank aggregation problem we use the LETOR (Liu et al., 2007a) benchmark data sets. These data sets were chosen because they are publicly available, include several baseline results, and provide evaluation tools to ensure accurate comparison between methods. In LETOR4.0 there are two rank aggregation data sets, MQ2007-agg and MQ2008-agg.

MQ2007-agg contains 1692 queries with 69,623 documents and MQ2008-agg contains 784 queries and a total of 15,211 documents. Each query contains several lists of partial rankings of the documents under that query. There are 21 such lists in MQ2007-agg and 25 in MQ2008-agg. These are the outputs of the search engines to which the query was submitted. In addition, in both data sets, each document is assigned one of three relevance levels: 2 = highly relevant, 1 = relevant and 0 = irrelevant. Finally, each data set comes with five precomputed folds with 60/20/20 splits for training/validation/testing. The results shown for each model are the averages of the test set results for the five folds.

The MQ2007-agg data set is approximately 35% sparse, meaning that for an average query the partial ranking matrix of documents by search engines will be missing 35% of its entries. MQ2008-agg is significantly more sparse with the sparsity factor of approximately 65%.

The goal is to use the rank lists to infer an aggregate ranking of the documents for each query which maximally agrees with the held-out relevance levels. To evaluate this agreement we use standard information retrieval metrics: Normalized Discounted Cumulative Gain (N@K) (Jarvelin and Kekalainen, 2000), Precision (P@K) and Mean Average Precision (MAP) (Baeza-Yates and Ribeiro-Neto, 1999). Given an aggregate ranking π, and relevance levels $\mathbf{L}_n$, NDCG is defined as:

$$NDCG(\pi, \mathbf{L}_n)@K = \frac{1}{G(\mathbf{L}_n, K)} \sum_{i=1}^{K} \frac{2^{\mathbf{L}_n(\pi^{-1}(i))} - 1}{\log(1+i)},$$

where $\mathbf{L}_n(\pi^{-1}(i))$ is the relevance level of the document in position i in π, and $G(\mathbf{L}_n, K)$ is a normalizing constant that ensures that a perfect ordering has an NDCG value of 1. The normalizing constant allows an NDCG measure averaged over multiple instances with different numbers of items to be meaningful. Furthermore, K is a truncation constant and is generally set to a small value to emphasize the utmost importance of getting the top ranked items correct.

MAP only allows binary (relevant/not relevant) document assignments, and is defined in terms of average precision (AP):

$$AP(\pi, \mathbf{L}_n) = \frac{\sum_{i=1}^{M_n} P@i * \mathbf{L}_n(\pi^{-1}(i))}{\sum_{i=1}^{M_n} \mathbf{L}_n(\pi^{-1}(i))},$$

where $P@i$ is the precision at i:

$$P@i = \sum_{j=1}^{i} \frac{\mathbf{L}_n(\pi^{-1}(j))}{i}.$$

MAP is then computed by averaging AP over all queries. To compute P@k and MAP on the MQ data sets the relevance levels are binarised with 1 converted to 0 and 2 converted

	NDCG					Precision					
	N@1	N@2	N@3	N@4	N@5	P@1	P@2	P@3	P@4	P@5	MAP
MQ2008											
BordaCount	23.68	28.06	30.80	34.32	37.13	29.72	30.42	29.38	29.75	29.03	39.45
CPS-best	26.52	31.38	34.59	37.63	40.04	31.63	32.27	32.27	31.66	30.64	41.02
SVP	32.49	36.20	38.62	40.17	41.85	38.52	36.42	34.65	32.01	30.23	43.61
Condorcet	35.67	37.39	39.11	40.50	41.59	40.94	37.43	34.73	32.08	29.59	42.63
Bradley-Terry	**38.05**	39.24	40.77	41.79	42.62	**44.77**	39.73	36.26	33.19	30.28	**44.35**
Plackett-Luce	35.20	38.49	39.70	40.49	41.55	41.32	38.96	35.33	32.02	29.62	42.20
θ-MPM	37.07	**40.29**	**41.78**	**42.76**	**43.69**	43.62	**40.94**	**37.24**	**33.64**	**30.81**	44.32
MQ2007											
BordaCount	19.02	20.14	20.81	21.28	21.88	24.88	25.24	25.69	25.80	25.97	32.52
CPS-best	31.96	33.18	33.86	34.09	34.76	38.65	38.65	38.14	37.19	37.02	40.69
SVP	35.82	35.91	36.53	37.16	37.50	41.61	40.28	39.50	38.88	38.10	42.73
Condorcet	37.31	37.63	38.03	38.37	38.66	43.26	42.14	40.94	39.85	38.75	42.56
Bradley-Terry	39.88	39.86	40.40	40.60	40.91	46.34	44.65	43.48	41.98	40.95	43.98
Plackett-Luce	40.63	40.39	40.26	40.71	40.96	46.93	45.10	43.09	42.32	41.09	43.64
θ-MPM	**41.13**	**41.21**	**41.09**	**41.41**	**41.53**	**47.35**	**45.78**	**44.17**	**43.01**	**41.97**	**44.35**

Table 1: MQ2008-agg and MQ2007-agg results; statistically significant results are underlined.

to 1. All presented NDCG, Precision and MAP results are averaged across the test queries and were obtained using the evaluation script available on the LETOR website.[3].

To investigate the properties of MPM we conducted extensive experiments with various versions of the model. Through these experiments we found that the θ version (see Section 2.5) had the best performance; below we refer to this model as θ-MPM. To learn this model we first used the training data to learn the adherence parameters $\mathbf{\Theta}$. Then keeping $\mathbf{\Theta}$ fixed we inferred the scores and variances on each test query via maximum likelihood and sorted the scores to produce a predicted ranking. This is similar to the framework used by the CPS model (Quin et al., 2010) where the training data is used to estimate the θ parameter. In all experiments we did not take the variances into account during the sort.

We compare the results of θ-MPM against the best methods currently listed on the LETOR4.0 website, namely the BordaCount model and the best of the three CPS models (combination of Mallows and Plackett-Luce models) on each of the MQ data sets. We also compare with the Condorcet, Bradley-Terry and Plackett-Luce models, as well as the singular value decomposition based method SVP (Gleich and Lim, 2011). These models cover most of the primary leading approaches in unsupervised preference aggregation research. The Bradley-Terry model is fit using the same count matrices $\mathbf{Y}_n^{\psi}$ that are used for MPM.

For all models we found that 100 steps of gradient descent were enough to obtain the optimal results. To avoid constrained optimization we reparametrized the variance parameters as $\gamma_{ni} = \exp(\beta_{ni})$ and optimized β_{ni} instead. This reparametrization was done for all the reported experiments.

3. LETOR data set can be found at `http://research.microsoft.com/en-us/um/beijing/projects/letor/`.

	N@1	N@2	N@3	N@4	N@5	N@6	N@7	N@8	N@9	N@10
Ratings imputed by PMF										
Bradley-Terry	40.09	36.00	35.20	34.96	34.49	34.40	31.63	32.08	32.46	32.35
Plackett-Luce	**69.56**	54.17	48.97	46.58	44.89	43.44	42.50	41.25	40.64	40.03
Neighbor-based	61.48	49.96	44.66	42.87	40.98	39.74	39.01	37.94	37.94	37.73
MPM	69.15	**54.29**	**49.72**	**46.98**	**45.52**	**44.13**	**43.25**	**42.62**	**42.04**	**41.57**
Ratings imputed by neighbor model										
Bradley-Terry	34.66	34.07	34.09	34.11	34.02	34.22	32.73	33.14	33.47	33.48
Plackett-Luce	70.81	56.33	50.97	48.27	46.64	45.17	44.17	43.01	42.23	41.74
PMF	69.17	55.93	50.90	48.22	46.65	**45.42**	**44.58**	**43.88**	**43.22**	**42.72**
MPM	**71.90**	**56.34**	**51.21**	**48.55**	**46.73**	45.41	44.34	43.58	42.94	42.43

Table 2: NDCG results for the MovieLens data set, for each user the missing ratings are filled using the probabilistic matrix factorization model (top half), and the neighbor-based approach (bottom half); statistically significant results are underlined.

The results[4] for MPM together with the baselines on MQ2008-agg and MQ2007-agg data sets are shown in the top and bottom halves of Table 1 respectively. For each data set we conducted a paired T-test between θ-MPM and the best baseline at each of the 5 truncations for NDCG and precision as well as MAP; the statistically significant results at the 0.05 level are underlined.

From the table we see that the θ-MPM models significantly outperforms the baselines on the MQ2007-agg data set on both NDCG and MAP metrics. θ-MPM is also the best model On MQ2008-agg, significantly improving over the baselines on truncations 2-5 for NDCG and 2,3 for Precision.

2.7 Collaborative Filtering Experiments

For collaborative filtering experiments we used the MovieLens data set,[5] a collection of 100,000 ratings (1-5) from 943 users on 1682 movies. This data set was chosen because it provides demographic information such as age and occupation for each user, as well as movie information such as genre, title and release year. Each user in this data set rated at least 20 movies, but the majority of ratings for each movie are missing and the rating matrix is more than 94% sparse. We formulate the preference aggregation as follows: given users' ratings the goal is to come up with a single ranking of the movies that accurately summarizes the majority of user preferences expressed in the data. This ranking could be used as an initial recommendation for a new user who has not provided any ratings yet, as well as in a summary page. Note that the aggregation can be further personalized by only aggregating over users that share similar demographic and/or other factors with the target user.

4. All NDCG and precision values in this and other tables were multiplied by 100 to make them more readable.
5. MovieLens data set can be found at `http://www.grouplens.org/node/73`.

To convert ratings into preferences we can either sort them (resolving ties), to obtain a partial ranking for each user, or use the pairwise method to obtain the count matrices $\mathbf{Y}^{\psi}$, where $\mathbf{Y}^{\psi}(i,j) = (\ell_i^{\psi} - \ell_j^{\psi})I[\ell_i^{\psi} > \ell_j^{\psi}]$ if movies x_i and x_j were rated by user ψ and 0 otherwise. We use the sort method for the permutation-based Plackett-Luce model and use the rating difference method for the pair-based Bradley-Terry and MPM models.

In collaborative filtering and in most other applications the primary goal of aggregation is to recommend items to a new or existing user. Items ranked in the top few positions are of particular interest because they are the ones that will typically be shown to the user. Intuitively a top ranked item should have ratings from many users (*high support*) and most who rated it should prefer it to other items (*strong preference*). Consequently NDCG is a good metric to evaluate the rankers for this problem because of its emphasis on the top ranked items and the truncation level structure. Unlike rank aggregation the ground truth aggregate ratings are not available for most collaborative filtering data. To get around this problem we complete the rating matrix by imputing the missing ratings for every user. We investigate two methods of imputing the ratings: a user independent method, where all the missing ratings are filled in by the same value, and two user dependent methods. The first method is neighbor-based and predicts the ratings using the nearest neighbors for a given user ψ. The second method uses the probabilistic matrix factorization model (PMF) (Salakhutdinov and Mnih, 2008) to factorize the rating matrix and predict missing entries. Both are commonly used for CF and have shown good empirical performance on various CF tasks such as the Netflix challenge. After completing the rating matrix we compute the NDCG value for every user by sorting the items according to scores:

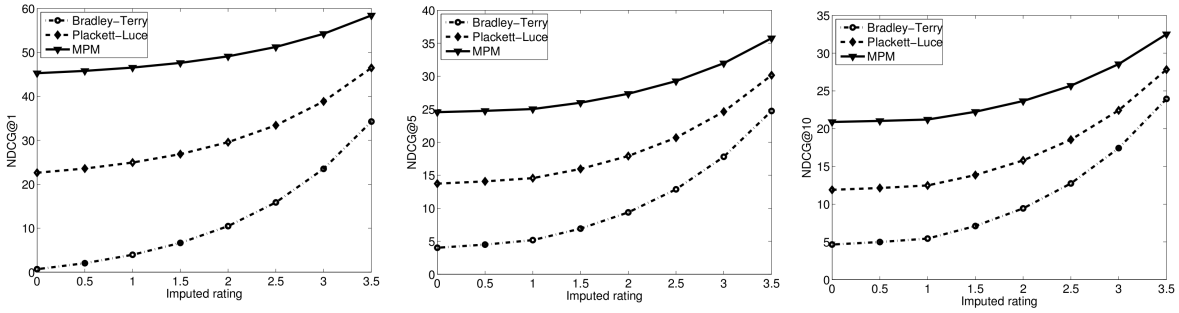
$$NDCG(\pi, \mathbf{L}^{\psi})@K = \frac{1}{G(\mathbf{L}^{\psi}, K)} \sum_{i=1}^{K} \frac{2^{\mathbf{L}^{\psi}(\pi^{-1}(i))} - 1}{\log(i+1)}. \tag{2}$$

Here π is the aggregated ranking obtained by sorting the items according to scores, and $\pi^{-1}(i)$ is the index of the item in position i in π; $\mathbf{L}^{\psi}$ is a (completed) vector of ratings for user ψ. $G(\mathbf{L}^{\psi}, K)$ is the normalizing constant and represents the maximum DCG value that could be obtained for ψ:

$$G(\mathbf{L}^{\psi}, K) = \sum_{i=1}^{K} \frac{2^{\mathbf{L}^{\psi}(\sigma^{-i}(i))} - 1}{\log(i+1)},$$

where σ is a permutation of $\mathbf{L}^{\psi}$ with the ratings sorted from largest to smallest. In this form, if an item in position i in π has a rating lower than the rating $\mathbf{L}^{\psi}(\sigma^{-1}(i))$ of the i'th highest rated item by user ψ, the corresponding term in the NDCG summation will decrease exponentially with the difference between $\mathbf{L}^{\psi}(\sigma^{-1}(i))$ and $\mathbf{L}^{\psi}(\pi^{-1}(i))$. We use this metric (averaged across all users) to evaluate the performance of the models.

We compare the results of MPM to the Bradley-Terry and Plackett-Luce models, the two best baselines on the rank aggregation task. For all models we found that 100 steps of gradient descent was enough to reach convergence. In addition to rank aggregation, we also examine CF methods directly to aggregate the items. To avoid biased evaluation in both cases we use a different CF method to aggregate the items, that is, if ratings are imputed with PMF (neighbor-based) then the neighbor-based (PMF) model is used to aggregate the items. We use Borda count to aggregate the items by first sorting the completed rating



(a) (b) (c)

Figure 3: Plots of NDCG at truncations 1, 5 and 10; in this setting all the missing ratings were repeatedly imputed by one of the constants shown on the x-axis and the rankings given by each method were evaluated using NDCG (Equation 2). All the differences are statistically significant.

vector for each user to get a user-dependent item ranking. The resulting rankings are then aggregated across all users using the Borda rule. We chose to use Borda count here because it is a well-explored approach that has stable performance and is commonly applied to social choice problems such as CF.

The NDCG results from the user dependent rating imputation method are shown in Table 2. From this table we see that MPM outperforms the best aggregation method, Plackett-Luce, both when ratings are imputed by the neighbor-based approach (top half of the table) and PMF (bottom half of the table). We also see that the neighbor-based CF model performs considerably worse than both MPM and Plackett-Luce while PMF has competitive performance, outperforming MPM at higher truncations. These results are consistent with the CF literature where PMF is typically found to perform better since it can capture more complex correlations that extend beyond simple neighbor relationships. However, unlike most aggregation methods that only learn one parameter (two for MPM) per item, in PMF we have to fit a set of parameters for each user and item. This significant increase in the number of parameters makes optimization more complex as PMF models are typically highly prone to overfitting. Moreover, it is unclear how learned models should be used to get the aggregate ranking as they only provide user-dependent rating predictions. This introduces an additional optimization step that needs to be run before ranking predictions can be made. Overall, for the MovieLens data we found that these models did not give significant gains while being considerably slower.

The NDCG plots for the user independent rating imputation method are shown in Figure 3. Here we concentrate on comparison with other aggregation methods and exclude CF methods for reasons mentioned above. The plots show NDCG at truncations 1, 5 and 10 for the three methods, when each of the values in $\{0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5\}$ was used to fill the missing ratings. Here, the value 3.5 was chosen as the upper boundary because it is the average rating for the MovieLens data set. A number of studies have shown that users tend to rate items that they like so the average of the observed ratings is typically significantly

Bradley-Terry	#u	#won	#lost	Plackett-Luce	#u	#won	#lost	MPM	#u	#won	#lost
Pather Panchali	8	1431	89	Shawshank Red.	283	32592	5943	Star Wars	583	49290	10112
Wallace & Gromit	67	7448	962	Wallace & Gromit	67	7448	962	Raiders of the L.	420	40057	10644
Casablanca	243	26837	4633	Usual Suspects	267	30779	6666	Godfather	413	36531	8040
Close Shave	112	11219	1963	Star Wars	583	49290	10112	Silence of the L.	390	38192	9125
Rear Window	209	22590	4513	Wrong Trousers	118	13531	2291	Shawshank Red.	283	32592	5943
.				.				.			
.				.				.			
.				.				.			
Children of Corn	19	62	3161	Barb Wire	30	462	5507	Cable Guy	106	3469	14377
Lawnmower Man 2	21	129	3868	Robocop 3	11	125	2535	Striptease	67	1347	9909
Free Willy 3	27	171	3912	Gone Fishin'	11	123	1098	Very Brady	93	3353	12509
Kazaam	10	128	2041	Highlander III	16	881	2826	Jungle2Jungle	132	2375	11086
Best of the Best 3	6	33	1445	Ready to Wear	18	289	3785	Island of Dr.	57	1176	9415

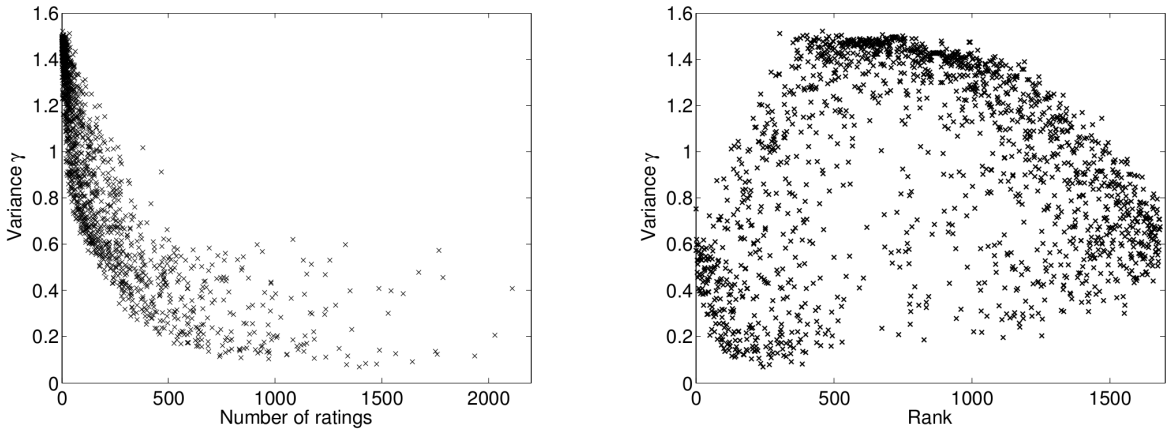
Table 3: Top 5 and bottom 5 movies found by each model. For each movie the table shows the number of users that rated it (#u) and the total number of pairwise contests that the movie won (#won) and lost (#lost) across all users.

higher than the average of the unobserved ones (Marlin et al., 2005). From the figure we see that MPM significantly outperforms both the Bradley-Terry and Plackett-Luce models. The differences are especially large when low values are imputed for the missing ratings. This indicates that the Bradley-Terry and Plackett-Luce models place items that were rated by very few users (*low support*) at the top of the list. This causes the imputed ratings to dominate the numerator in the NDCG summation making the results very sensitive to the magnitude of the imputed rating.

This effect can also be observed from Table 3, which shows the top and bottom 5 movies generated by each model together with statistics on the number of users that rated each movie and the number of pairwise contests lost and won by the movie (summed across all users). For a given user ψ and movie i with rating ℓ_i^ψ we find the number of pairwise wins by counting the number of pairs (i, j) with $\ell_i^\psi > \ell_j^\psi$; losses are found in a similar way. From the table we see that the Bradley-Terry model places the movie *Pather Panchali* at the top of the list. This movie is only rated by 8 out of 943 users and even though most users who rated it preferred it to other movies (#lost is low) there is still very little evidence that this movie represents the top preference for the majority of users. Due to its pairwise independence assumption the Bradley-Terry model is always likely to place movies with few ratings near the top/bottom of the list.

The Plackett-Luce model partially fixes this problem by considering items jointly, and places the frequently rated movie *Shawshank Redemption* first. However the model does not fully eliminate the problem, placing the very infrequently rated *Wallace & Gromit* (also ranked second by Bradley-Terry) in the second spot. Part of the reason for this comes from the fact that Plackett-Luce is a permutation based model and as such cannot model the strength of preferences, treating the preferences given for example by ratings $\{5, 2, 1\}$ the same as $\{5, 4, 3\}$.

On the other hand for the Multinomial Preference Model we see that the position of the item is related to both the number of observed preferences and the strength of those



(a) (b)

Figure 4: 4(a) shows the number of ratings versus the learned variance γ_i for each movie x_i. 4(b) shows the rank for each movie obtained after sorting the scores versus the learned γ_i.

preference. The top three movies are all rated by more than 400 users and are strongly preferred by the majority of those users.

A more severe pattern can be observed for the bottom 5 movies. Both Bradley-Terry and Plackett-Luce place movies rated by fewer than 30 users in the bottom 5 positions, labeling them the worst movies in the entire data set. This selection has very little evidence in the data and has a high probability of being wrong if more ratings are collected. For MPM all of the bottom 5 movies are rated by more than 50 users with 3 out of 5 movies rated by more than 90 users.

In addition to the retrieval accuracy we investigated the properties of the learned variance parameters γ. Figure 4(a) shows the learned variances together with the number of ratings for each movie. Note that the variance is inversely proportional to the number of ratings, so as the number of ratings increases the model becomes increasingly more certain in the preferences, decreasing the variance. In Figure 4(b) we plot γ against the aggregate rank for each movie. The general pattern is clear: the variance decreases towards the extremes of the ranking, indicating that the model is more certain in the movies that are placed near the top and near the bottom of the aggregate ranking. As shown above, this is due to the fact that the movies at the extremes of the ranking have many comparisons, allowing accurate inference of strong negative or positive preferences.

The plot however also shows outliers, which are the movies placed in the middle of the aggregate ranking with low variance/high confidence. After further inspection we found that each such movie had many positive as well as negative preferences. Examples of these include *Sabrina* (#u:190 #won:10190 #lost:12347), *Mrs. Doubtfire* (#u:192 #won:13251 #lost:17551) and *Ghost* (#u:170 #won:11785 #lost:14452). Note that all three movies were rated by more than 150 users and overall were neither strongly preferred nor strongly

disliked. The model thus appropriately placed them in the middle of the ranking with strong confidence. Moreover, note that it is impossible to express this confidence with scores alone since all the movies in the middle of the ranking have similar scores. The variances thus provide additional information about the decisions made by the model during the aggregation, which could be very useful for post-processing and evaluation.

3. Supervised Preference Aggregation

Research in preference aggregation has largely concentrated on the unsupervised aggregation problem described above. However, many of the recent aggregation problems are amenable to supervised learning, as ground truth preference information is available. The meta-search and crowdsourcing problems are both examples of supervised preference aggregation. Due to the popularity of these and other supervised problems the supervised aggregation framework has received a lot of attention recently, with a number of competitions conducted in TREC[6] as well as other conferences, and several meta-search data sets(Liu et al., 2007a) have been released to encourage research in the area. Despite these efforts, to the best of our knowledge most of the proposed models are still unsupervised and the supervised methods are unable to fully use the labeled data and optimize the aggregating function for the target metric. There is thus an evident need to develop an effective supervised aggregation framework.

To address this problem we first develop a supervised extension of the MPM model introduced in the previous section. We show how the labeled training data can be used to set the adherence parameters $\mathbf{\Theta}$ and experimentally verify that this approach improves the test accuracy of the model.

We then develop a general framework for supervised preference aggregation. Our framework builds on the idea of converting the observed preferences into pairwise matrices described in the previous section. We first show how these matrices can be used to derive effective fixed length item representations that make it possible to apply any learning-to-rank method to optimize the parameters of the aggregating function for the target IR metric. We then show how the pairwise matrices can also be employed as potentials in a ranking Conditional Random Field (CRF), and develop efficient learning and inference procedure to optimize the CRF for the target IR metric.

We validate all of the introduced models on two supervised rank aggregation data sets from Microsoft's LETOR4.0 data collection.

3.1 Framework

As in unsupervised preference aggregation, a typical supervised problem also consists of training instances where for each instance we are given a set of items. The experts generate preferences for the items and the preferences can be in the variety of forms ranging from full/partial rankings and ratings to relative item comparisons and combinations of these. However, unlike the unsupervised problem, we now also have access to the ground truth preference information over the items for each instance. The goal is to learn an aggregating function which maps expert preferences to an aggregate ranking that maximally agrees with the ground truth preferences.

6. TREC 2013 crowdsourcing track can be found at `https://sites.google.com/site/treccrowd/`.

	EXPERTS		
d_1	2	7	0
d_2	0	0	1
d_3	10	5	0
d_4	0	15	3

(a) $\mathbf{R}$

	d_1	d_2	d_3	d_4
d_1	0	0	0	0.28
d_2	0	0	0	0
d_3	0.12	0	0	0.41
d_4	0	0	0	0

(b) $\mathbf{Y}^2$

Figure 5: (a) An example rank matrix $\mathbf{R}$ where 4 documents are ranked by 3 experts. Note that in meta-search the rank for a given document can be greater than the number of documents in $\mathbf{R}$. (b) The resulting pairwise matrix $\mathbf{Y}^2$ for expert 2 (second column of $\mathbf{R}$) after the ranks are transformed to pairwise preferences using the log rank difference method.

In this work we concentrate on the rank aggregation instance of this problem from the information retrieval domain. However, the framework that we develop is general and can be applied to any supervised preference aggregation problem in the form defined above. In information retrieval the instances correspond to queries $\mathbf{Q} = \{q_1, ..., q_N\}$ and items to documents $\mathbf{D}_n = \{d_{n1}, ..., d_{nM_n}\}$ where M_n is the number of documents retrieved for q_n. For each query q_n the experts' preferences are summarized in an $M_n \times \Psi$ rank matrix $\mathbf{R}_n$ where $\mathbf{R}_n(i, \psi)$ denotes the rank assigned to document d_{ni} by the expert ψ. Note that the same Ψ experts rank items for all the queries so Ψ is query independent. Furthermore, $\mathbf{R}_n$ can be sparse, as experts might not assign ranks to every document in $\mathbf{D}_n$; we use $\mathbf{R}_n(i, \psi) = 0$ to indicate that document d_{ni} was not ranked by expert ψ. The sparsity arises in problems like meta-search where q_n is sent to different search engines and each search engine typically retrieves and ranks only a portion of the documents in $\mathbf{D}_n$. The ground truth preferences are expressed by the relevance levels $\mathbf{L}_n = \{\ell_{n1}, ..., \ell_{nM_n}\}$ which are typically assigned to the documents by human annotators.

In contrast to the learning-to-rank problem where each document is represented by a fixed length, query dependent, and typically heavily engineered feature vector, in rank aggregation the rank matrix $\mathbf{R}$ is the only information available to train the aggregating function. Additionally this matrix is typically very sparse. Hence there is no fixed length document description, as is required by most supervised methods. To overcome this problem we use the ideas behind the MPM approach and convert the rank matrix into a pairwise preference matrix. We then show how this conversion can be used to develop an effective supervised framework for this problem.

3.2 Pairwise Preferences

Given the $M_n \times \Psi$ ranking matrix $\mathbf{R}_n$ our aim is to convert it into Ψ $M_n \times M_n$ pairwise matrices $\mathbf{Y}_n = \{\mathbf{Y}_n^1, ..., \mathbf{Y}_n^\Psi\}$, where each $\mathbf{Y}_n^\psi$ expresses the preferences between pairs of documents based on the expert ψ. In Section 2.1 we considered binary and count-based

transformations; here we extend these ideas and consider a general transformation of the form $\mathbf{Y}_n^{\psi}(i,j) = g(\mathbf{R}_n(i,\psi), \mathbf{R}_n(j,\psi))$. We experiment with three versions for g that were proposed by Gleich and Lim (2011): Binary Comparison (Equation 1), and normalized and logarithmic versions of Rank Difference (Equation 2).

3. **Normalized Rank Difference**

   Here the normalized rank difference is used:

$$\mathbf{Y}_n^{\psi}(i,j) = I[\mathbf{R}_n(i,\psi) < \mathbf{R}_n(j,\psi)] \frac{\mathbf{R}_n(j,\psi) - \mathbf{R}_n(i,\psi)}{\max(\mathbf{R}_n(:,\psi))}.$$

   Normalizing by the maximum rank assigned by the expert ψ ($\max(\mathbf{R}_n(:,\psi))$) ensures that the entries of $\mathbf{Y}_n^{\psi}$ have comparable ranges across experts.

4. **Log Rank Difference**

   This method uses the normalized log rank difference:

$$\mathbf{Y}_n^{\psi}(i,j) = I[\mathbf{R}_n(i,\psi) < \mathbf{R}_n(j,\psi)] \frac{\log(\mathbf{R}_n(j,\psi)) - \log(\mathbf{R}_n(i,\psi))}{\log(\max(\mathbf{R}_n(:,\psi)))}.$$

In all cases both $\mathbf{Y}_n^{\psi}(i,j)$ and $\mathbf{Y}_n^{\psi}(j,i)$ are set to 0 if either $\mathbf{R}_n(i,\psi) = 0$ or $\mathbf{R}_n(j,\psi) = 0$ (missing ranking). Non-zero entries $\mathbf{Y}_n^{\psi}(i,j)$ represent the strength of the pairwise preference $\{d_{ni} \succ d_{nj}\}$ expressed by expert ψ. Figure 5 shows an example ranking matrix and the resulting pairwise matrix after the ranks are transformed to pairwise preferences using the log rank difference method. Note that preferences in the form of ratings and top-K lists can easily be converted into $\mathbf{Y}_n^{\psi}$ using the same transformations. Moreover, if pairwise preferences are observed, we simply skip the transformation step and fill the entries $\mathbf{Y}_n^{\psi}(i,j)$ directly.

As mentioned above, working with pairwise comparisons has a number of advantages, and models over pairwise preferences have been extensively used in areas such as social choice (David, 1988), information retrieval (Joachims, 2002; Burges, 2010), and collaborative filtering (Lu and Boutilier, 2011; Gleich and Lim, 2011). First, pairwise comparisons are the building blocks of almost all forms of evidence about preference and subsume the most general models of evidence proposed in literature. A model over pairwise preferences can thus be readily applied to a wide spectrum of preference aggregation problems and does not impose any restrictions on the input type. Second, pairwise comparisons are a relative measure and help reduce the bias from the preference scale. In meta-search for instance, each of the search engines that receives the query can retrieve diverse lists of documents significantly varying in size. By converting the rankings into pairwise preferences we reduce the list size bias emphasizing the importance of the relative position.

3.3 Previous Work

The majority of research on preference aggregation has concentrated on the unsupervised problem and was covered in detail in Section 2.2. In this section we review the supervised approaches for preference aggregation.

In meta-search a heuristic method called Reciprocal Rank Fusion (RRF) (Cormack et al., 2009) has recently been proposed. RRF uses the rank matrix to derive the document scores:

$$s_{ni} = \sum_{\psi=1}^{\Psi} \frac{1}{\alpha + \mathbf{R}_n(i,\psi)},$$

where α is a constant which mitigates the impact of high rankings from outlier experts and is set by cross validation. Once the scores are computed they are used to sort the documents. We place this method into the supervised category because of the need to set the constant α which has a significant effect on ranking accuracy (Cormack et al., 2009). Despite its simplicity RRF has obtained excellent empirical accuracy; however it also has some disadvantages. First, RRF relies on complete rankings and it is unclear how to extend it to problems like meta-search where many rankings are typically missing. Second, this method is designed specifically for rank aggregation and cannot be applied to other types of preferences.

A supervised score-based rank aggregation approach based on a Markov Chain was also recently introduced by Liu et al. (2007b). In this model the authors use the ground truth preferences to create a pairwise constraint matrix and then learn a scoring function such that the produced aggregate rankings satisfy as many pairwise constraints as possible. The main drawbacks of this approach are that it is computationally very intensive, requiring constrained optimization (semidefinite programming), and it does not incorporate the target IR metric into the optimization. The pairwise constraint idea was also recently extended by Chen et al. (2011) to a semi-supervised setting where the ground truth preferences are available for only a subset of the documents.

Extensive work in the learning-to-rank domain has demonstrated that optimizing the ranking function for the target metric produces significant gains in test accuracy (Li, 2011). However, to the best of our knowledge none of the models developed for supervised preference aggregation take the target metric into account during the optimization of the aggregating function. The models that we develop in the following sections aim to bridge this gap.

3.4 Supervised Extension for MPM

Before delving into the new models we first develop a simple supervised extension for the Multinomial Preference Model introduced above. The MPM can be considered fully unsupervised, as the adherence parameters $\mathbf{\Theta}$, the consensus scores and the variances are inferred from the observed preferences. This produces a predicted ranking for a given set of observed preferences by sorting the inferred scores, without ever using any known consensus rankings or relevance labels in the data. In this section we describe an approach to incorporate this ground truth information into this model.

Each θ^{ψ} models the adherence of the expert ψ to the consensus. For the labeled training instances the consensus is explicitly given by the relevance levels $\mathbf{L}_n$. This allows us to evaluate the adherence of each expert to the consensus exactly by computing the match between the preferences given by the expert and those expressed by the ground truth relevances. Using this we can set θ^{ψ} to the average distance between the preferences of the

expert ψ and the ground truth labels across the instances:

$$\theta^{\psi} = \frac{1}{N}\sum_{n=1}^{N} 1 - \mathcal{D}(\mathbf{L}_n, \mathbf{Y}_n^{\psi}),$$

where $\mathcal{D}$ is a normalized distance metric between preferences, such as Kendall's τ. Note that $\theta^{\psi} \to 1(\to 0)$ indicates that the preferences of expert ψ agree with (deviate from) the ground truth preferences (target consensus) across the training examples.

To apply the model we now (1) use the labeled training instances to find $\mathbf{\Theta}$ and (2) keeping $\mathbf{\Theta}$ fixed infer scores and variances for each test instance by maximizing the likelihood of the observed preferences.

3.5 Feature-Based Approach

We now introduce the first of the two fully supervised models for preference aggregation. The main idea behind this approach is to summarize the relative preferences for each document across the experts by a fixed length feature vector (Volkovs et al., 2012). This transforms the preference aggregation problem into a learning-to-rank one, and any of the standard methods can then be applied to optimize the aggregating function for the target IR metric such as NDCG. In following section we describe an approach to extract the document features.

3.5.1 Feature-Based Approach: Feature Extraction

Given the rank matrix $\mathbf{R}_n$ and the resulting pairwise matrix $\mathbf{Y}_n^{\psi}$ for expert ψ (as shown in Figure 5), our aim is to convert $\mathbf{Y}_n^{\psi}$ into a fixed length feature vector for each of the documents in $\mathbf{D}_n$. Singular Value Decomposition (SVD) based approaches for document summarization such as Latent Semantic Indexing (Deerwester et al., 1990) are known to produce good descriptors even for sparse term-document matrices. Another advantage of SVD is that it requires virtually no tuning and can be used to automatically generate the descriptors once the pairwise matrices are computed. Because of these advantages we chose to use SVD to extract the features. For a given $M_n \times M_n$ pairwise matrix $\mathbf{Y}_n^{\psi}$ the rank-p SVD factorization has the form:

$$\mathbf{Y}_n^{\psi} \approx \mathbf{U}_n^{\psi}\mathbf{\Sigma}_n^{\psi}\mathbf{V}_n^{\psi},$$

where $\mathbf{U}_n^{\psi}$ is an $M_n \times p$ matrix, $\mathbf{\Sigma}_n^{\psi}$ is an $p \times p$ diagonal matrix of singular values and $\mathbf{V}_n^{\psi}$ is an $M_n \times p$ matrix. The full SVD factorization has $p = M_n$, however, to reduce the noise and other undesirable artifacts of the original space most applications that use SVD typically set $p \ll M_n$. Reducing the rank is also an important factor in our approach as pairwise matrices tend to be very sparse with ranks significantly smaller than M_n.

Given the SVD factorization we use the resulting matrices as features for each document. It is important to note here that *both* $\mathbf{U}_n^{\psi}$ and $\mathbf{V}_n^{\psi}$ contain useful document information since $\mathbf{Y}_n^{\psi}$ is a pairwise document by document matrix. To get the features for document d_{ni} and expert ψ we use:

$$\phi(d_{ni}, \mathbf{Y}_n^{\psi}) = [\mathbf{U}_n^{\psi}(i,:), \mathrm{diag}(\mathbf{\Sigma}_n^{\psi}), \mathbf{V}_n^{\psi}(i,:)],$$

here $\mathbf{U}_n^{\psi}(i,:)$ and $\mathbf{V}_n^{\psi}(i,:)$ are the i'th rows of $\mathbf{U}_n^{\psi}$ and $\mathbf{V}_n^{\psi}$ respectively and $\mathrm{diag}(\mathbf{\Sigma}_n^{\psi})$ is the main diagonal of $\mathbf{\Sigma}_n^{\psi}$ represented as a $1 \times p$ vector. Note that the $\mathrm{diag}(\mathbf{\Sigma}_n^{\psi})$ component is

independent of i and will be the same for all the documents in $\mathbf{D}_n$. We include the singular values to preserve as much information from the SVD factorization as possible. The features $\phi(d_{ni}, \mathbf{Y}_n^\psi)$ summarize the relative preference information for d_{ni} expressed by the expert ϕ. To get a complete view across the experts we concatenate together the features extracted for each expert:

$$\phi(d_{ni}) = [\phi(d_{ni}, \mathbf{Y}_n^1), ..., \phi(d_{ni}, \mathbf{Y}_n^\Psi)].$$

Each $\phi(d_{ni}, \mathbf{Y}_n^\psi)$ contains $3p$ features so the entire representation will have $3\Psi p$ features. Moreover, note that since the experts and p are fixed across queries this representation will have the *same* length for every document in each query. We have thus created a fixed length *feature representation* for every document d_{ni}, effectively transforming the aggregation problem into a standard learning-to-rank one. During training our aim is now to learn a scoring function $f : \mathbb{R}^{3\Psi p} \rightarrow \mathbb{R}$ which maximizes the target IR metric such as NDCG. The feature representation allows us to fully use all of the available labeled training data to optimize the aggregating function for the target metric, which is not possible to do with the existing aggregation methods.

It is worth mentioning here that the SVD factorization of pairwise matrices has been used in the context of preference aggregation (see (Gleich and Lim, 2011) for example). However, previous approaches largely concentrated on applying SVD to fill the missing entries in the joint pairwise matrix $\mathbf{Y}_n^{tot} = \sum_{\psi=1}^{\Psi} \mathbf{Y}_n^\psi$ and then use the completed matrix to infer the aggregate ranking. Our approach on the other hand uses SVD to compress each pairwise matrix $\mathbf{Y}_n^\psi$ and produce fixed length feature vector for each document.

3.5.2 Feature-Based Approach: Learning and Inference

Given the document features extracted via the SVD approach our goal is to use the labeled training queries to optimize the parameters of the scoring function for the target IR metric. The main difference between the introduced feature-based rank aggregation approach and the typical learning-to-rank setup is the possibility of missing features. When a given document d_{ni} is not ranked by the expert ψ the row $\mathbf{Y}_n^\psi(i,:)$ and column $\mathbf{Y}_n^\psi(:,i)$ will both be missing (i.e., 0). To account for this we modify the conventional linear scoring function to include a bias term for each of the Ψ experts:

$$f(\phi(d_{ni}), \mathbf{W}) = \sum_{\psi \in \Psi} \mathbf{w}^\psi \cdot \phi(d_{ni}, \mathbf{Y}_n^\psi) + I[\mathbf{R}_n(i,\psi) = 0] b^\psi, \tag{3}$$

where $\mathbf{W} = \{\mathbf{w}^\psi, b^\psi\}_\psi$ is the set of free parameters to be learned with each $\mathbf{w}^\psi$ having the same dimension as $\phi(d_{ni}, \mathbf{Y}_n^\psi)$, and $I[]$ is an indicator function. The bias term b^ψ provides a base score for d_{ni} if d_{ni} is not ranked by expert ψ. The weights $\mathbf{w}^\psi$ control how much emphasis is given to preferences from expert ψ. It is important to note here that the scoring function can easily be made non-linear by adding additional hidden layer(s), as in conventional multilayer neural nets. In the form given by Equation 3 our model has a total of $(3p+1)\Psi$ parameters to be learned. We can use any of the developed learning-to-rank approaches (see Li, 2011 for a detailed description of many popular learning-to-rank methods) to optimize $\mathbf{W}$; in this work we chose to use the LambdaRank method. We chose LambdaRank because it has shown excellent empirical performance, for example, winning the Yahoo! Learning To Rank Challenge (Chapelle et al., 2010). We briefly describe

Algorithm 1 Feature-Based Learning Algorithm

Input: $\{(q_1, \mathbf{D}_1, \mathbf{L}_1, \mathbf{R}_1), ..., (q_N, \mathbf{D}_N, \mathbf{L}_N, \mathbf{R}_N)\}$
Parameters: learning rate η
for $n = 1$ **to** N **do** {feature extraction}
  from $\mathbf{R}_n$ compute $\mathbf{Y}_n = \{\mathbf{Y}_n^1, ..., \mathbf{Y}_n^\Psi\}$
  for $i = 1$ **to** M_n **do**
    compute features $\phi(d_{ni})$ using SVD
  end for
end for
initialize weights: $\mathbf{W}$
repeat {scoring function optimization}
  for $n = 1$ **to** N **do**
    compute λ-gradients: $\nabla\mathbf{W} = \sum_i \lambda_{ni} \frac{\partial s_{ni}}{\partial \mathbf{W}}$
    update weights: $\mathbf{W} = \mathbf{W} - \eta \nabla \mathbf{W}$
  end for
until convergence
Output: $\mathbf{W}$

LambdaRank here and refer the reader to Burges et al. (2006) and Burges (2010) for a more extensive treatment.

LambdaRank learns pairwise preferences over documents with emphasis derived from the NDCG gain found by swapping the rank position of the documents in any given pair, so it is a listwise algorithm (in the sense that the cost depends on the sorted list of documents). Formally, given a pair of documents (d_{ni}, d_{nj}) with $\ell_{ni} \neq \ell_{nj}$, the target probability that d_{ni} should be ranked higher than d_{nj} is defined as:

$$P_{nij} = \begin{cases} 1 & \text{if } \ell_{ni} > \ell_{nj} \\ 0 & \text{otherwise} \end{cases} .$$

The model's probability is then obtained by passing the difference in scores between d_{ni} and d_{nj} through a logistic:

$$\begin{aligned} Q_{nij} &= \frac{1}{1 + \exp(-(f(\phi(d_{ni}), \mathbf{W}) - f(\phi(d_{nj}), \mathbf{W})))} \\ &= \frac{1}{1 + \exp(-(s_{ni} - s_{nj}))}. \end{aligned}$$

The aim of learning is to match the two probabilities for every pair of documents with different labels. To achieve this a cross entropy objective is used:

$$O_n = -\sum_{\ell_{ni} > \ell_{nj}} P_{nij} \log(Q_{nij}).$$

This objective weights each pair of documents equally thus placing equal importance on getting the relative order of document correctly both at the top and at the bottom of the ranking. However, most target evaluation metrics including NDCG are heavily concentrated

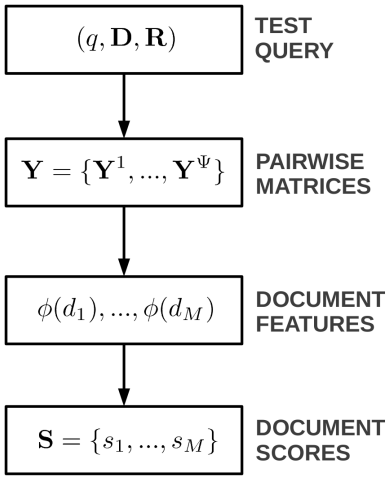


Figure 6: The inference diagram for the feature-based preference aggregation approach. Given a test query $(q, \mathbf{D}, \mathbf{R})$ the inference proceeds in three steps: (1) The ranking matrix $\mathbf{R}$ is converted to a set of pairwise matrices $\mathbf{Y} = \{\mathbf{Y}^1, ..., \mathbf{Y}^\Psi\}$. (2) SVD is used to extract document features from each pairwise matrix $\mathbf{Y}^\psi$. (3) The learned scoring function is applied to the features to produce the scores for each document. The scores are then sorted to get the aggregate ranking.

on the top of the ranking. To take this into account the LambdaRank framework uses a smooth approximation to the gradient of a target evaluation measure with respect to the score of a document d_{ni}, and we refer to this approximation as λ-gradient. The λ-gradient for NDCG is defined as the derivative of the cross entropy objective weighted by the difference in NDCG obtained when a pair of documents swap rank positions:

$$\lambda_{nij} = |\Delta NDCG@K(s_{ni}, s_{nj})| \frac{\partial O_n}{\partial(s_{ni} - s_{nj})}.$$

Thus, at the beginning of each iteration, the documents are sorted according to their current scores, and the difference in NDCG is computed for each pair of documents by keeping the ranks of all of the other documents constant and swapping only the rank positions for that pair (see Burges et al., 2006 for more details on λ calculation). The λ-gradient for document d_{ni} is computed by summing the λ's for all pairs of documents (d_{ni}, d_{nj}) for query q_n:

$$\lambda_{ni} = \sum_{j:\ell_{nj} \neq \ell_{ni}} \lambda_{nij}.$$

The $|\Delta NDCG|$ factor emphasizes those pairs that have the largest impact on NDCG. Note that the truncation in NDCG is relaxed to the entire document set to maximize the use of the available training data.

To make the learning algorithm more efficient the document features can be precomputed a priori and re-used throughout learning. This significantly reduces the computational

complexity at the cost of additional storage requirement of $O(M_n \Psi p)$ for each query q_n. The complete stochastic gradient descent learning procedure is summarized in Algorithm 1.

Once the parameters of the scoring function are learned then at test time, given a new query q with rank matrix $\mathbf{R}$, we (1) convert $\mathbf{R}$ into a set of pairwise matrices $\mathbf{Y} = \{\mathbf{Y}^1, ..., \mathbf{Y}^\Psi\}$; (2) extract the document features using rank-p SVD; and (3) apply the learned scoring function to get the score for every document. The scores are then sorted to get the aggregate ranking. This process is shown in Figure 6.

3.6 CRF Approach

The SVD-based feature approach introduced above is effective at producing compact document representation which we experimentally found to work well for this task. These representations also allow to apply any learning-to-rank approach making it very easy to incorporate the model into many existing IR frameworks. However, while the SVD model is robust, it requires applying SVD factorization at test time which can be computationally intensive. Moreover, SVD representation significantly compresses the pairwise matrices and might throw away useful information potentially affecting performance and making the model less interpretable. To avoid these disadvantages we develop another fully supervised model which uses the pairwise matrices directly. As mentioned above, the variable number of documents per query and absence of the fixed-length document representation make it difficult to apply the majority of supervised methods to this problem, since they require fixed-length item representations. CRFs on the other hand are well suited for tasks with variable input lengths, and have successfully been applied to problems that have this property, such as natural language processing (Sha and Pereira, 2003; Roth and Yih, 2005), computational biology (Sato and Sakakibara, 2005; Liu et al., 2006), and information retrieval (Qin et al., 2008; Volkovs and Zemel, 2009). Moreover, CRFs are very flexible and can be used optimize the parameters of the model for the target metric. For these reasons we develop a CRF framework for preference aggregation (Volkovs and Zemel, 2013).

3.6.1 CRF Approach: Model

The main idea behind this approach is based on an observation that the pairwise preference functions defined above naturally translate to pairwise potentials in a CRF model. Using these function we can evaluate the "compatibility" of any ranking π by comparing the order induced by the ranking with the pairwise preferences from each expert. This leads to an energy function:

$$E(\pi, \mathbf{Y}_n; \mathbf{W}) = \\ = -\frac{1}{M_n^2} \sum_{i=1}^{M_n} \frac{\sum_{\psi=1}^{\Psi} \left(b^\psi \varphi^\psi(\pi^{-1}(i)) + w_{pos}^\psi \sum_{j \neq i} \mathbf{Y}_n^\psi(\pi^{-1}(i), j) - w_{neg}^\psi \sum_{j \neq i} \mathbf{Y}_n^\psi(j, \pi^{-1}(i)) \right)}{\log(i+1)}, \quad (4)$$

where $\mathbf{Y}_n^\psi(\pi^{-1}(i), j)$ is the pairwise preference for the document in position i in π over document d_{nj} expressed by expert ψ, and $\mathbf{W} = \{b^\psi, w_{pos}^\psi, w_{neg}^\psi\}_{\psi=1}^\Psi$ is the set of free parameters to be learned.

This energy function contains a binary unary potential $\varphi^{\psi}(i, \mathbf{R}_n) = I[\mathbf{R}_n(i, \psi) = 0]$, where $I[]$ is an indicator function. This potential is active only when document d_{ni} is not ranked by the expert ψ, in which case b^{ψ} provides a base preference score for the item. The energy also contains pairwise potentials that directly use pairwise matrices $\mathbf{Y}$. Note that from the definition of $\mathbf{Y}$ in Section 3.2 it follows that only one of $\mathbf{Y}_n^{\psi}(\pi^{-1}(i), j)$ or $\mathbf{Y}_n^{\psi}(j, \pi^{-1}(i))$ can be non-zero for any pair of documents. Consequently, if $\mathbf{Y}_n^{\psi}(\pi^{-1}(i), j)$ is on (non-zero) then expert ψ "agrees" with the relative order induced by π (lowering the energy) and the strength of this agreement is given by w_{pos}^{ψ}. Similarly, if $\mathbf{Y}_n^{\psi}(j, \pi^{-1}(i))$ is on then expert ψ "disagrees" with the relative order, and raises the energy. The weights w_{pos}^{ψ} and w_{neg}^{ψ} thus control how much emphasis is given to positive and negative relative preferences from expert ψ. $1/\log(i+1)$ is the rank discount function similar to the one used in NDCG and other IR metrics, which emphasizes items at the top positions in the ranking. Finally, normalizing by $1/M_n^2$ ensures that the energy ranges are comparable across instances with different numbers of items.

The model assigns a probability to every ranking π based on this energy function:

$$P(\pi|\mathbf{Y}_n) = \frac{1}{Z(\mathbf{Y}_n)} \exp(-E(\pi, \mathbf{Y}_n; \mathbf{W})) \qquad Z(\mathbf{Y}_n) = \sum_{\pi} \exp(-E(\pi, \mathbf{Y}_n; \mathbf{W})),$$

where the partition function $Z(\mathbf{Y}_n)$ sums over all valid rankings π.

It is important to note here that in the proposed model a separate set of weights $\{b^{\psi}, w_{pos}^{\psi}, w_{neg}^{\psi}\}$ is learned for each expert ψ, which allows the model to effectively capture the correlations between individual expert preferences and the ground truth ones. However, this framework cannot be applied when the expert identity is unknown or when new experts, unseen during training, are introduced at test time. This is often the case in domains like crowdsourcing where the experts must be anonymized due to privacy considerations, and the number of experts is large so new experts are often introduced at test time. To generalize the model to these settings we can simply share the same parameters b, w_{pos} and w_{neg}, removing the dependence on ψ. The resulting *consensus* model only takes into account the net preference across all Ψ experts, ignoring the individual preferences. Though this makes it possible to apply the model to arbitrary expert sets, this may weaken it since preference information from individual experts can contain very useful information, especially in cases where the majority of experts are wrong. When a subset of the experts is known, it is possible to take an intermediate approach and learn individual weights $\{b^{\psi}, w_{pos}^{\psi}, w_{neg}^{\psi}\}$ for the known experts ψ, and consensus-based weights $\{b, w_{pos}, w_{neg}\}$ for the unknown experts. This demonstrates the flexibility of the proposed CRF framework which allows us to effectively learn to aggregate preferences in the settings where both item and expert sets can vary in length.

3.6.2 CRF Approach: Learning and Inference

The most popular approach to training CRFs is maximum likelihood. However, this approach does not take the target metric into account and thus might be ineffective at optimizing it. Recent work has explored different empirical and theoretical approaches to incorporate the target metric during CRF training (Volkovs and Zemel, 2009; Gimpel and Smith, 2010; McAllester and Keshet, 2011) Inspired by this work we follow the approach of

Algorithm 2 CRF Learning Algorithm

Input: $\{(q_1, \mathbf{D}_1, \mathbf{L}_1, \mathbf{R}_1), ..., (q_N, \mathbf{D}_N, \mathbf{L}_N, \mathbf{R}_N)\}$
Parameters: learning rate η, cut-off ϵ
for $n = 1$ **to** N **do** {pairwise matrices}
  from $\mathbf{R}_n$ compute $\mathbf{Y}_n = \{\mathbf{Y}_n^1, ..., \mathbf{Y}_n^\Psi\}$
end for
initialize weights: $\mathbf{W}$
repeat {CRF optimization}
  for $n = 1$ **to** N **do**
    if $M_n > \epsilon$ **then**
      downsample documents to get $\mathbf{L}_{n\epsilon}$, $\mathbf{Y}_{n\epsilon}$
    else
      $\mathbf{L}_{n\epsilon} = \mathbf{L}_n$ and $\mathbf{Y}_{n\epsilon} = \mathbf{Y}_n$
    end if
    compute exact gradients with $\{\mathbf{L}_{n\epsilon}, \mathbf{Y}_{n\epsilon}\}$:
      $\nabla\mathbf{W} = \partial O(\mathbf{L}_{n\epsilon}, \mathbf{Y}_{n\epsilon})/\partial\mathbf{W}$
    update weights: $\mathbf{W} = \mathbf{W} + \eta\nabla\mathbf{W}$
  end for
until convergence
Output: $\mathbf{W}$

Volkovs and Zemel (2009) and use the expected metric as the target objective to maximize:

$$O(\mathbf{L}_n, \mathbf{Y}_n) = \sum_\pi \mathcal{G}(\pi, \mathbf{L}_n) P(\pi|\mathbf{Y}_n), \tag{5}$$

where $\mathcal{G}$ can be any IR metric such as the NDCG or MAP. Note that even even the cases where $\mathcal{G}$ is non-smooth (e.g., NDCG, MAP) the above objective remains smooth with respect to $\mathbf{W}$ and can be maximized using standard gradient-based procedure. However, to optimize this objective we need to calculate $P(\pi|\mathbf{Y}_n)$ for all $M_n!$ rankings π of the items in each query q_n. This computation quickly becomes intractable since even for $M_n = 15$ one needs to sum over more than 10^{12} permutations. Standard MCMC and variational techniques can be used here to estimate the gradients, however, these methods are typically too slow to be applied to the IR domain where data sets often contain thousands of queries.

To avoid these problems we use an approach similar to the one suggested by Caetano et al. (2009) which we empirically found to work well. Every time a query q_n is visited and the number of documents is greater than ϵ, we randomly select a subset of ϵ documents and use the corresponding relevance labels $\mathbf{L}_{n\epsilon}$ and pairwise matrices $\mathbf{Y}_{n\epsilon} = \{\mathbf{Y}_{n\epsilon}^1, ..., \mathbf{Y}_{n\epsilon}^\Psi\}$ to compute the gradients for $\mathbf{W}$. Here each pairwise matrix $\mathbf{Y}_{n\epsilon}^\psi$ is a slice of the original matrix $\mathbf{Y}_n^\psi$ which includes only the selected ϵ documents. Choosing ϵ sufficiently small allows the gradients to be computed exactly by enumerating all possible $\epsilon!$ permutations of the documents.

Similarly to the feature-based model, this learning procedure can be made more efficient by precomputing both unary and pairwise potentials. This reduces the complexity of computing the model's energy from $O(M_n^2\Psi)$ to $O(M_n\Psi)$ at the cost of additional storage

requirement of $O(M_n)$ per query. The complete stochastic gradient descent learning procedure is summarized in Algorithm 2. Note that this algorithm can be used to optimize the parameters of the CRF for *any* of the target IR metrics.

Once the CRF is learned then given a new test query $(q, \mathbf{D}, \mathbf{R})$, the goal is to predict a single ranking π for the M documents in $\mathbf{D}$ based on the expert preferences $\mathbf{R}$. Fortunately, once $\mathbf{R}$ is converted into $\mathbf{Y} = \{\mathbf{Y}^1, ..., \mathbf{Y}^\Psi\}$ such inference can be done very efficiently in this CRF. Since $1/\log(i+1)$ is a monotonically decreasing function it is easy to verify that the ranking with the highest probability is obtained by sorting the items according to their total "score" given by the potentials:

$$\hat{\pi} = \arg\min_{\pi} E(\pi, \mathbf{Y}; \mathbf{W}) = \arg\operatorname{sort}([\vartheta_1, ..., \vartheta_M])$$

where the scores are:

$$\vartheta_i = -\sum_{\psi=1}^{\Psi} \left(b^\psi \varphi^\psi(i) + w^\psi_{pos} \sum_{j \neq i} \mathbf{Y}^\psi(i,j) - w^\psi_{neg} \sum_{j \neq i} \mathbf{Y}^\psi(j,i) \right).$$

It is important to note here that this inference procedure only requires computing simple sums and can thus be done very efficiently . This is a significant advantage over the feature-based approach which requires SVD factorization to be done for every test query.

3.7 Experiments

For our experiments we use the MQ2007-agg and MQ2008-agg rank aggregation data sets from the LETOR4.0 (Liu et al., 2007a) collection. Detailed description of the data sets is given is Section 2.6.

3.7.1 MPM Experiments

In the first set of experiments we compare the performances of the supervised and the unsupervised versions of the MPM model. Both versions are fit using the pairwise count matrix described in Section 2.1. Note that this method of converting rankings to pairwise preferences is very similar to the rank difference method discussed in Section 3.2 with the only difference being the exclusion of the normalization term $\max(\mathbf{R}_n(:, \psi))$. As before we use 100 steps of the gradient descent to fit the models and reparametrize the variance parameters as $\gamma_{ni} = \exp(\beta_{ni})$ to avoid constrained optimization. The results for the unsupervised (θ-MPM) and the supervised (θ_{sup}-MPM) models on both data sets are shown in Table 4. From the table we see that the supervised version of the MPM model significantly outperforms the unsupervised one on both data sets. This supports the expected conclusion that when ground truth preference data is available using it during model training improves performance.

To further investigate the differences between the two MPM models we plotted the adherence parameters $\mathbf{\Theta}$ found by each model. Figure 7 shows the adherence parameters $\mathbf{\Theta}$ set based on the labeled training examples, together with the one learned in an unsupervised fashion by doing gradient descent. From the figure we see many similarities in the two vectors, indicating that the model is able to capture the notion of "outliers" which correlates

	NDCG					Precision					
	N@1	N@2	N@3	N@4	N@5	P@1	P@2	P@3	P@4	P@5	MAP
MQ2008-agg											
θ-MPM	37.07	40.29	41.78	42.76	43.69	43.62	40.94	37.24	33.64	30.81	44.32
θ_{sup}-MPM	**38.17**	**40.57**	**42.19**	**43.07**	**43.99**	**44.89**	**41.13**	**37.67**	**33.80**	**31.17**	**44.71**
MQ2007-agg											
θ-MPM	41.13	41.21	41.09	41.41	42.53	47.35	45.78	44.17	43.01	41.97	44.35
θ_{sup}-MPM	**41.77**	**41.91**	**41.92**	**42.34**	**42.79**	**48.35**	**46.64**	**44.53**	**43.52**	**42.72**	**45.71**

Table 4: MQ2008-agg and MQ2007-agg MPM results; statistically significant results are underlined.

Figure 7: Top row: normalized $\mathbf{\Theta}$, found by the supervised procedure outlined in Section 3.4, for training Fold 1 of MQ2007-agg. Bottom row: learned $\mathbf{\Theta}$ on the same Fold. Here white = 1 and black = 0.

closely with the training labels. There are however a number of differences, such as the first three components being switched from on to off in the learned $\mathbf{\Theta}$. In our experiments we consistently found that setting $\mathbf{\Theta}$ using the training labels produced better performance.

3.7.2 SVD and CRF Experiments

In the second set of experiments we compare the performance of the SVD-based feature model trained with LambdaRank (denoted as SVDsup) and the CRF one. For each model we experimented with binary, rank difference, and log rank difference methods to compute the pairwise matrices (see Section 3.2) and selected the method that gave the best validation NDCG@10 results. For the SVD-based model we found through cross-validation that setting $p = 1$ (SVD rank) gave the best performance which is expected considering the sparsity level of the pairwise matrices. The LambdaRank training of the scoring function was run for 200 iterations with a learning rate of 0.01, and validation NDCG@10 was used to choose the best model. For the CRF model we used expected NDCG (see Equation 5) as the target objective and set $\epsilon = 6$ ensuring that at least one document of every relevance label was chosen each time.

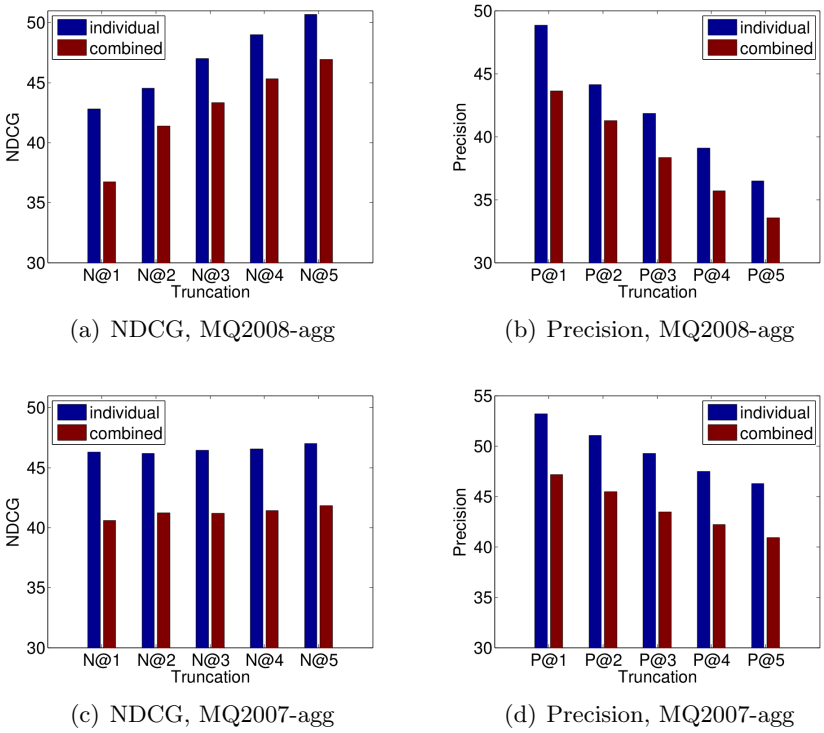
We compare the results of our model against the unsupervised baselines used in the MPM experiments (see Section 2.6). We also compare with the established meta-search standard Reciprocal Rank Fusion (RRF). To the best of our knowledge these models cover all of the primary leading approaches in the (supervised) rank aggregation research except for the Markov Chain model (Liu et al., 2007b) discussed above. We were unable to compare with this method because it is neither publicly available nor listed as one of the baselines on LETOR, making standardized comparison difficult.

	NDCG					Precision					
	N@1	N@2	N@3	N@4	N@5	P@1	P@2	P@3	P@4	P@5	MAP
MQ2008-agg											
BordaCount	23.68	28.06	30.80	34.32	37.13	29.72	30.42	29.38	29.75	29.03	39.45
CPS-best	26.52	31.38	34.59	37.63	40.04	31.63	32.27	32.27	31.66	30.64	41.02
SVP	32.49	36.20	38.62	40.17	41.85	38.52	36.42	34.65	32.01	30.23	43.61
Bradley-Terry	38.05	39.24	40.77	41.79	42.62	44.77	39.73	36.26	33.19	30.28	44.35
Plackett-Luce	35.20	38.49	39.70	40.49	41.55	41.32	38.96	35.33	32.02	29.62	42.20
Condorcet	35.67	37.39	39.11	40.50	41.59	40.94	37.43	34.73	32.08	29.59	42.63
RRF	38.77	40.73	43.48	45.70	47.17	44.89	41.32	38.82	36.51	34.13	47.71
θ_{sup}-MPM	38.17	40.57	42.19	43.07	43.99	44.89	41.13	37.67	33.80	31.17	44.71
SVDsup	**42.81**	44.53	47.02	49.00	50.69	**48.85**	44.13	41.84	**39.09**	36.50	50.32
CRF	42.29	**44.99**	**47.54**	**49.05**	**51.03**	48.67	**44.58**	**42.08**	38.75	**36.55**	**50.41**
MQ2007-agg											
BordaCount	19.02	20.14	20.81	21.28	21.88	24.88	25.24	25.69	25.80	25.97	32.52
CPS-best	31.96	33.18	33.86	34.09	34.76	38.65	38.65	38.14	37.19	37.02	40.69
SVP	35.82	35.91	36.53	37.16	37.50	41.61	40.28	39.50	38.88	38.10	42.73
Bradley-Terry	39.88	39.86	40.40	40.60	40.91	46.34	44.65	43.48	41.98	40.95	43.98
Plackett-Luce	40.63	40.39	40.26	40.71	40.96	46.93	45.10	43.09	42.32	41.09	43.64
Condorcet	37.31	37.63	38.03	38.37	38.66	43.26	42.14	40.94	39.85	38.75	42.56
RRF	41.93	42.66	42.42	42.73	43.13	48.70	47.20	44.84	43.52	42.52	46.72
θ_{sup}-MPM	41.77	41.91	41.92	42.34	42.79	48.35	46.64	44.53	43.52	42.72	45.71
SVDsup	46.13	46.76	46.71	**46.87**	**47.28**	52.90	51.39	**49.33**	**47.80**	46.66	50.05
CRF	**46.93**	**46.81**	**46.75**	46.51	46.93	**54.14**	**51.48**	49.12	47.54	**46.68**	**50.39**

Table 5: MQ2008-agg and MQ2007-agg results; statistically significant differences between CRF and SVDsup are underlined. All the differences between both CRF and SVDsup models and the best baseline are statistically significant.

The NDCG and Precision results for truncations 1-5 as well as MAP results are shown in Table 5. From the table we see that both SVDsup and CRF model significantly outperform all the other aggregation methods, improving by as much as 5 NDCG points over the best baseline on each of the MQ-agg data sets. Moreover, we also see that the CRF model has very strong performance producing similar results to the best SVDsup model. While both models use the same pairwise matrices $\mathbf{Y}$, inference in CRF is orders of magnitude faster than in SVDsup. Finally, we note that the results on the MQ2008-agg data set, which is more than 65%, sparse demonstrate that both models are robust and generalize well even in the sparse setting.

To investigate the utility of representing each expert's preferences with a separate set of parameters we ran experiments with a combined approach. For each query q_n we combined all pairwise preference matrices into a single matrix $\mathbf{Y}_n^{tot} = \sum_{\psi=1}^{\Psi} \mathbf{Y}_n^{\psi}$ and trained the SVDsup model on the SVD features from $\mathbf{Y}_n^{tot}$ only. Note that this model has no information about the individual expert preferences. We refer to this model as "combined" and compare it to the full model which uses SVD features from each $\mathbf{Y}_n^{\psi}$ referred to as "individual". The results for the two data sets are shown in Figure 8. From the figure we see that the individual model significantly outperforms the combined one on both data sets. The performance of



(a) NDCG, MQ2008-agg

(b) Precision, MQ2008-agg

(c) NDCG, MQ2007-agg

(d) Precision, MQ2007-agg

Figure 8: Test NDCG@1-5 and Precision@1-5 results for SVDsup trained on features extracted from each $\mathbf{Y}_n^{\psi}$ (individual), versus SVDsup trained on features extracted only from the joint preference matrix $\mathbf{Y}_n^{tot}$ (combined). Results for the CRF model are similar and are not shown.

the combined model is comparable to the best baseline consensus method, the Reciprocal Rank Fusion. This is not surprising since $\mathbf{Y}_n^{tot}$ summarizes the consensus preference across the experts, and without access to the individual preferences the ranker can only learn to follow the consensus. Note however, that the gain from using the individual expert features is very significant which further supports the conclusion that specialization is very important for the supervised preference aggregation. To further validate this conclusion we conducted the same experiment with the CRF model. We removed the expert specific weights w^{ψ}, w_{pos}^{ψ} and w_{neg}^{Ψ} and learned a single set of parameters $\mathbf{W} = \{b, w_{pos}, w_{neg}\}$ for all the experts. The results were similar to those in Figure 8 and we observed a significant drop in both NDCG and Precision accuracies.

Figure 9 further demonstrates the importance of specialization. The figure shows the expert ranking matrix together with the scores produced by the Reciprocal Rank Fusion and SVDsup for an example test query from MQ2008-agg. From the ground truth relevance levels ($\mathbf{L}$) it is seen that only document d_6 is relevant to this query. However, from the ranking matrix we see that the experts express strong net preference for documents d_1, d_4, d_5 and d_8, whereas the preferences for d_6 are mixed with many positive and negative preferences. The Reciprocal Rank Fusion is a consensus-based approach and as such ranks

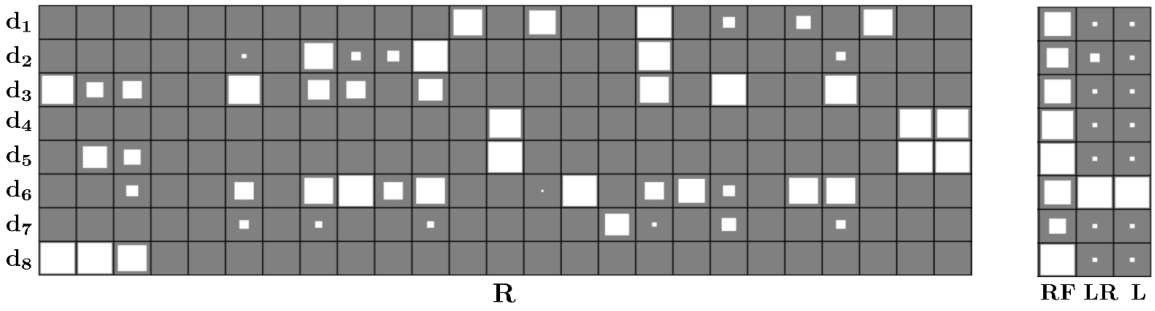


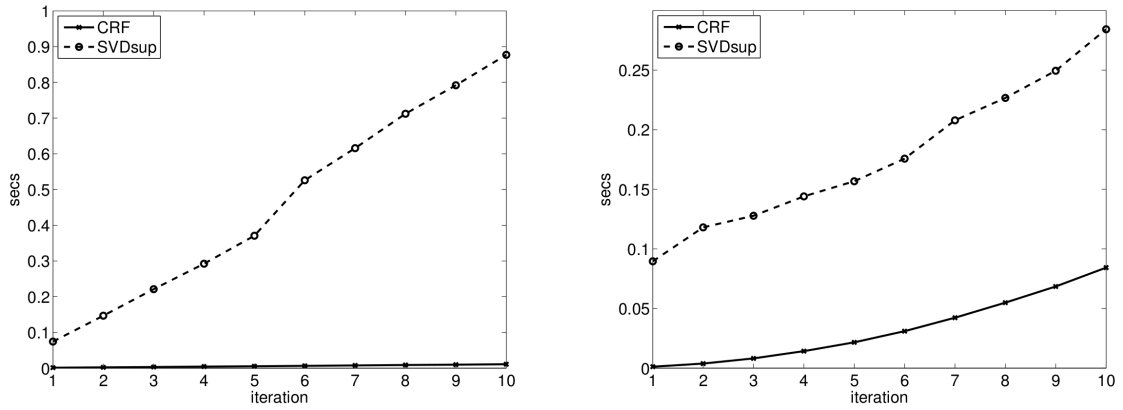
Figure 9: The matrix on the left shows the expert ranking matrix $\mathbf{R}$ for a test query in Fold 1 of the MQ2008-agg data set. 8 documents $\{d_1, ..., d_8\}$ were retrieved for this query and the normalized expert rankings are represented by white squares. The size of each square reflects the preference strength, so that large squares correspond to high rankings (strong preference). Missing rankings are represented by empty cells. The matrix on the right shows the normalized scores for each document produced by the Reciprocal Rank Fusion (RF) and the LambdaRank SVDsup (LR) models, as well as the ground truth relevance levels ($\mathbf{L}$). Note that only d_6 is relevant to this query.

documents d_1, d_4, d_5 and d_8 above d_6 with NDCG@1-4 of 0. Other consensus-based methods produce similar rankings. SVDsup on the other hand, is able to ignore the consensus and concentrate on a small subset of the preferences, placing d_6 on top and producing a perfect ranking. Moreover, note that the scores for the other documents produced by the SVDsup are significantly lower than the score for d_6, so the model is confident in this ranking. The query shown in Figure 9 is an example of a difficult query (often these correspond to long-tail queries) where the majority of experts generate incorrect preferences. For these queries the aggregated rankings produced by the consensus-based methods will also be incorrect. Our supervised approaches are able to fix this problem through specialization. By examining the queries in both MQ2008-agg and MQ2007-agg we found that both data sets contain a number of such queries and that both SVDsup and CRF performs significantly better on those queries than the consensus-based baselines.

3.7.3 Runtime Comparison

In the previous sections we demonstrated that fully supervised models SVDsup and CRF significantly outperform all baselines on two rank aggregation tasks. We also mentioned that the CRF model is considerably faster at inference time. In this section we quantify this difference.

We use test Fold 1 of the MQ2008-agg data set and conduct two sets of experiments. In the first experiment we repeatedly increase the number of experts. Starting with the initial expert matrix at iteration 1: $\mathbf{R}_n^{(1)} = \mathbf{R}_n$, we concatenate it with the original matrix to get an expanded one for iteration 2: $\mathbf{R}_n^{(2)} = [\mathbf{R}_n^{(1)}, \mathbf{R}_n]$. Thus, after t iterations the resulting matrix $\mathbf{R}_n^{(t)} = [\mathbf{R}_n^{(t-1)}, \mathbf{R}_n]$ contains M_n rows and $t \times \Psi$ columns. Concatenating



(a) Expert expansion runtimes

(b) Item expansion runtimes

Figure 10: Average per query runtimes (in seconds) for test Fold 1 of the MQ2008-agg. Figure 10(a) shows runtimes for the expert expansion experiment. Figure 10(b) shows runtimes for the item expansion experiment.

expert matrices allows us to test the inference procedure of each method on an increasingly larger data while preserving sparsity. In the second experiment we repeat this procedure but this time we append the matrices increasing the number of documents. Here, the ranking matrix at iteration t contains $t \times M_n$ rows and Ψ columns. The first experiment thus tests for scenarios where the expert set is large (expert expansion), that typically arise in domains like crowdsourcing. While the the second experiment tests for large item sets (item expansion) that often arise in domains like meta-search.

Figures 10 and 10(b) show, averaged across queries, runtimes (in seconds) for both methods at each expansion iteration. Figure 10(a) shows runtimes for the expert expansion while Figure 10(b) shows runtimes for the item expansion. From the figures we see significant differences in runtimes between the two methods. The difference is especially large for the expert expansion (Figure 10(a)) where SVDsup is on average almost 80 times slower than our CRF method at the tenth iteration. This difference is due to the fact that SVDsup has to run SVD factorization for every expert. Consequently, the number of SVD factorizations grows linearly with the number of experts significantly slowing down SVDsup. For the item expansion (Figure 10(b)) the number of experts stays constant while the dimension of the preference matrix increases. Since no additional SVD factorizations are required we found the speed of SVDsup to not increase as significantly as in the first experiment. However, even in this setting the CRF model is more than 3.5 times faster. From these results we can conclude that when inference speed is important CRF is a better model choice especially if the number of experts is large.

3.7.4 Assessing Expert Quality

An additional advantage of using preference matrices directly, as done in CRF, is model interpretability. By analyzing the learned potential weights we can gain insight into which

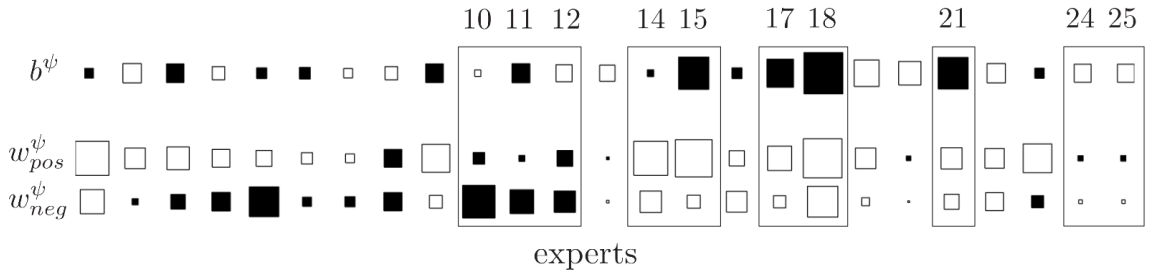


Figure 11: Learned b^ψ, w^ψ_{pos} and w^ψ_{neg} expert weights for training Fold 1 of MQ2008-agg; weights for other folds look analogous. White squares represent positive weights while black squares represent negative ones. The area of each square is proportional to weight magnitude.

experts are useful and how their preferences are combined. Figure 11 shows an example weight matrix learned by CRF model on the training Fold 1 of MQ2008-agg. Before delving into the figure we note that negative b^ψ raises the energy (lowering the probability, see Equation 4). Hence large negative values indicate that when preference from expert ψ is missing for a given document it is pushed down in the aggregate ranking, that is, expert ψ is important for aggregation. Similarly, positive w^ψ_{pos} lower the energy (upping the probability) while positive w^ψ_{neg} raise the energy. Consequently, when both weights are positive for a given expert ψ, documents d_{ni} strongly preferred by k (i.e., $\sum_{j\neq i} \mathbf{Y}^\psi_n(i,j) \gg \sum_{j\neq i} \mathbf{Y}^\psi_n(j,i)$) get pushed up in the ranking while those not preferred get pushed down.

Taking these relationships into account we see from Figure 11 that preferences from experts 14, 15, 17, 18 and 21 are good indicators of document relevance. The importance of these experts is shown by large negative values of b^ψ. Moreover, large positive values for both w^ψ_{pos} and w^ψ_{neg} indicate that strong net preference from each of these experts correlates closely with high relevance.

We also see that some experts are not useful for aggregation. For instance experts 24, and 25 all have positive b^ψ's meaning that when their preferences are absent the rank of a document actually improves. Each of these experts also has near-zero w^ψ_{pos} and w^ψ_{neg} indicating that when their preferences are present the model does not use them.

Finally, some experts are used for aggregation even though their preferences correlate inversely with ground truth. For instance, experts 10, 11 and 12 all have negative w^ψ_{pos} and w^ψ_{neg} weights indicating that documents strongly preferred by these experts will be pushed down in the ranking while those strongly opposed will be pushed up. Moreover, most weights for these experts are large indicating that they play an important role in the aggregation process. The model thus learned that these experts often give wrong relative orderings reversing which can still lead to useful predictions. It is worth noting here that this kind of inverse relationship is impossible to capture with unsupervised methods.

To further validate the utility of analyzing experts through CRF's parameters we removed experts whose preferences were found not to be useful by the CRF and retrained the model. Specifically, from Figure 11 we see that experts 13, 20, 24 and 25 are not being used by the model and when preferences from these experts are missing, the corresponding

	N@1	N@2	N@3	N@4	N@5
CRF	42.29	44.99	47.54	**49.05**	**51.03**
CRF*	**42.64**	**45.07**	**47.63**	49.00	50.90

Table 6: MQ2008-agg NDCG@1-5 results; CRF is trained on the full data, CRF* is trained on a subset of the data with experts 13, 20, 24 and 25 removed.

document actually gets a boost in ranking. These experts are clearly not useful for ranking so we removed them and retrained the model on the remaining 21 experts. The results are shown in Table 6, from the table we see that retrained model CRF* either performs comparably or outperforms the original model. This further supports the conclusion that useful insight into expert quality can be gained by analyzing weights learned by the model. Such analysis can be particularly useful in crowdsourcing and related domains where the goal is often to identify most accurate/reliable labelers from the crowd.

4. Conclusion and Future Work

In this work we have investigated the preference aggregation problem. The explosion in online technology has generated an immense amount of preference data and introduced many new ways to express and mine preferences. As the social web activity continues to grow effective preference aggregation techniques become increasingly more important as they have a direct impact on the success or failure of many web applications.

Machine learning has recently become the technique of choice to automatically mine preferences and learn effective aggregating functions. Building on the success of existing methods we have investigated both supervised and unsupervised aggregation problems and introduced new machine learning models for each problem type.

In the unsupervised problem we introduced a new probabilistic model over preferences based on a multinomial generative process. Preferences over items are expressed through real valued scores resulting in a convex optimization problem during inference which can be solved efficiently with standard gradient based techniques. Modeling the general partial pairwise preferences makes the model applicable to a wide range of preference aggregation problems. Empirically we have shown that our approach outperforms existing unsupervised aggregation methods on two unrelated problems: rank aggregation and collaborative filtering. Future work includes in this area includes investigating how the learned variances can be used to improve the final ranking. Another interesting direction is to explore mixtures of the MPM distributions where each mixing component is parametrized by its own set of scores. This idea is similar to the mixture of Mallows models discussed above and could be employed to learn a model for different user preference types and used for personalized recommendation.

In the supervised domain we have introduced a supervised extension to the Multinomial Preference Model as well as two fully supervised preference aggregation models. All of the presented approaches are based on pairwise preference matrices, and can also be applied to a variety of problems with different preference types. Both supervised models fully use the

labeled training data and can optimize the aggregating function for any target IR metric. The first model allows to apply any learning-to-rank algorithm during optimization and can thus be easily incorporated into many existing IR frameworks. The second model has a more involved learning procedure but is significantly faster during inference time and produces interpretable results. The two models thus offer a trade-off between ease of use and speed allowing the user to make an appropriate choice based on systems requirements. Future work in this domain involves applying these models to preference aggregation problems with other forms of expert and ground truth preferences. We also plan to investigate other ways of producing effective document representations from full or partial preferences.

Acknowledgments

We would like to thank Craig Boutilier, Hugo Larochelle, and Ruslan Salakhutdinov for many thoughtful discussions and suggestions. This research was supported by the Canadian Natural Sciences and Engineering Council (NSERC) and the Canadian Institute for Advanced Research (CIFAR).

References

E. Agichtein, E. Brill, and S. Dumais. Improving Web search ranking by incorporating user behavior information. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006.

K. J. Arrow. *Social Choice and Individual Values*. Yale University Press, 1951.

J. A. Aslam and M. Montague. Models for metasearch. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

R. Bradley and M. Terry. Rank analysis of incomplete block designs. I. The method of paired comparisons. *Biometrika*, 39, 1952.

C. J. C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical Report MSR-TR-2010-82, Microsoft Research, 2010.

C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *International Conference on Machine Learning*, 2005.

C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *Neural Information Processing Systems*, 2006.

T. S. Caetano, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. In *International Conference on Machine Learning*, 2009.

O. Chapelle, Y. Chang, and T.-Y. Liu. The Yahoo! learning to rank challenge, 2010. URL `http://learningtorankchallenge.yahoo.com/`.

S. Chen, F. Wang, Y. Song, and C. Zhang. Semi-supervised ranking aggregation. *Information Processing and Management*, 47, 2011.

Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. A short introduction to computational social choice. In *International Conference on Current Trends in Theory and Practice of Computer Science*, 2007.

G. V. Cormack, C. L. A. Clarke, and S. Büttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2009.

P. Dangauthier, R. Herbrich, T. Minka, and T. Graepel. TrueSkill through time: Revisiting the history of chess. In *Neural Information Processing Systems*, 2007.

H. A. David. *The Method of Paired Comparissons*. Hodder Arnold, 1988.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41, 1990.

A. E. Elo. *The Rating of Chess Players: Past and Present*. Acro Publishing, 1978.

R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *International Conference on Management of Data*, 2003.

K. Gimpel and N. A. Smith. Softmax-margin CRFs: Training log-linear models with cost functions. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2010.

D. F. Gleich and L.-H. Lim. Rank aggregation via nuclear norm minimization. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2011.

J. Guiver and E. Snelson. Bayesian inference for Plackett-Luce ranking models. In *International Conference on Machine Learning*, 2009.

F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y.-M. Wang, and C. Faloutsos. Click chain model in Web search. In *International World Wide Web Conference*, 2009.

J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999. URL `http://www.grouplens.org/node/73`.

K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

X. Jiang, L.-H. Lim, Y. Yao, and Y. Ye. Statistical ranking and combinatorial hodge theory. *Mathematical Programming*, 127, 2011.

T. Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2002.

T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Science*, 25, 2007.

A. Klementiev, D. Roth, and K. Small. Unsupervised rank aggregation with distance-based models. In *International Conference on Machine Learning*, 2008.

G. Lebanon and J. Lafferty. Cranking: Combining rankings using conditional probability models on permutations. In *International Conference on Machine Learning*, 2002.

H. Li. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan Claypool, 2011.

T. Liu, J. Xu, W. Xiong, and H. Li. LETOR: Benchmark dataset for search on learning to rank for information retrieval. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2007a.

Y. Liu, J. Carbonell, P. Weigele, and V. Gopalakrishnan. Protein fold recognition using segmentation conditional random fields (SCRFs). *Computational Biology*, 2006.

Y.-T. Liu, T.-Y. Liu, T. Qin, Z.-M. Ma, and H. Li. Supervised rank aggregation. In *International World Wide Web Conference*, 2007b.

T. Lu and C. Boutilier. Learning Mallows models with pairwise preferences. In *International Conference on Machine Learning*, 2011.

R. D. Luce. *Individual Choice Behavior: A Theoretical Analysis*. Wiley, 1959.

C. L. Mallows. Non-null ranking models. *Biometrika*, 44, 1957.

B. M. Marlin, R. S. Zemel, and S. T. Roweis. Unsupervised learning with non-ignorable missing data. In *International Conference on Artificial Intelligence and Statistics*, 2005.

D. Mase. A penalized maximum likelihood approach for the ranking of college football teams independent of victory margins. *The American Statistician*, 57, 2003.

D. McAllester and J. Keshet. Generalization bounds and consistency for latent structural probit and ramp loss. In *Neural Information Processing Systems*, 2011.

M. Meila, K. Phadnis, A. Patterson, and J. Bilmes. Consensus ranking under the exponential model. In *International Conference on Uncertainty in Artificial Intelligence*, 2007.

M. Montague and J. A. Aslam. Condorcet fusion for improved retrieval. In *International Conference on Information and Knowledge Management*, 2002.

R. Plackett. The analysis of permutations. *Applied Statistics*, 24, 1975.

T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li. Global ranking using continuous conditional random fields. In *Neural Information Processing Systems*, 2008.

T. Quin, X. Geng, and T.-Y. Liu. A new probabilistic model for rank aggregation. In *Neural Information Processing Systems*, 2010.

F. Rossi, K. Brent Venable, and T. Walsh. *A Short Introduction to Preferences: Between Artificial Intelligence and Social Choice*. Morgan & Claypool Publishers, 2011.

D. Roth and W.-Y. Yih. Integer linear programming inference for conditional random fields. In *International Conference on Machine Learning*, 2005.

R. Salakhutdinov and A. Mnih. Restricted boltzmann machines for collaborative filtering. In *Neural Information Processing Systems*, 2008.

K. Sato and Y. Sakakibara. RNA secondary structural alignment with conditional random fields. *Bioinformatics*, 2005.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2003.

L. L. Thurstone. The method of paired comparisons for social values. *Abnormal and Social Psychology*, 21, 1927.

M. N. Volkovs and R. S. Zemel. BoltzRank: Learning to maximize expected ranking gain. In *International Conference on Machine Learning*, 2009.

M. N. Volkovs and R. S. Zemel. A flexible generative model for preference aggregation. In *International World Wide Web Conference*, 2012.

M. N. Volkovs and R. S. Zemel. CRF framework for supervised preference aggregation. In *International Conference on Information and Knowledge Management*, 2013.

M. N. Volkovs, H. Lorochelle, and R. S. Zemel. Learning to rank by aggregating expert preferences. In *International Conference on Information and Knowledge Management*, 2012.

Journal of Machine Learning Research 15 (2014) 1177-1213 Submitted 2/13; Revised 10/13; Published 3/14

Prediction and Clustering in Signed Networks: A Local to Global Perspective

Kai-Yang Chiang KYCHIANG@CS.UTEXAS.EDU
Cho-Jui Hsieh CJHSIEH@CS.UTEXAS.EDU
Nagarajan Natarajan NAGA86@CS.UTEXAS.EDU
Inderjit S. Dhillon INDERJIT@CS.UTEXAS.EDU
Department of Computer Science
University of Texas at Austin
Austin, TX 78701, USA

Ambuj Tewari TEWARIA@UMICH.EDU
Department of Statistics, and
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109, USA

Editor: Jure Leskovec

Abstract

The study of social networks is a burgeoning research area. However, most existing work is on networks that simply encode whether relationships exist or not. In contrast, relationships in *signed* networks can be positive ("like", "trust") or negative ("dislike", "distrust"). The theory of social balance shows that signed networks tend to conform to some local patterns that, in turn, induce certain global characteristics. In this paper, we exploit both local as well as global aspects of social balance theory for two fundamental problems in the analysis of signed networks: sign prediction and clustering. Local patterns of social balance have been used in the past for sign prediction. We define more general measures of social imbalance (MOIs) based on ℓ-cycles in the network and give a simple sign prediction rule. Interestingly, by examining measures of social imbalance, we show that the classic Katz measure, which is used widely in unsigned link prediction, also has a balance theoretic interpretation when applied to signed networks. Motivated by the global structure of balanced networks, we propose an effective low rank modeling approach for both sign prediction and clustering. We provide theoretical performance guarantees for our low-rank matrix completion approach via convex relaxations, scale it up to large problem sizes using a matrix factorization based algorithm, and provide extensive experimental validation including comparisons with local approaches. Our experimental results indicate that, by adopting a more global viewpoint of social balance, we get significant performance and computational gains in prediction and clustering tasks on signed networks. Our work therefore highlights the usefulness of the global aspect of balance theory for the analysis of signed networks.

Keywords: signed networks, sign prediction, balance theory, low rank model, matrix completion, graph clustering

1. Introduction

The study of networks is a highly interdisciplinary field that draws ideas and inspiration from multiple disciplines including biology, computer science, economics, mathematics, physics, sociology, and statistics. In particular, *social network analysis* deals with networks that form between people. With roots in sociology, social network analysis has evolved considerably. Recently, a major force in its evolution has been the growing importance of online social networks that were themselves enabled by the Internet and the World Wide Web. A natural result of the proliferation of online social networks has been the increased involvement in social network analysis of people from computer science, data mining, information studies, and machine learning.

Traditionally, online social networks have been represented as graphs, with nodes representing entities, and edges representing relationships between entities. However, when a network has like/dislike, love/hate, respect/disrespect, or trust/distrust relationships, such a representation is inadequate since it fails to encode the *sign* of a relationship. Recently, online networks where two opposite kinds of relationships can occur have become common. For example, online review websites such as Epinions allow users to either like or dislike other people's reviews. Such networks can be modeled as *signed networks*, where edge weights can be either greater or less than 0, representing positive or negative relationships respectively. The development of theory and algorithms for signed networks is an important research task that cannot be successfully carried out by merely extending the theory and algorithms for unsigned networks in a straightforward way. First, many notions and algorithms for unsigned networks break down when edge weights are allowed to be negative. Second, there are some interesting theories that are applicable only to signed networks.

Perhaps the most basic theory that is applicable to signed social networks but does not appear in the study of unsigned networks is that of *social balance* (Harary, 1953; Cartwright and Harary, 1956). The theory of social balance states that relationships in friend-enemy networks tend to follow patterns such as "an enemy of my friend is my enemy" and "an enemy of my enemy is my friend". A notion called weak balance (Davis, 1967) further generalizes social balance by arguing that in many cases an enemy of one's enemy can indeed act as an enemy. Both strong and weak balance are defined in terms of *local* structure at the level of triangles. Interestingly, the local structure dictated by balance theory also leads to a special *global* structure of signed networks. We review the connection between local and global structure of balance signed networks in Section 2.

Social balance has been shown to be useful for prediction and clustering tasks for signed networks. For instance, consider the *sign prediction problem* where the task is to predict the (unknown) sign of the relationship between two given entities. Ideas derived from local balance of signed networks can be successfully used to obtain algorithms for sign prediction (Leskovec et al., 2010a; Chiang et al., 2011). In addition, the *clustering problem* of partitioning the nodes of a graph into tightly knit clusters turns out to be intimately related to weak balance theory. We will see how a clustering into mutually antagonistic groups naturally emerges from weak balance theory (see Theorem 8 for more details).

The goal of this paper is to develop algorithms for prediction and clustering in signed networks by adopting the local to global perspective that is already present in the theory of social balance. What we find particularly interesting is that the local-global interplay

that occurs in the *theory* of social balance also occurs in our *algorithms*. We hope to convince the reader that, even though the local and global definitions of social balance are theoretically equivalent, algorithmic and performance gains occur when a more global approach in algorithm design is adopted.

We mentioned above that a key challenge in designing algorithms for signed networks is that the existing algorithms for unsigned networks may not be easily adapted to the signed case. For example, it has been shown that spectral clustering algorithms for unsigned networks cannot, in general, be directly extended to signed networks (Chiang et al., 2012). However, we do discover interesting connections between methods meant for unsigned networks and those meant for signed networks. For instance, in the context of sign prediction, we see that the *Katz measure*, which is widely used for unsigned link prediction, actually has a justification as a sign prediction method in terms of balance theory. Similarly, methods based on *low rank matrix completion* can be motivated using the global viewpoint of balance theory. Thus, we see that existing methods for unsigned network analysis can reappear in signed network analysis albeit due to different reasons.

Here are the key contributions we make in this paper:

- We provide a local to global perspective of the sign prediction problem, and show that our global methods are superior on synthetic as well as real-world data sets.
- In particular, we propose sign prediction methods based on local structures (triads and higher-order cycles) and low-rank modeling. The methods that use local structures are motivated by a local viewpoint of social balance, whereas the low-rank modeling approach can be viewed as a global approach motivated by a corresponding global viewpoint of social balance.
- We show that the Katz measure used for unsigned networks can be interpreted from a social balance perspective: this immediately yields a sign prediction method.
- We provide theoretical guarantees for sign prediction and signed network clustering of balanced signed networks, under mild conditions on their structure.
- We provide comprehensive experimental results that establish the superiority of global methods over local methods studied in the past and other state-of-the-art approaches.

Parts of this paper have appeared previously in Chiang et al. (2011) and Hsieh et al. (2012). The sign prediction methods based on paths and cycles were first presented in Chiang et al. (2011), and low-rank modeling in Hsieh et al. (2012). In this paper, we provide a more detailed and unified treatment of our previous research; in particular, we provide a local-to-global perspective of the proposed methods, and a much more comprehensive theoretical and experimental treatment.

The organization of this paper is guided by the local versus global aspects of social balance theory. We first review some basics of signed networks and balance theory in Section 2. We recall notions such as (strong) balance and weak balance while emphasizing the connections between local and global structures of balanced signed networks. We will see that local balance structure is revealed by triads (triangles) and cycles, while global balance structure manifests itself as clusterability of the nodes in the network. An understanding

of the local viewpoint of social balance as well as its global implications are reviewed in Section 2 to help the reader better appreciate the methods developed in the paper.

We introduce sign prediction methods motivated from the local viewpoint of social balance in Section 3. In particular, we propose measures of social imbalance (MOIs), and a simple sign prediction method for a given measure of imbalance. The proposed measures of imbalance satisfy the property that they are zero if and only if the network is balanced. While the imbalance measure based on triads has already been studied, the local definition of social balance based on general cycles can be used to obtain a more general measure of imbalance. We also propose a simple and efficient relaxation of the proposed measure. We show the validity of the relaxation in Theorem 10. An infinite-order version of the measure for sign prediction leads to a familiar proximity measure for (unsigned) networks called the Katz measure, as stated in Theorem 12. Other than serving to reinterpret the Katz measure from the perspective of social balance in signed networks, this result is the first connection that we see between link prediction in unsigned networks and sign prediction in signed networks. The sign prediction method recently proposed by Leskovec et al. (2010a) was also motivated by the local definition of social balance. This method, however, is limited to using triangles in the network. In our experiments, we observe that a variant of their method that considers higher-order cycles performs better.

In Section 4, we develop a completely global approach based on the global structure of balanced signed networks. We appeal to the global clusterability of complete weakly balanced networks (stated in Theorem 8) to develop our global approach. Broadly, we show that such networks have low rank adjacency matrices, so that we can solve the sign prediction problem by reducing it to a low rank matrix completion problem. Specifically, we show that the adjacency matrix of a complete k-weakly balanced network, that can be partitioned into $k > 2$ groups such that within-cluster edges are positive and the rest are negative, has rank k (Theorem 13). The result follows by observing that the column space of the signed adjacency matrix is spanned by a set of k linearly independent vectors corresponding to the k groups.

Our approach attempts to fill in the unobserved (missing) edges of a signed network so that the resulting network is weakly balanced. Our result on the low-rank nature of signed adjacency matrices allows us to pose the sign prediction problem as a low-matrix completion problem. The inherent rank constraint in the problem is non-convex. Therefore, we resort to using multiple approximation strategies for solving the problem. First, we look at a standard convex relaxation of rank constraint using the trace norm. The approach comes with recovery guarantees due to Candés and Tao (2009), and it requires the adjacency matrix to be ν-incoherent (see Definition in (10)). We analytically show that incoherence, in the case of complete k-weakly balanced signed networks, is directly related to the notion of *group imbalance*, which measures how skewed the group sizes are (Theorem 17). We would expect a large group imbalance to make the recovery of the adjacency matrix harder. We rigorously show the recovery guarantee in terms of group imbalance for signed networks in Theorem 18. Unfortunately, solving the aforementioned convex relaxation is computationally prohibitive in practice. We discuss two approaches for approximately solving the low-rank matrix completion problem: one based on Singular Value Projection proposed by Jain et al. (2010) and the other based on matrix factorization, which is both scalable and empirically more accurate.

Furthermore, the low rank modeling approach can also be used for the clustering of a signed network. Our clustering method proceeds as follows. First, we use a low-rank matrix completion algorithm on its adjacency matrix. Then we cluster the top-k eigenvectors of the completed matrix using any feature-based clustering algorithm. By doing so, we show that under the same assumptions that guarantee the recovery of signed networks, the true clusters can be identified from the top-k eigenvectors (Theorem 19).

In Section 5, we show some evidence of local and global balance in real networks. Our experiments on synthetic and real networks show that global methods (based on low rank models) generally perform better, in terms of accuracy of sign prediction, than local methods (based on triads and cycles). Finally, we discuss related work in Section 6, and state our conclusions in Section 7.

2. Signed Networks and Social Balance

In this section, we set up our notation for signed networks, review the basic notions of balance theory, and describe the two main tasks (sign prediction and clustering) addressed in this paper.

2.1 Categories of Signed Networks

The most basic kind of a signed network is a *homogeneous* signed network. Formally, a homogeneous signed network is represented as a graph with the adjacency matrix $A \in \{-1, 0, 1\}^{n \times n}$, which denotes relationships between entities as follows:

$$A_{ij} = \begin{cases} 1, & \text{if } i \ \& \ j \text{ have positive relationship,} \\ -1, & \text{if } i \ \& \ j \text{ have negative relationship,} \\ 0, & \text{if relationship between } i \ \& \ j \text{ is unknown (or missing).} \end{cases}$$

We should note that we treat a zero entry in A as an *unknown* relationship instead of no relationship, since we expect any two entities have some (hidden) positive or negative attitude toward each other even if the relationship itself might not be observed. From an alternative point of view, we can assume there exists an underlying *complete* signed network $A^\star$, which contains relationship information between all pairs of entities. However, we only observe some entries of $A^\star$, denoted by Ω. Thus, the partially observed network A can be represented as:

$$A_{ij} = \begin{cases} A^\star_{ij}, & \text{if } (i,j) \in \Omega, \\ 0, & \text{otherwise.} \end{cases}$$

A signed network can also be *heterogeneous*. In a heterogeneous signed network, there can be more than one kind of entity, and relationships between two, same or different, entities can be positive and negative. For example, in the online video sharing website Youtube, there are two kinds of entities—users and videos, and every user can either *like* or *dislike* a video. Therefore, the Youtube network can be seen as a bipartite signed network, in which the positive and negative links are between users and videos.

In this paper, we will focus our attention on homogeneous signed networks, that is, networks where relationships are between the same kind of entities. For heterogeneous

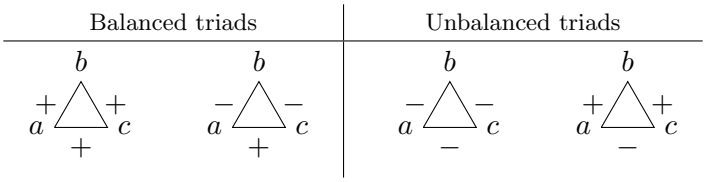


Table 1: Configurations of balanced and unbalanced triads.

signed networks, it is possible to do some preprocessing to reduce them to homogeneous networks. For instance, in a Youtube network, we could possibly infer the relationships between users based on their taste of videos. These preprocessing tasks, however, are not trivial.

In the remaining part of the paper, we will use the term "network" as an abbreviation for "signed network", unless we explicitly specify otherwise. In addition, we will now mainly focus on undirected signed graphs (i.e., A is symmetric) unless we specify otherwise. For a directed signed network, a simple but sub-optimal way to apply our methods is by considering the symmetric network, $\text{sign}(A + A^T)$. Of course, making the network symmetric erases edges with conflicting signs between a pair of nodes. It is important to know how much information is lost in the process. We found that in real networks, the percentage of conflicting edges is extremely small (see Table 4 in Section 5). The observation suggests that making the network undirected preserves the sign structure for the most part, and is sufficient for analysis.

2.2 Social Balance

A key idea behind many methods that estimate a high dimensional complex object from limited data is the exploitation of *structure*. In the case of signed networks, researchers have identified various kinds of non-trivial structure (Harary, 1953; Davis, 1967). In particular, one influential theory, known as social balance theory, states that relationships between entities tend to be *balanced*. Formally, we say a triad (or a triangle) is *balanced* if it contains an even number of negative edges. This is in agreement with beliefs such as "a friend of my friend is more likely to be my friend" and "an enemy of my friend is more likely to be my enemy". The configurations of balanced and unbalanced triads are shown in Table 1.

Though social balance specifies the patterns of triads, one can generalize the balance notion to general ℓ-cycles. An ℓ-cycle is defined as a simple path from some node to itself with length equal to ℓ. The following definition extends social balance to general ℓ-cycles:

Definition 1 (Balanced ℓ-cycles) *An ℓ-cycle is said to be balanced when it contains an even number of negative edges.*

Table 2 shows some instances of balanced and unbalanced cycles based on the above definition. To define balance for general networks, we first define the notion of balance for *complete* networks:

Definition 2 (Balanced complete networks) *A complete network is said to be balanced when all triads in the network are balanced.*

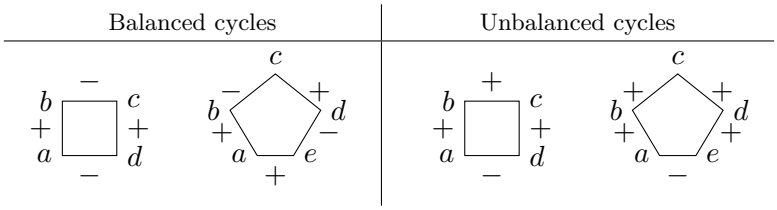


Table 2: Some instances of balanced and unbalanced cycles.

Of course, most real networks are not complete. To define balance for general networks, we adopt the perspective of a missing value estimation problem as follows:

Definition 3 (Balanced networks) *A (possibly incomplete) network is said to be balanced when it is possible to assign* ± 1 *signs to all missing entries in the adjacency matrix, such that the resulting complete network is balanced.*

So far, the notion of balance is defined by specifying patterns of *local* structures in networks (i.e., the patterns of triads). The following result from balance theory shows that balanced networks have a *global* structure.

Theorem 4 (Balance theory, Cartwright and Harary, 1956) *A network is balanced iff either (i) all edges are positive, or (ii) we can divide nodes into two clusters (or groups), such that all edges within clusters are positive and all edges between clusters are negative.*

Now we revisit balanced ℓ-cycles defined at the beginning of this subsection. Under that definition, we can verify if a network is balanced or not by looking at all cycles in the network due to the following well-known theorem (whose proof can be found in Easley and Kleinberg (2010, Chapter 5)).

Theorem 5 *A network is balanced iff all its ℓ-cycles are balanced.*

One possible weakness of social balance theory is that the defined balance relationships might be too strict. In particular, researchers have argued that the degree of imbalance in the triad with two positive edges (the fourth triad in Table 1) is much stronger than that in the triad with all negative edges (the third triad in Table 1). By allowing triads with all negative edges, a weaker version of balance notion can be defined (Davis, 1967).

Definition 6 (Weakly balanced complete networks) *A complete network is said to be weakly balanced when all triads in the network are weakly balanced.*

Note that ℓ-cycles with an odd number of negative edges are allowed under weak balance. The definition for general incomplete networks can be obtained by adopting the perspective of a missing value estimation problem:

Definition 7 (Weakly balanced networks) *A (possibly incomplete) network is said to be weakly balanced when it is possible to obtain a weakly balanced complete network by filling the missing edges in its adjacency matrix.*

Though the above definitions define weak balance in terms of patterns of local triads, one can show that weakly balanced networks have a special global structure, analogous to Theorem 4:

Theorem 8 (Weak balance theory, Davis 1967) *A network is weakly balanced iff either (i) all of its edges are positive, or (ii) we can divide nodes into k clusters, such that all the edges within clusters are positive and all the edges between clusters are negative.*

Note that when $k = 2$, this theorem simply reduces to Theorem 4.

At this juncture, it would be helpful to recall that balance and weak balance are more natural for the analysis of undirected (signed) networks. More recently, Leskovec et al. (2010b) analyzed directed signed networks using the concept of "status", that is characteristic of directed networks. We do not explore the "status structure" of signed networks in this paper, but the theory seems to be promising and worthy of study in the future.

2.3 Key Problems in Signed Network Analysis

As in classical social network analysis, we are interested in what we can infer given a signed network topology. In particular, we will focus on two core problems—sign prediction and clustering.

In the *sign prediction* problem, we intend to infer the unknown relationship between a pair of entities i and j based on partial observations of the entire network of relationships. More specifically, if we assume that we are given a (usually incomplete) network A sampled from some underlying (complete) network $A^\star$, then the sign prediction task is to recover the sign patterns of one or more edges in $A^\star$. This problem bears similarity to the *structural link prediction problem* in *unsigned* networks (Liben-Nowell and Kleinberg, 2007; Menon and Elkan, 2011). Note that the *temporal link prediction problem* has also been studied in the context of an unsigned network evolving in time. The input to the prediction algorithm then consists of a series of networks (snapshots) instead of a single network. We do not consider such temporal problems in this paper.

Clustering is another important problem in network analysis. Recall that according to weak balance theory (Theorem 8), we can find k groups such that they are mutually antagonistic in a weakly balanced network. Motivated by this, the clustering task in a signed network is to identify k antagonistic groups in the network, such that most edges within the same cluster are positive while most edges between different clusters are negative. Notice that since the (weak) balance notion only applies to signed networks, most traditional clustering algorithms for unsigned networks cannot be directly applied.

3. Local Methods for Sign Prediction

The definition of structural balance based on triangles is local in nature. A natural approach for designing sign prediction algorithms proceeds by reasoning locally in terms of unbalanced triangles, which motivates the following *measure of imbalance*:

$$\mu_{\text{tri}}(A) := \sum_{\tilde{\sigma} \in SC_3(A)} \mathbf{1}\left[\tilde{\sigma} \text{ is unbalanced}\right] , \tag{1}$$

where $SC_3(A)$ refers to the set of triangles (simple cycles of length-3) in the network A. In general, we use $SC_\ell(A)$ to denote the set of all simple ℓ-cycles in the network A. A definition essentially similar to the one above appears in the recent work of van de Rijt (2011, p. 103) who observes that the equivalence

$$\mu_{\text{tri}}(A) = 0 \text{ iff } A \text{ is balanced}$$

holds only for complete graphs. For an incomplete graph, imbalance might manifest itself only if we look at longer simple cycles. Accordingly, we define a higher-order analogue of (1),

$$\mu^s_\ell(A) := \sum_{i=3}^{\ell} \beta_i \sum_{\tilde{\sigma} \in SC_i(A)} \mathbf{1}\left[\tilde{\sigma} \text{ is unbalanced}\right] , \tag{2}$$

where $\ell \geq 3$ and β_i's are coefficients weighting the relative contributions of unbalanced simple cycles of different lengths. If we choose a decaying choice of β_i, like $\beta_i = \beta^i$ for some $\beta \in (0, 1)$, then we can even define an infinite-order version,

$$\mu^s_\infty(A) := \sum_{i \geq 3} \beta_i \sum_{\tilde{\sigma} \in SC_i(A)} \mathbf{1}\left[\tilde{\sigma} \text{ is unbalanced}\right] . \tag{3}$$

It is clear that $\mu^s_\infty(\cdot)$ is a genuine measure of imbalance in the sense formalized by the following corollary of Theorem 5.

Corollary 9 *Fix an observed graph A. Let $\beta_i > 0$ be any sequence such that the infinite sum in* (3) *is well-defined. Then, $\mu^s_\infty(A) > 0$ iff A is unbalanced.*

The basic idea of using a measure of imbalance for predicting the sign of a given query link $\{i, j\}$, such that $i \neq j$ and $A_{i,j} = 0$, is as follows. Given the observed graph A and query $\{i, j\}$, $i \neq j$, we construct two graphs: $A^{+(i,j)}$ and $A^{-(i,j)}$. These are obtained from A by setting A_{ij} and A_{ji} to $+1$ and -1 respectively. Formally, these two augmented graphs are defined as:

$$A^{+(i,j)}_{uv} = \begin{cases} 1, & \text{if } (u,v) = (i,j) \text{ or } (j,i) \\ A_{uv}, & \text{otherwise.} \end{cases} \qquad A^{-(i,j)}_{uv} = \begin{cases} -1, & \text{if } (u,v) = (i,j) \text{ or } (j,i) \\ A_{uv}, & \text{otherwise.} \end{cases}$$

Then, given a measure of imbalance, denoted as $\mu(\cdot)$, the predicted sign of $\{i, j\}$ is simply:

$$\text{sign}\left(\mu\left(A^{-(i,j)}\right) - \mu\left(A^{+(i,j)}\right)\right) . \tag{4}$$

Note that, to use the above for prediction, we should use a $\mu(\cdot)$ for which the quantity (4) is efficiently computable. The measure of imbalance based on triads $\mu_{\text{tri}}(A)$ and the more general measure $\mu^s_\infty(A)$ involve counting simple cycles in a graph. However, enumerating simple cycles of a graph is NP-hard.[1] To get around this computational issue, we slightly change the definition of $\mu_\ell(\cdot)$ to the following.

$$\mu_\ell(A) := \sum_{i=3}^{\ell} \beta_i \sum_{\sigma \in C_i(A)} \mathbf{1}\left[\sigma \text{ is unbalanced}\right] . \tag{5}$$

1. By straightforward reduction to the Hamiltonian cycle problem (Karp, 1972).

As before, we allow $\ell = \infty$ provided the β_i's decay sufficiently rapidly.

$$\mu_\infty(A) := \sum_{i \geq 3} \beta_i \sum_{\sigma \in C_i(A)} \mathbf{1}\left[\sigma \text{ is unbalanced}\right] \ . \tag{6}$$

The only difference between these definitions and (2),(3) is that here we sum over *all* cycles (denoted by $C_i(A)$), not just simple ones. However, we still get a valid notion of imbalance as stated by the following result.

Theorem 10 *Fix an observed graph A. Let $\beta_i > 0$ be any sequence such that the infinite sum in* (6) *is well-defined. Then, $\mu_\infty(A) > 0$ iff A is unbalanced.*

It turns out, somewhat surprisingly, that computing (4) with $\mu(A) = \mu_{\text{tri}}(A)$ simply reduces to computing (i, j) entry in A^2. The following key lemma gives an efficient way to compute (4) with $\mu(\cdot) = \mu_\ell(\cdot)$ in general. Indeed, it amounts to computing higher powers of the adjacency matrix.

Lemma 11 *Fix A and let $i \neq j$ be such that $(i, j) \notin \Omega$. Let $A^{+(i,j)}$ and $A^{-(i,j)}$ be the augmented graphs. Then, for any $\ell \geq 3$,*

$$\sum_{\sigma \in C_\ell\left(A^{-(i,j)}\right)} \mathbf{1}\left[\sigma\right] \quad - \sum_{\sigma \in C_\ell\left(A^{+(i,j)}\right)} \mathbf{1}\left[\sigma\right] = A^{\ell-1}_{i,j} \ ,$$

where $\mathbf{1}\left[\sigma\right]$ is the abbreviation of $\mathbf{1}\left[\sigma \text{ is unbalanced}\right]$.

Using Lemma 11, it is easy to see that the prediction (4) using (5) reduces to

$$\text{sign}\left(\mu_\ell\left(A^{-(i,j)}\right) - \mu_\ell\left(A^{+(i,j)}\right)\right) = \text{sign}\left(\sum_{t=3}^{\ell} \beta_t A^{t-1}_{i,j}\right) \ ,$$

and the prediction (4) using (6) reduces to

$$\text{sign}\left(\mu_\infty\left(A^{-(i,j)}\right) - \mu_\infty\left(A^{+(i,j)}\right)\right) = \text{sign}\left(\sum_{\ell \geq 3} \beta_\ell A^{\ell-1}_{i,j}\right) \ . \tag{7}$$

Lemma 11 is also key to interpreting a classical proximity measure called the *Katz* measure in the context of sign prediction, discussed next. Proofs of Theorem 10 and Lemma 11 are presented in Appendix A.

3.1 Katz Measure is Valid for Signed Networks

The classic method of Katz (1953) has been used successfully for *unsigned* link prediction (Liben-Nowell and Kleinberg, 2007). Formally, given an unsigned network A with a query node-pair (i, j), the Katz measure for the link (i, j) is defined as:

$$Katz(i, j) := \sum_{\ell=2}^{\infty} \beta^\ell A^\ell_{ij},$$

where $\beta > 0$ is a constant so that the above infinite sum is well defined ($\beta < \frac{1}{\|A\|_2}$ suffices, where $\|A\|_2$ is the spectral norm of A).[2] Intuitively, the Katz measure sums up all possible paths between i and j. The number of length ℓ paths can grow exponentially as ℓ increases (for example, there are $\Omega(n^{\ell-1})$ length ℓ paths between i and j if the network is complete). Therefore, the contributions from length ℓ paths are exponentially damped by the constant β^ℓ. One can also verify the above definition has the following matrix form:

$$Katz(i,j) = ((I - \beta A)^{-1} - I - \beta A)_{ij}.$$

A higher Katz score indicates more proximity between nodes i and j, and the link (i,j) is therefore more likely to exist or to form in the future.

While Katz has been used as an effective proximity measure for link prediction in unsigned networks, it is not obvious what the *signed* Katz score corresponds to in signed networks. Following (7), the connection between Katz measure and $\mu_\infty(\cdot)$ stands out. The following theorem shows that by considering a sign prediction method based on $\mu_\infty(\cdot)$ we obtain an interesting interpretation of the Katz measure on a signed network from a balance theory viewpoint.

Theorem 12 (Balance Theory Interpretation of the Katz Measure) *Consider the sign prediction rule* (4) *using* $\mu_\infty(\cdot)$ *in the reduced form* (7). *In the special case when* $\beta_\ell = \beta^{\ell-1}$ *with* β *small enough* ($\beta < 1/\|A\|_2$), *the rule can be expressed as the Katz prediction rule for edge sign prediction, in closed form:*

$$\operatorname{sign}\left(\left((I - \beta A)^{-1} - I - \beta A\right)_{i,j}\right) .$$

Our sign prediction rule for a given measure of imbalance relies on social balance theory for signed networks. However, real world networks may not conform to the prediction of balance theory or may do so only to a certain extent. Furthermore, balance theory was developed for undirected networks and hence methods based on measures of imbalance can deal only with undirected networks. To partly mitigate the lapse, we can use measures of imbalance to derive *features* that can then be fed to a supervised learning algorithm (like logistic regression) along with the signs of the known edges in the network. When we learn weights for such features, we are weakening our reliance on social balance theory but can naturally deal with directed graphs. By using features constructed from higher-order cycles, we extend the supervised approach used by Leskovec et al. (2010a) that was limited to learning from triads-based features. While the learning approach itself is straightforward, construction of higher-order features for directed signed networks requires some attention. We defer the details to Appendix B. In the experiments (Section 5), we denote the local method (7) corresponding to the measure of imbalance based on cycles of length ℓ by MOI-ℓ. Note that MOI-∞ refers to the signed Katz measure. The supervised learning method

2. Our definition is slightly different from the definition in Liben-Nowell and Kleinberg (2007), where the summation starts from $\ell = 1$. However, the measures are identical for the purpose of link prediction, as the prediction needs to be made only when the query nodes i and j have no existing edge, that is, $A_{ij} = 0$.

discussed in Appendix B, where we use logistic regression to train weights for features constructed from higher-order cycles of length up to ℓ, is referred to as HOC-ℓ in the experiments.

4. Global Methods: Low Rank Modeling

In Section 3, we have seen how to use ℓ-cycles for sign prediction. We have also seen that ℓ-cycles play a major role in how balance structure manifests itself *locally*. By increasing ℓ, the level at which balance structure is considered becomes less localized. Still, it is natural to ask whether we can design algorithms for signed networks by directly making use of their *global* structure. To be more specific, let us revisit the definition of complete weakly balanced networks. In general, weak balance can be defined from either a local or a global point of view. From a local point of view, a given network is weakly balanced if all *triads* are weakly balanced, whereas from a global point of view, a network is weakly balanced if its *global structure* obeys the clusterability property stated in Theorem 8. Therefore, it is natural to ask whether we can directly use this global structure for sign prediction. In the sequel, we show that weakly balanced networks have a "low-rank" structure, so that the sign prediction problem can be formulated as a low rank matrix completion problem.

We begin by showing that given a complete k-weakly balanced network, its adjacency matrix $A^\star$ has rank at most k:

Theorem 13 (Low Rank Structure of Signed Networks) *The adjacency matrix $A^\star$ of a complete k-weakly balanced network has rank 1 if $k \leq 2$, and has rank k for all $k > 2$.*

Proof Since $A^\star$ is k-weakly balanced, the nodes can be divided into k groups, say $S^{(1)}, S^{(2)}, \ldots, S^{(k)}$. Suppose group $S^{(i)}$ contains nodes $s_1^{(i)}, s_2^{(i)}, \ldots, s_{n_i}^{(i)}$, then the corresponding columns vectors of $A^\star$ all have the following form (after suitable reordering of nodes):

$$\mathbf{b}_i = [-1 \quad \cdots \quad -1 \underbrace{1 \quad \cdots \quad 1}_{\text{the } i^{th} \text{ group}} -1 \quad \cdots \quad -1]^T,$$

and so the column space of $A^\star$ is spanned by $\{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$.

First consider $k \leq 2$, that is, the network is strongly balanced. If $k = 1$, it is easy to see that $\text{rank}(A^\star) = 1$. If $k = 2$, then $\mathbf{b}_1 = -\mathbf{b}_2$. Therefore, $\text{rank}(A^\star)$ is again 1.

Now consider $k > 2$. In this case, we argue that $\text{rank}(A^\star)$ exactly equals k by showing that $\mathbf{b}_1, \ldots, \mathbf{b}_k$ are linearly independent. We consider the $k \times k$ square matrix M such that $M_{ij} = -1, \forall i \neq j$ and $M_{ii} = 1, \forall i$. It is obvious that $\mathbf{1} = [1\ 1 \cdots 1]^T$ is an eigenvector of M with eigenvalue $-(k-2)$. We can further construct the other $k-1$ linearly independent eigenvectors, each with eigenvalue 2:

$$\mathbf{e}_1 - \mathbf{e}_2,\ \mathbf{e}_1 - \mathbf{e}_3,\ \ldots,\ \mathbf{e}_1 - \mathbf{e}_k,$$

where $\mathbf{e}_i \in \mathbb{R}^k$ is the i^{th} column of the $k \times k$ identity matrix. These $k-1$ eigenvectors are clearly linearly independent. Therefore, $\text{rank}(M) = k$.

From the above we can show that $\text{rank}(A^\star) = k$. Suppose that $\mathbf{b}_1, \ldots, \mathbf{b}_k$ are not linearly independent, then there exists $\alpha_1, \ldots, \alpha_k$, with some $\alpha_i \neq 0$, such that $\sum_{i=1}^k \alpha_i \mathbf{b}_i = \mathbf{0}$.

Using this set of α's, it is easy to see that $\sum_{i=1}^{k} \alpha_i M_i = \mathbf{0}$ (where M_i is the i^{th} column of M), but this contradicts the fact that $\text{rank}(M) = k$. Therefore, $\text{rank}(A^\star) = k$. ■

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & -1 & -1 & -1 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & -1 & -1 & -1 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & -1 & -1 & -1 \\ -1 & -1 & -1 & \mathbf{1} & -1 & -1 \\ -1 & -1 & -1 & -1 & \mathbf{1} & \mathbf{1} \\ -1 & -1 & -1 & -1 & \mathbf{1} & \mathbf{1} \end{pmatrix} = \begin{pmatrix} -1.045 & 0.265 & -0.402 \\ -1.045 & 0.265 & -0.402 \\ -1.045 & 0.265 & -0.402 \\ 0.319 & -1.260 & -0.830 \\ 0.919 & 0.670 & -0.541 \\ 0.919 & 0.670 & -0.541 \end{pmatrix} \begin{pmatrix} -1.045 & 0.265 & 0.402 \\ -1.045 & 0.265 & 0.402 \\ -1.045 & 0.265 & 0.402 \\ 0.319 & -1.260 & 0.830 \\ 0.919 & 0.670 & 0.541 \\ 0.919 & 0.670 & 0.541 \end{pmatrix}^T$$

Figure 1: An illustrative example of the low-rank structure of a 3-weakly balanced network. The network can be represented as a product of two rank-3 matrices, and so the adjacency matrix has rank no more than 3.

Figure 1 is an example of a complete 3-weakly balanced network. As shown, we see its adjacency matrix can be expressed as a product of two rank-3 matrices, indicating its rank is no more than three. In fact, by Theorem 13, we can conclude that its rank is exactly 3.

The above reasoning shows that (adjacency matrices of) complete weakly balanced networks have low rank. However, most real networks are not complete graphs. Recall that in order to define balance on incomplete networks, we try to fill in the unobserved or missing edges (relationships) so that balance is obtained. Following this desideratum, we can think of sign prediction in signed networks as a low-rank matrix completion problem. Specifically, suppose we observe entries $(i, j) \in \Omega$ of a complete signed network $A^\star$. We want to find a complete matrix by assigning ± 1 to every unknown entry, such that the resulting complete graph is weakly balanced and hence, the completed matrix is low rank. Thus, our missing value estimation problem can be formulated as:

$$\begin{aligned} \text{minimize} \quad & \text{rank}(X) \\ \text{s.t.} \quad & X_{ij} = A^\star_{ij}, \ \forall \ (i,j) \in \Omega, \\ & X_{ij} \in \{\pm 1\}, \ \forall \ (i,j) \notin \Omega. \end{aligned} \tag{8}$$

Once we obtain the minimizer of (8), which we will denote by $X^\star$, we can infer the missing relationship between i and j by simply looking up the sign of the entry $X^\star_{ij}$. So the question is whether we can solve (8) efficiently. In general, (8) is known to be NP-hard; however, recent research has shown the surprising result that under certain conditions, the low-rank matrix completion problem (8) can be solved by convex optimization to yield a *global* optimum in polynomial time (Candés and Recht, 2008). In the following subsections, we identify such conditions as well as approaches to approximately solve (8) for real-world signed networks.

4.1 Sign Prediction via Convex Relaxation

One possible approximate solution for (8) can be obtained by dropping the discrete constraints and replacing $\text{rank}(X)$ by the trace norm of X (denoted by $\|X\|_*$), which is the

tightest convex relaxation of rank (Fazel et al., 2001). Thus, a convex relaxation of (8) is:

$$\begin{aligned} \text{minimize} \quad & \|X\|_* \\ \text{s.t.} \quad & X_{ij} = A^\star_{ij},\ \forall\ (i,j) \in \Omega. \end{aligned} \tag{9}$$

It turns out that, under certain conditions, by solving (9) we can recover the *exact* missing relationships from the underlying complete signed network. This result is the consequence of recent research (Candés and Recht, 2008; Candés and Tao, 2009) which has shown that perfect recovery is possible if the observed entries are uniformly sampled and $A^\star$ has high incoherence, which may be defined as follows:

Definition 14 (Incoherence) *An $n \times n$ matrix X with singular value decomposition $X = U\Sigma V^T$ is ν-incoherent if*

$$\max_{i,j} |U_{ij}| \leq \frac{\sqrt{\nu}}{\sqrt{n}} \quad \textit{and} \quad \max_{i,j} |V_{ij}| \leq \frac{\sqrt{\nu}}{\sqrt{n}}. \tag{10}$$

Intuitively, higher incoherence (smaller ν) means that large entries of the matrix are not concentrated in a small part. The following theorem shows that under high incoherence and uniform sampling, solving (9) exactly recovers $A^\star$ with high probability.

Theorem 15 (Recovery Condition Candés and Tao, 2009) *Let $A^\star$ be an $n \times n$ matrix with rank k, with singular value decomposition $A^\star = U\Sigma V^T$. In addition, assume $A^\star$ is ν-incoherent. Then there exists some constant C, such that if $C\nu^4 nk^2 \log^2 n$ entries are uniformly sampled, then with probability at least $1 - n^{-3}$, $A^\star$ is the unique optimizer of* (9).

In particular, if the underlying matrix has bounded rank (i.e., $k = O(1)$), the number of sampled entries required for recovery reduces to $O(\nu^4 n \log^2 n)$.

Based on Theorem 15, we now show that the notion of incoherence can be connected to the relative sizes of the clusters in signed networks. As a result, by solving (9), we will show that we can recover the underlying signed network with high probability if there are no "small" groups. To start, we define the *group imbalance* of a signed network as follows:

Definition 16 (Group Imbalance) *Let $A^\star$ be the adjacency matrix of a complete k-weakly balanced network with n nodes, and let $n_1, \ldots, n_k$ be the sizes of the groups. Group imbalance τ of $A^\star$ is defined as*

$$\tau := \max_{i=1,\ldots,k} \frac{n}{n_i}.$$

By definition, $k \leq \tau \leq n$. Larger group imbalance τ indicates the presence of a very small group, which would intuitively make recovery of the underlying network harder (under uniform sampling). For example, consider an extreme scenario that a k-weakly balanced network contains n nodes, with two groups that contain only one node each. Then if nodes $n-1$ and n form these singleton groups, the adjacency matrix of this network has group imbalance $\tau = n$ with $A^\star_{n-1,n-1} = A^\star_{n,n} = 1$ and $A^\star_{n-1,n} = A^\star_{n,n-1} = -1$. However, without observing $A^\star_{n-1,n}$ or $A^\star_{n,n-1}$, it is impossible to determine whether the last two nodes are in the same cluster, or whether each of them belongs to an individual cluster. When n is

very large, the probability of observing one of these two entries will be extremely small. Therefore, under uniform sampling of $O(n \log^2 n)$ entries, it is unlikely that any matrix completion algorithm will be able to exactly recover this network.

Motivated by this example, we now analytically show that group imbalance τ determines the possibility of recovery. We first show the connection between τ and incoherence ν.

Theorem 17 (Incoherence of Signed Networks) *Let $A^\star$ be the adjacency matrix of a complete k-weakly balanced network with group imbalance τ. Then $A^\star$ is τ-incoherent.*

Proof Recall that incoherence ν is defined as the maximum absolute value in the (normalized) singular vectors of $A^\star$, which are identical to its eigenvectors (up to signs), since $A^\star$ is symmetric.

Let $\mathbf{u}$ be any unit eigenvector of $A^\star$ ($\|\mathbf{u}\|_2 = 1$) with eigenvalue λ. Suppose i and j are in the same group, then the i^{th} and j^{th} rows of $A^\star$ are identical, that is, $A^\star_{i,:} = A^\star_{j,:}$. As a result, the i^{th} and j^{th} elements of all eigenvectors will be identical (since $u_i = A^\star_{i,:}\mathbf{u}/\lambda = A^\star_{j,:}\mathbf{u}/\lambda = u_j$). Thus, $\mathbf{u}$ has the following form:

$$\mathbf{u} = [\underbrace{\alpha_1, \alpha_1, \ldots, \alpha_1}_{n_1}, \underbrace{\alpha_2, \ldots, \alpha_2}_{n_2}, \ldots, \underbrace{\alpha_k, \ldots, \alpha_k}_{n_k}]^T. \tag{11}$$

Because $\|\mathbf{u}\|_2 = 1$, $\sum_{i=1}^k n_i \alpha_i^2 = 1$, and so $n_i \alpha_i^2 \leq 1$, $\forall i$, which implies $|\alpha_i| \leq 1/\sqrt{n_i}$, $\forall i$. Thus,

$$\max_i |u_i| = \max_i |\alpha_i| \leq \max_i \frac{1}{\sqrt{n_i}} = \max_i \frac{\sqrt{n/n_i}}{\sqrt{n}} \leq \frac{\sqrt{\tau}}{\sqrt{n}}.$$

Therefore, $A^\star$ is τ-incoherent. ■

Putting together Theorems 15 and 17, we now have the main theorem of this subsection:

Theorem 18 (Recovery Condition for Signed Networks) *Suppose we observe edges A_{ij}, $(i,j) \in \Omega$, from an underlying k-weakly balanced signed network $A^\star$ with n nodes, and suppose that the following assumptions hold:*

A. k is bounded ($k = O(1)$),

B. the set of observed entries Ω is uniformly sampled, and

C. number of samples is sufficiently large, that is, $|\Omega| \geq C\tau^4 n \log^2 n$, where τ is the group imbalance of the underlying complete network $A^\star$.

Then $A^\star$ can be perfectly recovered by solving (9), with probability at least $1 - n^{-3}$.

In particular, if n/n_i is upper bounded so that τ is a constant, then we only need $O(n \log^2 n)$ observed entries to *exactly* recover the complete k-weakly balanced network.

4.2 Sign Prediction via Singular Value Projection

Though the convex optimization problem (9) can be solved to yield the global optimum, the computational cost of solving it might be too prohibitive in practice. Therefore, recent research provides more efficient algorithms to approximately solve (8) (Cai et al., 2010; Jain

et al., 2010). In particular, we consider the Singular Value Projection (SVP) algorithm proposed by Jain et al. (2010) which attempts to solve the low-rank matrix completion problem in an efficient manner. The SVP algorithm considers a robust formulation of (8) as follows:

$$\begin{aligned} \text{minimize} \quad & \|\mathcal{P}(X) - A\|_F^2 \\ \text{s.t.} \quad & \text{rank}(X) \leq k, \end{aligned} \tag{12}$$

where the projection operator $\mathcal{P}$ is defined as:

$$(\mathcal{P}(X))_{ij} = \begin{cases} X_{ij}, & \text{if } (i,j) \in \Omega \\ 0, & \text{otherwise.} \end{cases}$$

Note that the objective (12) recognizes that there might be some violations of weak balance in the observations A, and minimizes the squared-error instead of trying to enforce exact equality as in (9). In an attempt to optimize (12), the SVP algorithm iteratively calculates the gradient descent update $\hat{X}^{(t)}$ of the current solution $X^{(t)}$, and projects $\hat{X}^{(t)}$ onto the non-convex set of matrices whose rank are at most k using SVD. After the SVP algorithm terminates and outputs $\bar{X}$, one can take the sign of each entry of $\bar{X}$ to obtain an approximate solution of (8). The SVP procedure for sign prediction is summarized in Algorithm 1.

Algorithm 1: Sign Prediction via Singular Value Projection (SVP)

Input: Adjacency matrix A, rank k, tolerance ϵ, max iteration $t_{\max}$, step size η
Output: $\bar{X}$, the completed low-rank matrix that approximately solves (8)

1. Initialize $X^{(0)} \leftarrow 0$ and $t \leftarrow 0$.
2. Do
   - $\hat{X}^{(t)} \leftarrow X^{(t)} - \eta(\mathcal{P}(X^{(t)}) - A)$
   - $[U_k, \Sigma_k, V_k] \leftarrow$ Top k singular vectors and singular values of $\hat{X}^{(t)}$
   - $X^{(t+1)} \leftarrow U_k \Sigma_k {V_k}^T$
   - $t \leftarrow t + 1$

   while $\|\mathcal{P}(X^{(t)}) - A\|_F^2 > \epsilon$ and $t < t_{\max}$
3. $\bar{X} \leftarrow \text{sign}(X^{(t)})$

In addition to its efficiency, experimental evidence provided by Jain et al. (2010) suggests that if observations are uniformly distributed, then all iterates of the SVP algorithm are ν-incoherent, and if this occurs, then it can be shown that the matrix completion problem (8) can be exactly solved by SVP. In Section 5, we will see that SVP performs well in recovering weakly balanced networks.

4.3 Sign Prediction via Matrix Factorization

A limitation of both convex relaxation and SVP is that they require uniform sampling to ensure good performance. However, this assumption is violated in most real-life applications, and so these approaches do not work very well in practice. In addition, both the methods cannot scale to very large data sets, as they require computation of the SVD. Thus, we use a gradient based matrix factorization approach as an approximation to the signed network

completion problem. In Section 5, we will see that this matrix factorization approach can boost the accuracy of estimation as well as scale to large real networks.

In the matrix factorization approach, we consider the following problem:

$$\min_{W,H\in\mathbb{R}^{n\times k}} \sum_{(i,j)\in\Omega} (A_{ij} - (WH^T)_{ij})^2 + \lambda\|W\|_F^2 + \lambda\|H\|_F^2. \tag{13}$$

Problem (13) is non-convex, and an alternating minimization algorithm is commonly used to solve it. Recent theoretical results show that the alternating minimization algorithm provably solves (13) under similar conditions as trace-norm minimization (Jain et al., 2013). The matrix factorization approach is widely used in practical collaborative filtering applications as its performance is competitive to or better than trace-norm minimization, while scalability is much better. For example, to solve the Netflix problem, (13) has been applied with a fair amount of success to factorize data sets with 100 million ratings (Koren et al., 2009).

Nevertheless, there is an issue when modeling signed networks using (13): the squared loss in the first term of (13) tends to force entries of WH^T to be either $+1$ or -1. However, what we care about in this completion task is the consistency between $\text{sign}((WH^T)_{ij})$ and $\text{sign}(A_{ij})$ rather than their difference. For example, $(WH^T)_{ij} = 10$ should have zero loss when $A_{ij} = +1$ if only the signs are important.

To resolve this issue, instead of using the squared loss, we use a loss function that only penalizes the inconsistency in *sign*. More precisely, objective (13) can be generalized as:

$$\min_{W,H\in\mathbb{R}^{n\times k}} \sum_{(i,j)\in\Omega} \ell oss\ (A_{ij}, (WH^T)_{ij}) + \lambda\|W\|_F^2 + \lambda\|H\|_F^2. \tag{14}$$

In order to penalize inconsistency of sign, we can change the loss function to be the sigmoid or squared-hinge loss:

$$\begin{aligned} \ell oss_{\text{ sigmoid}}(x, y) &= 1/(1 + \exp(xy)), \\ \ell oss_{\text{ square-hinge}}(x, y) &= (\max(0, 1 - xy))^2. \end{aligned} \tag{15}$$

In Section 5, we will see that applying sigmoid or squared-hinge loss functions slightly improves prediction accuracy.

4.4 Time Complexity of Sign Prediction Methods

There are two main optimization techniques for solving (13) for large-scale data: Alternating Least Squares (ALS) and Stochastic Gradient Descent (SGD) (Koren et al., 2009). ALS solves the squared loss problem (13) by alternately minimizing W and H. When one of W or H is fixed, the optimization problem becomes a least squares problem with respect to the other variable, so that we can use well developed least squares solvers to solve each subproblem. Given an $n \times n$ observed matrix with m observations, it requires $O(mk^2)$ operations to form the Hessian matrices, and $O(nk^3)$ operations to solve each least squares subproblem. Therefore, the time complexity of ALS is $O(t_1(mk^2 + nk^3))$ where t_1 is the number of iterations.

HOC	LR-ALS	LR-SGD
$O(2^\ell nm)$	$O(t_1(nk^3 + mk^2))$	$O(t_2 km)$

Table 3: Time complexity of cycle-based method (HOC) and low rank modeling methods (LR-ALS, LR-SGD). The HOC time only considers feature computation time. The time for low rank modeling consists of total model construction time.

However, ALS can only be used when the loss function is the squared loss. To solve the general form (14) with various loss functions, we use stochastic gradient descent (SGD). In SGD, for each iteration, we pick an observed entry (i, j) at random, and only update the i^{th} row $\mathbf{w}_i^T$ of W and the j^{th} row $\mathbf{h}_j^T$ of H. The update rule for $\mathbf{w}_i^T$ and $\mathbf{h}_j^T$ is given by:

$$\mathbf{w}_i^T \leftarrow \mathbf{w}_i^T - \eta \left(\frac{\partial \ell oss\ (A_{ij}, (WH^T)_{ij})}{\partial \mathbf{w}_i^T} + \lambda \mathbf{w}_i^T \right),$$
$$\mathbf{h}_j^T \leftarrow \mathbf{h}_j^T - \eta \left(\frac{\partial \ell oss\ (A_{ij}, (WH^T)_{ij})}{\partial \mathbf{h}_j^T} + \lambda \mathbf{h}_j^T \right),$$

where η is a small step size. Each SGD update costs $O(k)$ time, and the total cost of sweeping through all the entries is $O(mk)$. Therefore, the time complexity for SGD is $O(t_2 mk)$, where t_2 is the number of iterations taken by SGD to converge. Notice that although the complexity of SGD is linear in k, it usually takes many more iterations to converge compared with ALS, that is, $t_2 > t_1$. For other approaches to solve (14) for large scale data, please see Yu et al. (2013).

On the other hand, all cycle-based algorithms introduced in Section 3 require time at least $O(nm)$, because they involve multiplication of m-sparse $n \times n$ matrices in model construction. In particular, the time complexity for HOC-ℓ methods is $O(2^\ell nm)$ (see Appendix B for details), which is much more expensive than both ALS and SGD as shown in Table 3 (note that in real large-scale social networks, $m > n \gg t_1, t_2, k$).

4.5 Clustering Signed Networks

In this section, we see how to take advantage of the low-rank structure of signed networks to find clusters. Based on weak balance theory, the general goal of clustering for signed graphs is to find a k-way partition such that most within-group edges are positive and most between-group edges are negative. One of the state-of-the-art algorithms for clustering signed networks, proposed by Kunegis et al. (2010), extends spectral clustering by using the signed Laplacian matrix. Given a partially observed signed network A, the signed Laplacian $\bar{L}$ is defined as $\bar{D} - A$, where $\bar{D}$ is a diagonal matrix such that $\bar{D}_{ii} = \sum_{j \neq i} |A_{ij}|$. By this definition, the clustering of signed networks can be derived by computing the top k eigenvectors of $\bar{L}$, say $U \in \mathbb{R}^{n \times k}$, and subsequently running the k-means algorithm on U to get the clusters. This procedure is analogous to the standard spectral clustering algorithm on unsigned graphs; the only difference being that the usual graph Laplacian is replaced by the signed Laplacian.

However, there is no theoretical guarantee that the use of the signed Laplacian can recover the true groups in a weakly-balanced signed network. To overcome this theoretical

Algorithm 2: Clustering with Matrix Completion

Input: Adjacency matrix A, number of clusters k

Output: Cluster indicators

1. $X^\star \leftarrow$ Completion(A) with any matrix completion algorithm.
2. $U \leftarrow$ Top k eigenvectors of $X^\star$.
3. Run any feature-based clustering algorithm on U.

defect, we now give an algorithm which, under certain conditions, is able to recover the real structure even with partial observations. The key idea is to first use a low-rank matrix completion algorithm before performing the clustering. The following theorem shows that the eigenvectors of the completed matrix possess a desirable property.

Theorem 19 *Let A_{ij}, $(i,j) \in \Omega$, be entries observed from a complete k-weakly balanced network $A^\star$ with n nodes, and assume that the solution of (9) is $X^\star$ with eigenvectors $U = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k]$. If the assumptions in Theorem 18 are all satisfied, then the rows of U, $U_{i,:} = U_{j,:}$ iff i and j are in the same cluster in $A^\star$with probability at least $1 - n^{-3}$.*

Proof From Theorem 18, we know the recovered matrix $X^\star$ will be $A^\star$ with probability $\geq 1 - n^{-3}$ if the assumptions hold. Suppose $\mathbf{u}_1, \ldots, \mathbf{u}_k$ are the k eigenvectors of $X^\star$. From the proof of Theorem 17, the eigenvectors will have the form in (11), which means that the i^{th} and j^{th} rows of U, $U_{i,:} = U_{j,:}$ if i and j are in the same cluster. Furthermore, when i and j are in different clusters, $A^\star_{i,:} \neq A^\star_{j,:}$, so $U_{i,:}$ cannot equal $U_{j,:}$. This proves the theorem. ■

Following this theorem, the true clusters can be identified from the eigenvectors of $X^\star$ when the assumptions in Theorem 18 hold. Therefore, perfect clustering is guaranteed in this scenario.

More generally, we can use any matrix completion method to complete A. For example, if we take SVP as the matrix completion approach, we can obtain a perfect clustering result if all iterates of the algorithm are ν-incoherent. Under the latter condition, SVP can recover $A^\star$ exactly, so the property of eigenvectors in Theorem 19 can again be used. Our clustering algorithm that uses matrix completion is summarized in Algorithm 2.

It should not be surprising that our clustering algorithm is superior to (signed) spectral clustering. In some sense, our approach can be viewed as a spectral method, except that it first fills in the missing links from the training data by doing matrix completion. This step is simple yet crucial in signed networks as it overcomes the sparsity of the network. We will see that our clustering algorithm outperforms the (signed) spectral clustering method in Section 5.

5. Experimental Results

We now present experimental results for sign prediction and clustering using our proposed methods. For sign prediction, we show that local methods, such as MOI and HOC (see Section 3), yield better predictive accuracy when longer cycles are considered. In addition, if we consider the global low-rank structure of the network, prediction via matrix factorization further outperforms local methods in terms of both accuracy and running time. For

clustering, we show that clustering via low rank model gives us significantly better results than clustering via signed Laplacian. These results suggest the usefulness of the global perspective on social balance.

5.1 Description of Data Sets

In our experiments, we consider both synthetic and real-life data sets. To construct synthetic networks, we first consider a complete k-weakly balanced network $A^\star$, and sample entries from $A^\star$ to form the partially observed network A, with three controlling parameters: *sparsity* s, *noise level* ϵ and *sampling process* $\mathcal{D}$. The sparsity s controls the percentage of edges we sample from $A^\star$. The noise level ϵ specifies the probability that the sign of a sampled edge is flipped. The sampling process $\mathcal{D}$ specifies how the sampled entries are distributed. In particular, we will focus on two sampling distributions: uniform and power-law distribution, denoted as $\mathcal{D}_{\text{uni}}$ and $\mathcal{D}_{\text{pow}}$ respectively. Thus, a partially observed network A can be described as $A = A^\star(s, \epsilon, \mathcal{D})$.

We also consider three real-life signed networks: Epinions, Slashdot, Wikipedia. Epinions is a consumer review network in which users can either trust or distrust other consumer's reviews. Slashdot is a discussion web site in which users can recognize others as friends or foes. Wikipedia is a who-votes-for-whom network in which users can vote for or against others to be administrators in Wikipedia. These three data sets have previously been used as benchmarks for sign prediction (Leskovec et al., 2010a; Chiang et al., 2011). Table 4 shows the statistics of the three networks.

	# nodes	# edges	+ edges	- edges	edges with conflicting signs
Wikipedia	7,065	103,561	78.8%	21.2%	0.71%
Slashdot	82,144	549,202	77.4%	22.6%	0.64%
Epinions	131,828	840,799	85.0%	15.0%	0.57%

Table 4: Network Statistics

5.2 Evidence of Local and Global Patterns in Real Signed Networks

We have seen that cycles in signed networks exhibit structural balance according to balance theory, and that we can make use of cycles for predictions (see Section 3). Indeed, cycles tend to be balanced in real-life networks. In all three real networks we consider, Leskovec et al. (2010b) found that balanced triads (i.e., 3-cycles) are much more likely to be observed than unbalanced triads. Our study also shows that the local patterns (i.e., ℓ-cycles) of the three networks tend to be balanced. For each network A, we consider all patterns of 3-cycles and 4-cycles in the symmetric network $\text{sign}(A + A^T)$. For convenience, we use $C_{\ell i}$ to denote the i^{th} pattern (of signs) of an ℓ-cycle. The patterns of these cycles are shown in Table 5. We first calculate the probability that the configuration of a given ℓ-cycle is $C_{\ell i}$, denoted $P(C_{\ell i})$. We then randomly shuffle the signs of edges in the network and calculate the same probability on the shuffled network, which is denoted $P_0(C_{\ell i})$. Thus $P_0(C_{\ell i})$ can be viewed as the (expected) probability that $C_{\ell i}$ is observed if the signs of edges have no particular

	Epinions			Slashdot			Wikipedia		
Type of cycle	$P(C_{\ell i})$	$P_0(C_{\ell i})$	$S(C_{\ell i})$	$P(C_{\ell i})$	$P_0(C_{\ell i})$	$S(C_{\ell i})$	$P(C_{\ell i})$	$P_0(C_{\ell i})$	$S(C_{\ell i})$
$C_{31}: +++$	0.8259	0.5754	1107.0	0.7301	0.4502	425.2	0.6996	0.4806	335.4
$C_{33}: +--$	0.0791	0.0706	72.3	0.1364	0.1260	23.5	0.0840	0.1105	-64.7
$C_{41}: ++++$	0.7538	0.4777	14464.7	0.6723	0.3435	5120.8	0.6080	0.3757	3557.6
$C_{43}: ++--$	0.0911	0.0787	1210.6	0.1127	0.1286	-352.1	0.1007	0.1155	-344.1
$C_{44}: +-+-$	0.0065	0.0393	-4418.5	0.0138	0.0645	-1528.0	0.0139	0.0578	-1396.4
$C_{46}: ----$	0.0103	0.0008	8722.8	0.0263	0.0030	3147.7	0.0054	0.0022	505.4
$C_{32}: ++-$	0.0834	0.3493	-1218.4	0.1125	0.4111	-458.7	0.2052	0.3987	-302.5
$C_{34}: ---$	0.0117	0.0047	220.9	0.0211	0.0127	56.9	0.0013	0.0102	8.5
$C_{42}: +++-$	0.1174	0.3875	-14508.8	0.1413	0.4211	-4191.5	0.2473	0.4167	-2548.5
$C_{45}: +---$	0.0208	0.0160	1017.7	0.0337	0.0392	-212.0	0.0247	0.0320	-309.3
Balanced 3-cycles	0.9050	0.6459	1182.9	0.8665	0.5763	443.9	0.7835	0.5911	299.6
Balanced 4-cycles	0.8617	0.5965	14147.8	0.8250	0.5397	4234.7	0.7280	0.5513	2635.6

Table 5: Statistics of balanced and unbalanced ℓ-cycles, $\ell = 3, 4$ (note that $\sum_i P(C_{\ell i}) = \sum_i P_0(C_{\ell i}) = 1$). The first 6 cycles in the table are balanced while the last 4 cycles are unbalanced. The last two rows show that overall balanced 3-cycles and 4-cycles are much more than expected.

pattern. With the two probabilities, we calculate the "surprise" of observing $C_{\ell i}$ as follows:

$$S(C_{\ell i}) := \frac{\Delta_\ell P(C_{\ell i}) - \Delta_\ell P_0(C_{\ell i})}{\sqrt{\Delta_\ell P_0(C_{\ell i})(1 - P_0(C_{\ell i}))}},$$

where Δ_ℓ is the number of ℓ-cycles in the network. The above quantity is basically the number of standard deviations that the observed value of $C_{\ell i}$ differs from the expected value of $C_{\ell i}$ in the shuffled network. See Leskovec et al. (2010b) for more discussion.

Table 5 shows the observed probability, the expected probability, and the surprise value of each $C_{\ell i}$ in three networks. Although it is not true that *each* of the balanced cycles is much more likely to appear, the last two rows in Table 5 show that differences between $P(C)$ and $P_0(C)$ and the surprise values of *overall* balanced 3- and 4-cycles are quite large. This implies that given an arbitrary 3- or 4-cycle in these networks, it is very likely to be balanced. Overall, we find that local balanced patterns are somewhat significant.

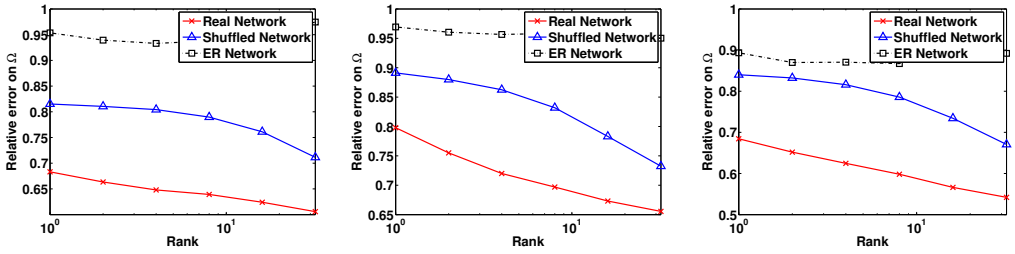
On the other hand, in Section 4, we have seen that low rank structure emerges when we theoretically examine weakly balanced networks. We now show that real networks tend to exhibit low-rank structure to a much greater extent compared to random networks. As a baseline, for each real network we create two corresponding random networks for comparison: the first one is the (symmetric) ER network generated from the Erdös-Rényi model (Erdös and Rényi, 1960) that preserves the sparsity and the ratio of positive to negative edges of the compared real network. The second one is the shuffled network with the same network structure as the compared real network, except that we randomly shuffle the signs of edges.

The experiment is conducted as follows. We first derive the low-rank complete matrix $X^\star$ by running matrix completion on the observed entries A_{ij}. Then, we look at the relative error on the observed set Ω:

$$\text{err}_\Omega = \frac{\|W \circ (X^\star - A)\|_F}{\|A\|_F},$$

where $W_{ij} = 1$ if $(i, j) \in \Omega$ and $W_{ij} = 0$ otherwise, and $\circ$ denotes element-wise multiplication. Clearly, smaller err_Ω indicates better approximation for the observed entries.

In our experiment, we use matrix factorization for matrix completion, with ranks $k = 1, 2, 4, 8, 16$ and 32. For each network (real networks and their corresponding random networks), we complete the network with different k values and compute err_Ω. The result is shown in Figure 2. Compared to the two random networks, the three real-life networks achieve much smaller err_Ω for each small k. This suggests that low-rank matrices provide a better approximation of the observed entries for real-life networks.



(a) Epinions (b) Slashdot (c) Wikipedia

Figure 2: Relative error on Ω, the observed entries, between adjacency matrix and completed matrix, for real-life networks versus random networks. Real-life networks achieve much smaller relative error for every k as compared with random networks.

5.3 Sign Prediction

We now compare the performance of our proposed methods for sign prediction. As introduced in Section 3, there are two families of cycle-based methods: one based on measures of imbalance (MOI), and the other based on the supervised learning using higher order cycles (HOC). Both families depend on a parameter $\ell \geq 3$ that denotes the order of the cycles that the method is based on. For MOI, we consider all ℓ less than 10 as well as ∞ (recall that in this case, MOI becomes the Katz measure), and for HOC we consider $\ell = 3, 4, 5$. Note that the set of features used by HOC-$(\ell + 1)$ is a strict superset of the features used by HOC-ℓ.

We also consider two global approaches for low rank matrix completion—Singular Value Projection and matrix factorization from Sections 4.2 and 4.3 respectively. The SVP approach (denoted as LR-SVP) is chosen to demonstrate that perfect recovery can be achieved if the observations are uniformly distributed. For matrix factorization, we consider the ALS method that solves problem (13), as well as SGD methods that solve the general problem (14) with sigmoid loss and square-hinge loss, defined in (15). We denote these methods as LR-ALS, LR-SIG and LR-SH, respectively.

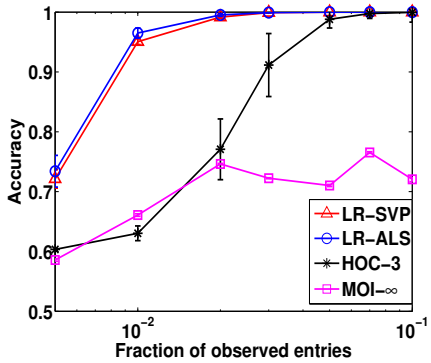
5.3.1 Results on Synthetic Data Sets

We first compare all categories of approaches on synthetic data sets. We choose LR-SVP, LR-ALS, MOI-∞ and HOC-3 as representatives of the two approaches of low rank matrix completion, MOI-based, and HOC-based methods respectively. We consider the underlying network $A^\star$ to be a complete 5-weakly balanced network, where the five clusters have sizes 100, 200, 300, 400 and 500. Instead of observing all of $A^\star$, we assume that we only observe a partial network A using three procedures: uniform sampling, uniform sampling with noise, and sampling with power-law distribution. For each algorithm, we input the observed entries as training data and calculate the sign prediction accuracy on the rest of the entries.

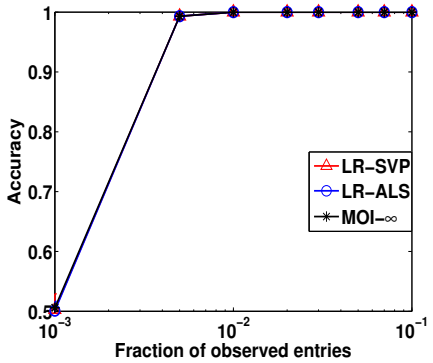
Uniform sampling: In this scenario, we generate several observed networks $A = A^\star(s, 0, \mathcal{D}_{\text{uni}})$. We vary s from 0.001 to 0.1 and plot the prediction accuracy in Figure 3a. Under this setting, LR-SVP and LR-ALS outperform the cycle-based methods. We observe that MOI-∞ performs the worst with accuracy only 50%-70%. However, if we repeat the same experiment substituting $A^\star$ with $A_2^\star$, where $A_2^\star$ is a complete strongly balanced network whose two groups have size 1000, we observe that MOI and global methods perform alike as shown in Figure 3b. This is because MOI uses cycle-based measurements to make more cycles become *balanced*. This prediction policy is most appropriate when $k = 2$ (that is, the underlying network $A^\star$ has strong balance), but performs poorly when the underlying network is weakly balanced (i.e., more than two groups). HOC-3 works much better than MOI-∞ since it learns a classifier from cycle-based features rather than simply making cycles balanced, but its accuracy drops dramatically when s is less than 0.05. On the other hand, both LR-SVP and LR-ALS show high accuracy for all $s \geq 0.01$. In particular, LR-SVP can achieve 100% accuracy when $s > 0.07$, which reconfirms the theoretical recovery guarantee stated in Theorem 18. Moreover, LR-ALS can also recover the ground truth, an observation that is consistent with previous results.

Uniform sampling with noise: To make the synthetic data more similar to real data, we further add noise to the observations. We generate observed networks $A = A^\star(0.1, \epsilon, \mathcal{D}_{\text{uni}})$, where ϵ varies from 0.01 to 0.25. The result is shown in Figure 3c. We can see that global methods are still clearly better than cycle-based methods when noise level is higher. Moreover, LR-SVP perfectly recovers $A^\star$ when the noise level $\epsilon < 0.05$, and LR-ALS also achieves perfect recovery with a smaller ϵ.

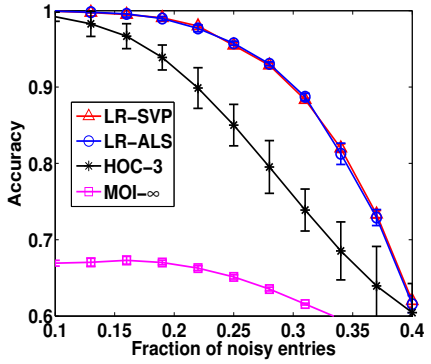
Sampling with power-law distribution: As Sections 4.1 and 4.2 pointed out, the exact recovery guarantees of convex relaxation and SVP for matrix completion crucially rely on the assumption that observed entries are uniformly sampled. However, in most real networks (for example, Slashdot in Kunegis et al. 2009), the degree distribution of observed entries follows a power law. Therefore, we examine how the approaches perform on power-law distributed networks. The power-law distributed networks are generated using the Chung-Lu-Vu (CLV) model proposed by Chung et al. (2004), which allows one to generate random graphs with arbitrary expected degree sequence. Similar to the uniform sampling case, we perform the sign prediction task on $A = A^\star(s, 0, \mathcal{D}_{\text{pow}})$ varying s from 0.001 to 0.1, and plot the prediction accuracy in Figure 3d. We can see that MOI-∞ still has poor performance for weakly balanced graphs. However, unlike the uniform sampling case, LR-SVP has lower accuracy rate compared to HOC-3 when $s < 0.1$. On the other hand, LR-ALS still performs better than all other methods on power-law distributed graphs.



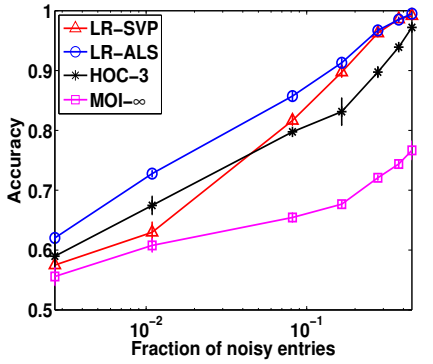
(a) Uniformly sampled without noise ($k = 5$)



(b) Uniformly sampled without noise on balanced networks ($k = 2$)



(c) Uniformly sampled with noise ($k = 5$)



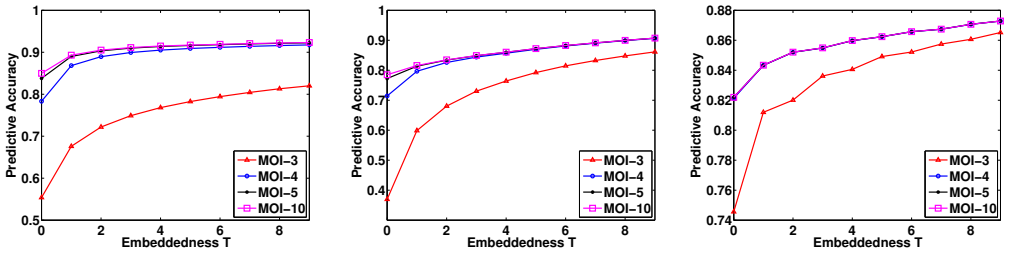
(d) Power-law distributed networks ($k = 5$)

Figure 3: Sign prediction accuracy of local and global methods on synthetic data sets. On (strongly) balanced networks (3b), MOI-∞ is seen to perform as well as low rank modeling methods (LR-SVP and LR-ALS). However, in weakly balanced networks, global methods LR-SVP and LR-ALS outperform cycle-based methods such as MOI-∞ and HOC-3 (supervision on high order cycles). In addition, low rank modeling with matrix factorization (LR-ALS) is more robust than singular value projection (LR-SVP) when the observations are sampled from a power-law distribution.

From results on synthetic data shown in Figure 3, we can conclude that global methods generally do better than local methods, and the low rank model with matrix factorization (LR-ALS) performs the best in most cases, even when observed entries are not uniformly distributed.

5.3.2 Results on Real-life Data Sets

Now we further evaluate our sign prediction methods on three real-life networks. To begin with, we evaluate and compare MOI methods using a *leave-one-out* type methodology:



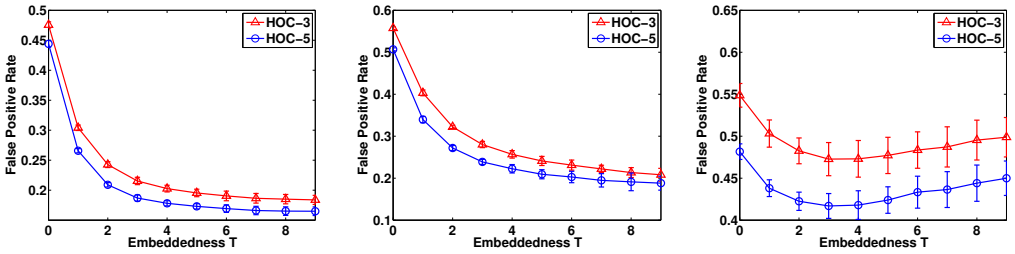
(a) Epinions (b) Slashdot (c) Wikipedia

Figure 4: Accuracy of MOI-based methods for cycle lengths $\ell = 3, 4, 5, 10$. These plots show the accuracy of MOI-ℓ methods for edges with embeddedness *at least* T for various thresholds T. We see that the difference in the performance of MOI-3 and higher order methods is larger for edges with lower embeddedness. We also see that the improvement obtained by going beyond order 5 is not very significant.

each edge in the network is successively removed and the method tries to predict the sign of that edge using the rest of the network. Figure 4 shows the accuracy of MOI based methods. Note that the accuracy is shown for edges with embeddedness under a certain threshold. First, we see that the accuracy is a non-decreasing function of the embeddedness threshold. Next, it is clear that higher-order methods perform significantly better than the MOI-3 (triangle-based) method. Finally, the performance boost is larger for edges with low embeddedness. This is expected as edges of low embeddedness by definition do not have many common neighbors for their end-points, and higher-order cycles have relatively better information for such edges. We also observe from our experiments that beyond $\ell = 5$, the performance gain is not very significant.

Next, we compare HOC methods with various ℓ to see how much high order cycles can benefit us in supervision. We resort to 10-*fold cross-validation*: we (randomly) created ten disjoint test folds each consisting of 10% of the total number of edges in the network. For each test fold, the remaining 90% of the edges serve as the training set. (For a given test fold, the feature extraction and logistic model training are done on a graph with the test edges removed, not just the signs.) To evaluate HOC methods, we consider not only prediction accuracies but also false-positive rates. We report accuracies as well as false-positive rates by averaging them over the ten folds. As shown in Table 6, in all the data sets, there is a small improvement in accuracy by using higher order cycles (HOC-5). The false positive rate, however, reveals a more interesting phenomenon in Figure 5. Indeed, higher order methods (such as HOC-5) significantly reduce the false positive rate as compared to HOC-3. However Figure 5 shows that, unlike MOI based methods, edge embeddedness does not seem to affect the decrease in false positive rate for HOC methods. We see this trend across all the data sets.

At this point, we see that for cycle-based methods, considering higher order cycles benefits the accuracy of sign prediction and lowers the false positive rate. Furthermore, the results are consistent across the three diverse networks. These results confirm the intuition



(a) Epinions (b) Slashdot (c) Wikipedia

Figure 5: False positive rates of higher order cycle (HOC) Methods for $\ell = 3, 5$. These plots show the false positive rate of HOC-ℓ methods for edges with embeddedness *at least* T for various thresholds T. We see that considering higher order cycles has the benefit of significantly reducing false-positives while simultaneously achieving slightly better overall accuracy (refer to Table 6). However, unlike what we see for MOI methods, the improvement here does not seem to depend strongly on edge embeddedness. The false positive rates for HOC-4 are very similar to that of HOC-5 and hence are not shown for clarity.

that getting more global information improves quality of prediction, and motivate us to consider the global structure of networks.

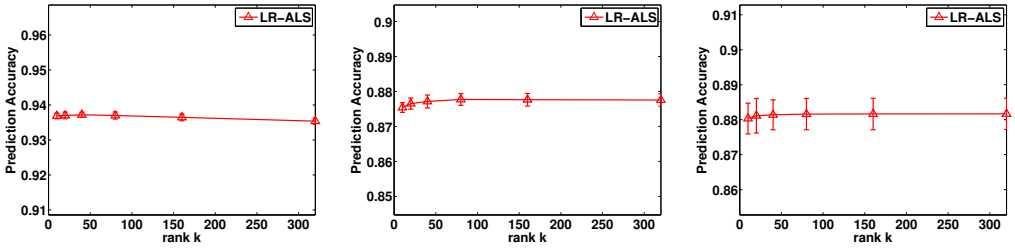
Now we turn our attention to low rank modeling approaches. We have seen that LR-SVP does not perform well under power-law distributions of observed relationships in synthetic networks (see Figure 3d), so we consider the more robust matrix factorization approach for solving the matrix completion problem, including LR-ALS, LR-SIG and LR-SH, for experiments on real data sets. Again, we use 10-fold cross validation setting, and report the average prediction accuracy for each data set in Table 6. From the table, we observe that global methods clearly outperform cycle-based methods. In particular, we observe that HOC-5 only improves HOC-3 by less than 1.5%, while global methods consistently improve the accuracy of HOC-5 by more than 2% over all data sets. In addition, LR-SIG and LR-SH further improve the accuracy of LR-ALS. This shows that the sigmoid and square-hinge losses are more suitable for sign prediction, which supports the discussion in Section 4.3. On real data sets, we do not have prior information about the target rank k. However, Figure 6 shows that the performance of LR-based methods is not sensitive to k.

In Figure 7, we further select a representative of each category, MOI-10, HOC-5 and LR-ALS, and show their performances with different levels of edge embeddedness (LR-SIG and LR-SH perform similar to LR-ALS in all data sets). In addition, we also compare our methods with the methods A_sym_exp and A_exp proposed by Kunegis et al. (2009), which predicts the sign of edges using matrix exponential with low rank approximation.[3][4] For LR-ALS, A_sym_exp and A_exp we choose the rank $k = 40$.

3. The method A_sym_exp considers the symmetric matrix A_sym $= A + A^T$ and its eigen-decomposition $U\Sigma U^T$, and computes the matrix exponential of A_sym with rank-k approximation, $U_k \exp(\Sigma_k) U_k^T$.
4. They incorrectly refer to A_exp as the exponential of A as they in fact compute A_exp as $U_k \exp(\Sigma_k) V_k^T$, where $U_k \Sigma_k V_k^T$ is the best rank-k approximation of A.

	Epinions	Slashdot	Wikipedia
MOI-3	0.5539	0.3697	0.7456
MOI-10	0.8497	0.7850	0.8220
HOC-3	0.9014 ± 0.0013	0.8303 ± 0.0018	0.8424 ± 0.0063
HOC-5	0.9080 ± 0.0012	0.8469 ± 0.0015	0.8605 ± 0.0049
LR-ALS	0.9437 ± 0.0007	0.8774 ± 0.0018	0.8814 ± 0.0043
LR-SIG	$\mathbf{0.9448} \pm 0.0009$	0.8806 ± 0.0017	$\mathbf{0.8839} \pm 0.0049$
LR-SH	0.9437 ± 0.0015	$\mathbf{0.8835} \pm 0.0015$	0.8810 ± 0.0042

Table 6: The sign prediction accuracy for cycle-based methods (MOI and HOC) and low rank modeling methods (LR-ALS, LR-SIG and LR-SH) along with standard deviation if the accuracy is averaged by 10-fold cross validation. Note that for MOI methods we report leave-one-out accuracy. We can see that the low rank modeling approaches are better than cycle-based methods.



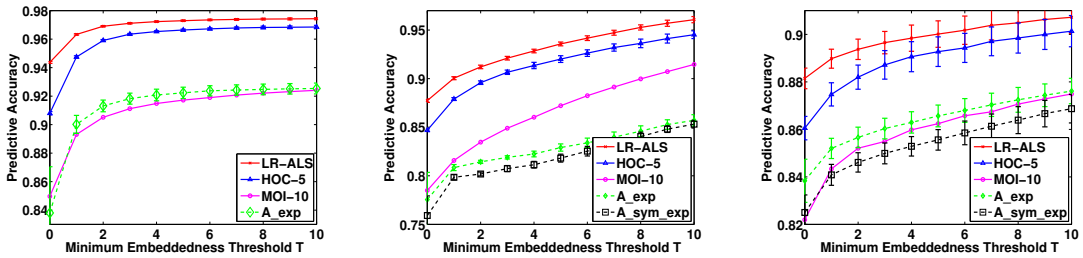
(a) Epinions (b) Slashdot (c) Wikipedia

Figure 6: Sign prediction accuracy for low rank modeling with matrix factorization (LR-ALS) with different ranks. We see that LR-ALS is quite robust to the rank.

From Figure 7 we see that matrix exponential and MOI methods perform alike as one would expect. HOC learns the weights carefully to determine which configurations of cycles are more important, and therefore performs better than the former two methods that use fixed weights. Also, one might expect that cycle-based approaches should perform better on edges with higher embeddedness because of the relatively richer cycle information available. However, LR-ALS achieves higher prediction accuracy regardless of the embeddedness. All the above results show that global methods are more effective than local methods.

5.3.3 Running Time Comparison

As discussed in Section 4.3, low rank modeling with matrix factorization is more efficient than cycle-based algorithms in terms of time complexity, which we now confirm. The running times are summarized in Table 7. To show the scalability of matrix factorization methods, we construct a large-scale data set Cluster10, which contains 1.1 million nodes and 120 million edges (about 100 times more than Epinions). Cluster10 is constructed by uniformly sampling edges from a 10-weakly balanced network, in which clusters have sizes



(a) Epinions (b) Slashdot (c) Wikipedia

Figure 7: Sign prediction accuracy of various methods with different levels of embeddedness, along with standard deviation if the accuracy is averaged by 10-fold cross validation. For MOI methods we report leave-one-out accuracy. These plots show the accuracy for edges with embeddedness at least T. The A_sym_exp method in Epinions cannot achieve accuracy 80% for all embeddedness levels so it is not shown in the plot. We can see that LR-ALS consistently achieves the highest accuracy for all thresholds T.

	HOC-3	HOC-4	HOC-5	LR-ALS	LR-SGD
Wikipedia	18.08	74.52	462.92	**2.26**	2.41
Slashdot	133.4	1,936.0	$> 10,000$	**17.4**	24.7
Epinions	560.64	6,156.8	$> 10,000$	**28.67**	37.2
Cluster10	$> 10,000$	$> 10,000$	$> 10,000$	**455.1**	1,152

Table 7: Running time (in seconds) for low rank modeling methods (LR-ALS and LR-SGD) and supervision on high order cycles (HOC) on real data sets and a 1.1 million node synthetic data Cluster10. We see that LR methods with matrix factorization are clearly more efficient than cycle-based algorithms.

20000, 40000, ..., 200000. For matrix factorization approach, we report the time needed to solve the model by ALS (with square loss) and SGD (with sigmoid and square-hinge losses). The time LR-SGD is thus the average time to solve LR-SIG and LR-SH models. For HOC methods the training time is dominated by the feature construction step, thus we only report the time for computation of features. Therefore, the reported time for HOC is an underestimate of the time required to construct the HOC model; even then we can see that the time required by LR-ALS, LR-SIG and LR-SH is much lower than HOC methods.

In conclusion, for the sign prediction problem, the low rank model with matrix factorization is clearly the best method in terms of accuracy and scalability.

5.4 Clustering

We now show that our clustering approach, which completes the low-rank structure of signed networks before performing clustering, outperforms spectral clustering based on the signed Laplacian (Kunegis et al., 2010). We conduct experiments on synthetic data generated

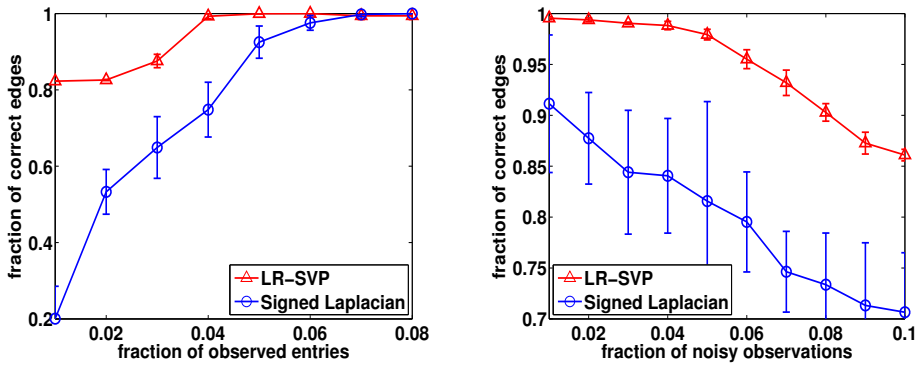
from weakly balanced networks (note that we do not have ground truth for clustering in the real-life data sets). We consider a 10-weakly balanced network $A^\star$ where size of each group is 100, and sample entries from $A^\star$ using uniform sampling and uniform sampling with noise.

To measure the performance of clustering, we calculate the number of edges that satisfy the ground-truth clustering, which is defined by

$$\sum_{i,j:s_i=s_j} I(\bar{s}_i = \bar{s}_j) + \sum_{i,j:s_i\neq s_j} I(\bar{s}_i \neq \bar{s}_j). \tag{16}$$

where $s_1, \ldots, s_n$ denote the ground-truth clustering assignment for each node, and $\bar{s}_1, \ldots, \bar{s}_n$ are the clustering results given by the clustering algorithm.

Following the procedure outlined in Section 5.3, in the uniform sampling case, we consider the networks $A = A^\star(s, 0, \mathcal{D}_{\text{uni}})$ with $s \in [0.01, 0.06]$, while in sampling with noise case we consider networks $A = A^\star(0.05, \epsilon, \mathcal{D}_{\text{uni}})$ with $\epsilon \in [0.01, 0.08]$. For each observed network, we apply Algorithm 2 (see Section 4.5) and clustering via the signed Laplacian, and evaluate clustering results using (16). The results of these two scenarios are shown in Figure 8. In both the scenarios, our proposed clustering approach is significantly better than clustering based on the signed Laplacian, and shows that recovering the low-rank structure of signed networks leads to improved clustering results.



(a) Data without noise (b) Data with noise

Figure 8: Clustering partially observed synthetic data - clustering with matrix completion using SVP (LR-SVP) performs significantly better than clustering with the signed Laplacian.

6. Related Work

Signed networks have been studied since the early 1950s. Harary and Cartwright were the first to mathematically study structural balance. They defined balanced triads and proved the global structure of balanced signed networks (Harary, 1953; Cartwright and Harary, 1956). Davis (1967) further generalized the balance notion to weak balance by allowing

triads with all negative edges, and showed that weakly balanced graphs have the global structure of mutual antagonistic groups.

Though theoretical studies of signed networks have been conducted for a long time, it was not until this decade that analysis of real signed networks could be done at a large scale as large real networks have become more accessible recently. For example, Kunegis et al. (2009) performed several analysis tasks on Slashdot, and Leskovec et al. (2010a,b) studied the local and global structure of three real signed networks. They designed several computational experiments to justify that the structure of these signed networks match balance theory.

In this paper, we focused on problems in signed networks. However, these problems have their counterparts in unsigned networks. For instance, structural link prediction in unsigned networks corresponds to the sign prediction problem. Structural link prediction has been well explored, and it is usually solved by computing a similarity measure between nodes (Liben-Nowell and Kleinberg, 2007), such as those proposed by Katz (1953) and Adamic and Adar (2003). The sign prediction problem, however, was not formally considered until the work by Guha et al. (2004), in which they develop a trust propagation framework to predict trust or distrust between entities. More recently, Kunegis et al. (2009, 2010) reconsidered this problem by using various similarity functions and kernels such as matrix exponential and signed Laplacian. Leskovec et al. (2010a) proposed a machine learning formulation of this problem, arguing that learning from only local triangular structure of edges can achieve reasonable accuracy.

Sign prediction using our global method is closely related to the low-rank matrix completion problem. In the last five years, there has been substantial research studying exact recovery conditions for this problem (Candés and Recht, 2008; Candés and Tao, 2009), and algorithms with theoretical guarantees have also been proposed, such as SVT (Cai et al., 2010) and SVP (Jain et al., 2010). While the matrix completion problem has been considered mostly in collaborative filtering, our low rank model arises naturally from the weak balance of signed networks.

Clustering is another fundamental problem in network analysis. For unsigned networks, there are several proposed algorithms that have been shown to be effective, such as clustering via graph Laplacian (Ng et al., 2001), modularity (Newman, 2006) and multilevel approaches (Dhillon et al., 2007). However, most of these approaches cannot be directly extended to signed networks since weak balance theory does not apply to unsigned networks. As a result, researchers have tried to tailor unsigned network clustering algorithms in order to make them applicable to signed networks. For instance, Doreian and Mrvar (1996) proposed a local search strategy which is similar to the Kernighan-Lin algorithm (Kernighan and Lin, 1970). Starting with an initial clustering assignment, it tries to move nodes one by one to get a more preferable clustering. Yang et al. (2007) proposed an agent-based method which basically conducts a random walk on the graph. Kunegis et al. (2010) generalized spectral algorithms to signed networks. They proposed a spectral approach using the signed Laplacian, and showed that partitioning signed networks into two groups using the signed Laplacian kernel is analogous to considering ratio cut on unsigned networks. Anchuri and Magdon-Ismail (2012) proposed hierarchical iterative methods that solve 2-way signed modularity objectives using spectral relaxation at each hierarchy. Chiang et al. (2012) proposed graph kernels for

signed network clustering, and showed that the multilevel framework can be extended to this problem.

Another line of research related to signed graph clustering problem is correlation clustering. The goal of correlation clustering is the following: given n objects where certain pairs of objects are labelled as similar and certain pairs as dissimilar, find a clustering that maximizes the number of similar pairs within clusters, plus the number of dissimilar pairs between clusters. The problem was first considered by Bansal et al. (2004), who proved that finding the optimal correlation clustering is NP-hard, and proposed two approximation algorithms to maximize the number of edges that satisfy "agreement" (a positive within-cluster edge or a negative between cluster-edge) and to minimize the number of edges that do not, under the special case that all pairwise label information is given. Bounds for general correlation clustering setting have been obtained by Demaine et al. (2006). On the other hand, some researchers have also considered the correlation clustering problem from the statistical learning theory viewpoint. For example, Joachims and Hopcroft (2005) give error bounds for the problem if only partial pairs are observed. Recently, Cesa-Bianchi et al. (2012) proposed a method for sign prediction by learning a correlation clustering index. They consider three types of learning models: batch, online and active learning, and provide theoretical bounds for prediction mistakes under each setting. Though there is no social balance notion in the correlation clustering problem, it can be viewed as finding a clustering of signed graph where nodes correspond to objects, and positive/negative edges correspond to similar/dissimilar pairs. Therefore, our proposed method can also be applied to the problem of correlation clustering.

7. Conclusions and Future Work

In this paper, we studied the usefulness of social balance in signed networks, with two fundamental applications: sign prediction and clustering. Starting from a local view of social balance, we developed sign prediction methods based on length-ℓ cycles. The predictive accuracies are improved if longer cycles are taken into consideration, suggesting that a broader view of local patterns helps in sign prediction. We then considered the global perspective on social balance, and showed that the adjacency matrices of (weakly) balanced networks are low rank. Based on this observation, we modeled the sign prediction problem as a low-rank matrix completion problem. We discussed three approaches to matrix completion: convex relaxation, singular value projection, and matrix factorization. In addition, we applied this low rank modeling technique to the clustering problem. In experiments, we observe that sign prediction via matrix factorization not only outperforms local methods (MOI and HOC), but requires much less running time. Clustering results are also more favorable via the matrix completion approach in comparison with the existing signed Laplacian approach. All of these results consistently demonstrate the effectiveness of the global viewpoint of social balance.

For future work, one possible direction is to explore analysis tasks for heterogeneous signed networks. Since there are different types of entities in heterogeneous networks, currently there are no clear answers to questions such as: do balance relationships exist on such networks? How do we quantitatively measure balance if balance patterns exist? How is balance at a local level related to the global structure? Furthermore, another possible

direction is to examine other theories for directed signed networks. Leskovec et al. (2010a,b) has found evidence that status theory holds in real signed networks, but that the patterns conforming to status theory are quite different from those conforming to balance theory. Thus, it is natural to ask how to design algorithms by pursuing global patterns conforming to status theory. These interesting directions are worth exploring in future research.

Acknowledgments

We gratefully acknowledge the support of NSF grants CCF-0916309, CCF-1117055, and DOD Army grant W911NF-10-1-0529. Most of the contribution of Ambuj Tewari to this work occurred while he was a postdoctoral fellow at the University of Texas at Austin.

Appendix A. Proofs

Proof of Theorem 10 One direction is trivial. If A is unbalanced then there is an unbalanced simple cycle. However, any simple cycle is obviously a cycle and hence the sum in (6) will be strictly positive.

For the other direction, suppose $\mu_\infty(A) > 0$. This implies there is an unbalanced cycle σ in the graph. Decompose the unbalanced cycle into finitely many simple cycles. We will be done if we could show that one of these simple cycles has to be unbalanced. It is easy to see why this is true: if all of these simple cycles were balanced, they all would have had an even number of negative edges, but then the total number of negative edges in σ could not have been odd. ■

Proof of Lemma 11 Define the sets of ℓ-cycles,

$$C_\ell^+(i,j) := \{\sigma \in C_\ell\Big(A^{+(i,j)}\Big) : \sigma \text{ includes } (i,j)\}$$
$$C_\ell^-(i,j) := \{\sigma \in C_\ell\Big(A^{-(i,j)}\Big) : \sigma \text{ includes } (i,j)\} \ ,$$

that include the edge (i,j). Note that, since $A^{+(i,j)}$ and $A^{-(i,j)}$ only differ in the sign of the edge (i,j), we have,

$$C_\ell\Big(A^{+(i,j)}\Big) \backslash C_\ell^+(i,j) = C_\ell\Big(A^{-(i,j)}\Big) \backslash C_\ell^-(i,j) \ .$$

Thus, we have,

$$\begin{aligned}
&\sum_{\sigma \in C_\ell\left(A^{-(i,j)}\right)} \mathbf{1}\left[\sigma\right] - \sum_{\sigma \in C_\ell\left(A^{+(i,j)}\right)} \mathbf{1}\left[\sigma\right] \\
&= \sum_{\sigma \in C_\ell^-(i,j)} \mathbf{1}\left[\sigma\right] + \sum_{\sigma \in C_\ell\left(A^{-(i,j)}\right) \backslash C_\ell^-(i,j)} \mathbf{1}\left[\sigma\right] - \sum_{\sigma \in C_\ell^+(i,j)} \mathbf{1}\left[\sigma\right] - \sum_{\sigma \in C_\ell\left(A^{+(i,j)}\right) \backslash C_\ell^+(i,j)} \mathbf{1}\left[\sigma\right] \\
&= \sum_{\sigma \in C_\ell^-(i,j)} \mathbf{1}\left[\sigma\right] - \sum_{\sigma \in C_\ell^+(i,j)} \mathbf{1}\left[\sigma\right] \ . \qquad (17)
\end{aligned}$$

Now cycles in $C_\ell^-(i,j)$ are in 1-1 correspondence with paths π in $\mathcal{P}_{\ell-1}(i,j)$ of length $\ell-1$, in the original graph A, that go from i to j. Moreover, $\sigma \in C_\ell^-(i,j)$ is unbalanced iff

the corresponding path in $\mathcal{P}_{\ell-1}(i,j)$ has an *even* number of -1's. Similarly, $\sigma \in C_\ell^+(i,j)$ is unbalanced iff the corresponding path in $\mathcal{P}_{\ell-1}(i,j)$ has an *odd* number of -1's. Thus, continuing from (17):

$$\begin{aligned}
&\sum_{\sigma \in C_\ell^-(i,j)} \mathbf{1}[\sigma] - \sum_{\sigma \in C_\ell^+(i,j)} \mathbf{1}[\sigma] \\
&= \sum_{\pi \in \mathcal{P}_{\ell-1}(i,j)} \mathbf{1}[\pi \text{ has even no. of } -1\text{'s}] - \sum_{\pi \in \mathcal{P}_{\ell-1}(i,j)} \mathbf{1}[\pi \text{ has odd no. of } -1\text{'s}] \\
&= \sum_{i_1, i_2, \ldots, i_{\ell-2}} A_{i,i_1} \cdot A_{i_1,i_2} \cdot \ldots \cdot A_{i_{\ell-2},j} = \left(A^{\ell-1}\right)_{i,j} ,
\end{aligned}$$

where the second equality is true because A only has $\pm 1, 0$ entries. ■

Appendix B. Supervised Higher-Order Cycle (HOC) methods

We begin by describing the approach used by Leskovec et al. (2010a). The features for an edge are based on the sign configurations of triads it is a part of. Fix an edge $e = (i,j)$. Consider an arbitrary common neighbor (in an undirected sense) k of i and j. Links between i and k can be in 4 possible configurations:

$$i \xrightarrow{+} k \qquad i \xleftarrow{+} k$$
$$i \xrightarrow{-} k \qquad i \xleftarrow{-} k \ .$$

Similarly, there are 4 possible configurations for links between k and j. Thus, we can get a total of 16 features for the edge e by considering the number of common neighbors k in each of the $4 \times 4 = 16$ configurations. Though we draw 4 configurations separately, links with different directions can simultaneously exist between i and j, possibly with different sign.

These configurations corresponds to features for a supervised variant of the k-cycle method for $k = 3$. Let A^+ and A^- be the matrices of positive and negative edges such that $A = A^+ + A^-$. In terms of matrix powers, these sixteen features are nothing but the (i,j) entries in the sixteen matrices:

$$A^{b_1} \cdot A^{b_2} \qquad A^{b_1} \cdot \left(A^{b_2}\right)^T \qquad \left(A^{b_1}\right)^T \cdot A^{b_2} \qquad \left(A^{b_1}\right)^T \cdot \left(A^{b_2}\right)^T , \tag{18}$$

where $b_1, b_2 \in \{\pm\}$, and $(A^{b_1})^T$ denotes the transpose of A^{b_1}. Note that we have described the features of a *directed* edge $e = (i,j)$.

B.1 Using Higher-order Cycles

A criticism against using only these triangle-based features is that there could be many people in the social network who do not share friends. In fact, this is the case in most of the networks used by Leskovec et al. (2010a). The reason their method is able to predict well on such pairs is that they additionally use seven other "degree-type" features like in-degree

and out-degree (and their signed variants). Thus, the prediction for an edge with zero emdeddedness (embeddedness refers to the number of common neighbors of the vertices of an edge) relies completely on the degree-based features. We can additionally incorporate features from higher-order cycles. Generalizing the construction (18), we can define 64 fourth-order features (corresponding to 4-cycles in the graph) of an edge (i, j) as the (i, j) entries in the matrices:

$$\left(A^{b_1}\right)^{t_1} \cdot \left(A^{b_2}\right)^{t_2} \cdot \left(A^{b_3}\right)^{t_3} ,$$

where $b_i \in \{\pm\}$ indicates whether we look at the positive or negative part of A and $t_i \in \{T, 1\}$ indicates whether or not we transpose it. There are 4 possibilities for each b_i, t_i pair, resulting in a total of $4 \times 4 \times 4 = 64$ possibilities.

By now the reader can guess the construction of features of a general order $\ell \geq 3$. For the edge (i, j), they will be the (i, j) entries in the $4^{\ell-1}$ matrices

$$\left(A^{b_1}\right)^{t_1} \cdot \left(A^{b_2}\right)^{t_2} \cdots \cdot \left(A^{b_{\ell-1}}\right)^{t_{\ell-1}} ,$$

with $b_i \in \{\pm\}, t_i \in \{T, 1\}$.

Note that the number of features is exponential in ℓ, and therefore it is not feasible to obtain features from arbitrarily long cycles. We use $\ell \leq 5$ for supervised HOC methods in our experiments that are presented in Section 5.

B.2 Reducing the Number of Features

The number of features can quickly become unmanageable, and computationally infeasible, as soon as ℓ is beyond 5. While dimensionality of the feature space may be the primary concern, the combinatorial nature of the features also raises the following intuitive concern: the interpretability of features rendered by high-order cycles, say when $\ell = 6$, composed of different signs and directions, is a challenge. For example, it is intuitively hard to appreciate the difference between the two walks $i \xrightarrow{+} k_1 \xrightarrow{+} k_2 \xrightarrow{-} k_3 \xrightarrow{+} k_4 \xrightarrow{+} j$ and $i \xrightarrow{+} k_1 \xrightarrow{+} k_2 \xleftarrow{-} k_3 \xrightarrow{+} k_4 \xrightarrow{+} j$.

With this realization, one way to quickly reduce the number of features, yet retain the information in longer cycles, is to consider the underlying undirected graph, ignoring the directions. In particular, the ℓ^{th} order features will be from the matrices

$$A^{b_1} \cdot A^{b_2} \cdots \cdot A^{b_{k-1}},$$

with $b_i \in \{\pm\}$. Note that since we are considering the undirected graph, we ensure that the features are symmetric by summing features of the form $A^{b_1}A^{b_2}$ and $A^{b_2}A^{b_1}$. Thus the number of ℓ^{th} order features to compute is reduced to $O(2^\ell)$ from $O(4^\ell)$. Though the number of features is still exponential in ℓ, the construction of features becomes easier for small values of ℓ.

We note that another way to avoid dealing with too many features is to use a *kernel* instead. A kernel computes inner products in feature space without explicitly constructing the feature map. One can then use off-the-shelf SVM classifiers to perform the classification. We leave this promising approach of directly defining a kernel on *pairs of nodes* of a graph and using it for link prediction to future work.

B.3 Classifier

We use a simple logistic regression where the imbalance of an edge is modeled as a linear combination of the features, which are imbalances in cycles of various lengths and characteristics themselves. Let V be the set of vertices in the network and $\Phi : V \times V \to \mathbb{R}^p$ denote the feature map. Then,

$$P(A_{ij} = +1) = \frac{1}{1 + \exp\left(-w_0 - \langle \mathbf{w}, \Phi(i,j) \rangle\right)},$$

using which logistic regression is used to learn w_0 and the weight vector $\mathbf{w} = [w_1 \cdots w_p]^T \in \mathbb{R}^p$. The prediction of any query (i, j) is then given by $\text{sign}(P(A_{ij} = +1) - 0.5)$.

References

Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25 (3):211–230, 2003.

Pranay Anchuri and Magdon-Ismail. Communities and balance in signed networks: A spectral approach. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining*, pages 235–242, 2012.

Nikihil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56:89–113, 2004.

Jian-Feng Cai, Emmanuel J. Candés, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *Society for Industrial and Applied Mathematics (SIAM)*, 20(4):1956–1982, 2010.

Emmanuel J. Candés and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9:712–772, 2008.

Emmanuel J. Candés and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transaction of Information Theory*, 56(5):2053–2080, 2009.

Dorwin Cartwright and Frank Harary. Structure balance: A generalization of Heider's theory. *Psychological Review*, 63(5):277–293, 1956.

Nicolo Cesa-Bianchi, Claudio Gentile, Fabio Vitale, , and Giovanni Zappella. A correlation clustering approach to link classification in signed networks. In *The 25th Annual Conference on Learning Theory*, pages 34.1–34.20, 2012.

Kai-Yang Chiang, Nagarajan Natarajan, Ambuj Tewari, and Inderjit S. Dhillon. Exploiting longer cycles for link prediction in signed networks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 1157–1162, 2011.

Kai-Yang Chiang, Joyce Whang, and Inderjit S. Dhillon. Scalable clustering of signed networks using balance normalized cut. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 615–624, 2012.

Fan Chung, Linyuan Lu, and Van Vu. Spectra of random graphs with given expected degrees. *Internet Mathematics*, 1:6313–6318, 2004.

James A. Davis. Clustering and structural balance in graphs. *Human Relations*, 20(2): 181–187, 1967.

Erik D. Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006.

Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.

Patrick Doreian and Andrej Mrvar. A partitioning approach to structural balance. *Social Networks*, 18(21):149–168, 1996.

David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. 2010.

Paul Erdös and Alfréd Rényi. On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

Maryam Fazel, Haitham Hindi, and Stephen P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference*, volume 6, pages 4734–4739, 2001.

Ramanathan V. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th International Conference on World Wide Web*, pages 403–412, 2004.

Frank Harary. On the notion of balance of a signed graph. *Michigan Mathematical Journal*, 2(2):143–146, 1953.

Cho-Jui Hsieh, Kai-Yang Chiang, and Inderjit S. Dhillon. Low rank modeling of signed networks. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 507–515, 2012.

Prateek Jain, Reghu Meka, and Inderjit S. Dhillon. Guaranteed rank minimization via singular value projection. In *Advances in Neural Information Processing Systems*, pages 934–945, 2010.

Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, pages 665–674, 2013.

Thorsten Joachims and John Hopcroft. Error bounds for correlation clustering. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, New York, NY, USA, 2005. ACM.

Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1): 39–43, 1953.

Brian W. Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(1):291–307, 1970.

Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42:30–37, 2009.

Jérôme Kunegis, Andreas Lommatzsch, and Christian Bauckhage. The slashdot zoo: Mining a social network with negative edges. In *Proceedings of the 18th International Conference on World Wide Web*, pages 741–750, 2009.

Jérôme Kunegis, Stephan Schmidt, Andreas Lommatzsch, Jürgen Lerner, Ernesto W. DeLuca, and Sahin Albayrak. Spectral analysis of signed graphs for clustering, prediction and visualization. In *Proceedings of the SIAM International Conference on Data Mining*, pages 559–570, 2010.

Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th International Conference on World Wide Web*, pages 641–650, 2010a.

Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1361–1370, 2010b.

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 437–452, 2011.

Mark Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of USA*, 103(23):8577–8582, 2006.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856. MIT Press, 2001.

Arnout van de Rijt. The micro-macro link for the theory of structural balance. *Journal of Mathematical Sociology*, 35(1):94–113, 2011.

Bo Yang, William Cheung, and Jiming Liu. Community mining from signed social networks. *IEEE Transaction on Knowledge and Data Engineering*, 19(10):1333–1348, 2007.

Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit S. Dhillon. Parallel matrix factorization for recommender systems. *Knowledge and Information Systems(KAIS)*, 2013.

Journal of Machine Learning Research 15 (2014) 1215-1247 Submitted 1/13; Revised 12/13; Published 4/14

Bayesian Nonparametric Comorbidity Analysis of Psychiatric Disorders

Francisco J. R. Ruiz* FRANRRUIZ@TSC.UC3M.ES
Isabel Valera* IVALERA@TSC.UC3M.ES
Department of Signal Processing and Communications
University Carlos III in Madrid
Avda. de la Universidad, 30
28911 Leganés (Madrid, Spain)

Carlos Blanco CBLANCO@NYSPI.COLUMBIA.EDU
Department of Psychiatry, New York State Psychiatric Institute
Columbia University
1051 Riverside Drive, Unit #69
New York, NY 10032 (United States of America)

Fernando Perez-Cruz FERNANDO@TSC.UC3M.ES
Department of Signal Processing and Communications
University Carlos III in Madrid
Avda. de la Universidad, 30
28911 Leganés (Madrid, Spain)

Editor: Athanasios Kottas

Abstract

The analysis of comorbidity is an open and complex research field in the branch of psychiatry, where clinical experience and several studies suggest that the relation among the psychiatric disorders may have etiological and treatment implications. In this paper, we are interested in applying latent feature modeling to find the latent structure behind the psychiatric disorders that can help to examine and explain the relationships among them. To this end, we use the large amount of information collected in the National Epidemiologic Survey on Alcohol and Related Conditions (NESARC) database and propose to model these data using a nonparametric latent model based on the Indian Buffet Process (IBP). Due to the discrete nature of the data, we first need to adapt the observation model for discrete random variables. We propose a generative model in which the observations are drawn from a multinomial-logit distribution given the IBP matrix. The implementation of an efficient Gibbs sampler is accomplished using the Laplace approximation, which allows integrating out the weighting factors of the multinomial-logit likelihood model. We also provide a variational inference algorithm for this model, which provides a complementary (and less expensive in terms of computational complexity) alternative to the Gibbs sampler allowing us to deal with a larger number of data. Finally, we use the model to analyze comorbidity among the psychiatric disorders diagnosed by experts from the NESARC database.

Keywords: Bayesian nonparametrics, Indian buffet process, categorical observations, multinomial-logit function, Laplace approximation, variational inference

∗. Both authors contributed equally.

1. Introduction

Health care increasingly needs to address the management of individuals with multiple coexisting diseases, who are now the norm, rather than the exception. In the United States, about 80% of Medicare spending is devoted to patients with four or more chronic conditions, with costs growing as the number of chronic conditions increases (Wolff et al., 2002). This explains the growing interest of researchers in the impact of comorbidity on a range of outcomes, such as mortality, health-related quality of life, functioning, and quality of health care. However, attempts to study the impact of comorbidity are complicated by the lack of consensus about how to define and measure it (Valderas et al., 2009).

Comorbidity becomes particularly relevant in psychiatry, where clinical experience and several studies suggest that the relation among the psychiatric disorders may have etiological and treatment implications. Several studies have focused on the search of the underlying interrelationships among psychiatric disorders, which can be useful to analyze the structure of the diagnostic classification system, and guide treatment approaches for each disorder (Blanco et al., 2013). Krueger (1999) found that 10 psychiatric disorders (available in the National Comorbidity Survey) can be explained by only two correlated factors, one corresponding to internalizing disorders and the other to externalizing disorders. The existence of the internalizing and the externalizing factors was also confirmed by Kotov et al. (2011). More recently, Blanco et al. (2013) have used factor analysis to find the latent feature structure under 20 common psychiatric disorders, drawing on data from the National Epidemiologic Survey on Alcohol and Related Conditions (NESARC). In particular, the authors found that three correlated factors, one related to externalizing, and the other two to internalizing disorders, characterized well the underlying structure of these 20 diagnoses. From a statistical point of view, the main limitation of this study lies on the use of factor analysis, which assumes that the number of factors is known and that the observations are Gaussian distributed. However, the latter assumption does not fit the observed data, since they are discrete in nature.

In order to avoid the model selection step needed to infer the number of factors in factor analysis, we can resort to Bayesian nonparametric tools, which allow an open-ended number of degrees of freedom in a model (Jordan, 2010). In this paper, we apply the Indian Buffet Process (IBP) (Griffiths and Ghahramani, 2011), because it allows us to infer which latent features influence the observations and how many features there are. We adapt the observation model for discrete random variables, as the discrete nature of the data does not allow using the standard Gaussian observation model. There are several options for modeling discrete outputs given the hidden latent features, like a Dirichlet distribution or sampling from the features, but we opted for the generative model partially introduced by Ruiz et al. (2012), in which the observations are drawn from a multinomial-logit distribution, because it resembles the standard Gaussian observation model, as the observation probability distribution depends on the IBP matrix weighted by some factors.

The IBP model combined with discrete observations has already been tackled in several related works. Williamson et al. (2010) propose a model that combines properties from both the hierarchical Dirichlet process (HDP) and the IBP, called IBP compound Dirichlet (ICD) process. They apply the ICD to focused topic modeling, where the instances are documents and the observations are words from a finite vocabulary, and focus on decoupling

the prevalence of a topic in a document and its prevalence in all documents. Despite the discrete nature of the observations under this model, these assumptions are not appropriate for observations such as the set of possible diagnoses or responses to the questions from the NESARC database, since categorical observations can only take values from a finite set where elements do not present any particular ordering. Titsias (2007) introduced the infinite gamma-Poisson process as a prior probability distribution over non-negative integer valued matrices with a potentially infinite number of columns, and he applied it to topic modeling of images. In this model, each (discrete) component in the observation vector of an instance depends only on one of the active latent features of that object, randomly drawn from a multinomial distribution. Therefore, different components of the observation vector might be equally distributed. Our model is more flexible in the sense that it allows different probability distributions for every component in the observation vector, which is accomplished by weighting differently the latent variables. Furthermore, a preliminary version of this model has been successfully applied to identify the factors that model the risk of suicide attempts (Ruiz et al., 2012).

The rest of the paper is organized as follows. In Section 2, we review the IBP model and the basic Gibbs sampling inference for the IBP, and set the notation used throughout the paper. In Section 3, we propose the generative model which combines the IBP with discrete observations generated from a multinomial-logit distribution. In this section, we focus on the inference based on the Gibbs sampler, where we make use of the Laplace approximation to integrate out the random weighting factors in the observation model. In Section 4, we develop a variational inference algorithm that presents lower computational complexity than the Gibbs sampler. In Section 5, we validate our model on synthetic data and apply it over the real data extracted from the NESARC database. Finally, Section 6 is devoted to the conclusions.

2. The Indian Buffet Process

Unsupervised learning aims to recover the latent structure responsible for generating the observed properties of a set of objects. In latent feature modeling, the properties of each object can be represented by an unobservable vector of latent features, and the observations are generated from a distribution determined by those latent feature values. Typically, we have access to the set of observations and the main goal of latent feature modeling is to find out the latent variables that represent the data.

The most common nonparametric tool for latent feature modeling is the Indian Buffet Process (IBP). The IBP places a prior distribution over binary matrices, in which the number of rows is finite but the number of columns (features) K is potentially unbounded, that is, $K \to \infty$. This distribution is invariant to the ordering of the features and can be derived by taking the limit of a properly defined distribution over $N \times K$ binary matrices as K tends to infinity (Griffiths and Ghahramani, 2011), similarly to the derivation of the Chinese restaurant process as the limit of a Dirichlet-multinomial model (Aldous, 1985). However, given a finite number of data points N, it ensures that the number of non-zero columns, namely, K_+, is finite with probability one.

Let $\mathbf{Z}$ be a random $N \times K$ binary matrix distributed following an IBP, i.e., $\mathbf{Z} \sim \text{IBP}(\alpha)$, where α is the concentration parameter of the process, which controls the number of non-zero

columns K_+. The n^{th} row of $\mathbf{Z}$, denoted by $\mathbf{z}_{n\bullet}$, represents the vector of latent features of the n^{th} data point, and every entry nk is denoted by z_{nk}. Note that each element $z_{nk} \in \{0, 1\}$ indicates whether the k^{th} feature contributes to the n^{th} data point. Since only the K_+ non-zero columns of $\mathbf{Z}$ contain the features of interest, and due to the exchangeability property of the features under the IBP prior, they are usually grouped in the left hand side of the matrix, as illustrated in Figure 1.

Given a binary latent feature matrix $\mathbf{Z}$, we assume that the $N \times D$ observation matrix $\mathbf{X}$, where the n^{th} row contains a D-dimensional observation vector $\mathbf{x}_{n\bullet}$, is distributed according to a probability distribution $p(\mathbf{X}|\mathbf{Z})$. For instance, in the standard observation model by Griffiths and Ghahramani (2011), $p(\mathbf{X}|\mathbf{Z})$ is a Gaussian probability density function. Throughout the paper, we denote by $\mathbf{x}_{\bullet d}$ the d^{th} column of $\mathbf{X}$, and the elements in $\mathbf{X}$ by x_{nd}.

$$\mathbf{Z} = \underbrace{\left[\begin{array}{cccc|ccc} z_{11} & z_{12} & \cdots & z_{1K_+} & 0 & 0 & \cdots \\ z_{21} & z_{22} & \cdots & z_{2K_+} & 0 & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \\ z_{N1} & z_{N2} & \cdots & z_{NK_+} & 0 & 0 & \cdots \end{array}\right]}_{K \text{ columns (features)}} \Bigg\} N \text{ data points}$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{K_+ \text{ non-zero columns}}$$

Figure 1: Illustration of an IBP matrix.

2.1 The Stick-Breaking Construction

The stick-breaking construction of the IBP is an equivalent representation of the IBP prior, useful for inference algorithms other than Gibbs sampling, such as slice sampling or variational inference algorithms (Teh et al., 2007; Doshi-Velez et al., 2009).

In this representation, the probability of each latent feature being active is represented explicitly by a random variable. In particular, the probability of feature z_{nk} taking value 1 is denoted by ω_k, that is,

$$z_{nk} \sim \text{Bernouilli}(\omega_k).$$

Since this probability does not depend on n, the stick-breaking representation explicitly shows that the ordering of the data does not affect the distribution.

The probabilities ω_k are, in turn, generated by first drawing a sequence of independent random variables $v_1, v_2, \ldots$ from a beta distribution of the form

$$v_k \sim \text{Beta}(\alpha, 1).$$

Given the sequence of variables $v_1, v_2, \ldots$, the probability ω_1 is assigned to v_1, and each subsequent ω_k is obtained as

$$\omega_k = \prod_{i=1}^{k} v_i,$$

resulting in a decreasing sequence of probabilities ω_k. Specifically, the expected probability of feature z_{nk} being active decreases exponentially with the index k.

This construction can be understood with the stick-breaking process illustrated in Figure 2. Starting with a stick of length 1, at each iteration $k = 1, 2, \ldots$, a piece is broken off at a point v_k relative to the current length of the stick. The variable ω_k corresponds to the length of the stick just broken off, and the other piece of the stick is discarded.

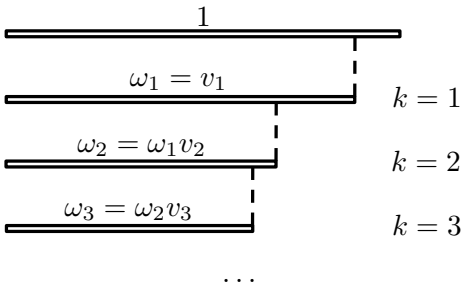


Figure 2: Illustration of the stick-breaking construction of the IBP.

2.2 Inference

Markov Chain Monte Carlo (MCMC) methods have been broadly applied to infer the latent structure $\mathbf{Z}$ from a given observation matrix $\mathbf{X}$ (see, e.g., in Griffiths and Ghahramani (2011); Williamson et al. (2010); Van Gael et al. (2009); Titsias (2007)), being Gibbs sampling the standard method of choice. This algorithm iteratively samples the value of each element z_{nk} given the remaining variables, that is, it samples from

$$p(z_{nk} = 1|\mathbf{X}, \mathbf{Z}_{\neg nk}) \propto p(\mathbf{X}|\mathbf{Z})p(z_{nk} = 1|\mathbf{Z}_{\neg nk}), \tag{1}$$

where $\mathbf{Z}_{\neg nk}$ denotes all the entries of $\mathbf{Z}$ other than z_{nk}. The conditional distribution $p(z_{nk} = 1|\mathbf{Z}_{\neg nk})$ can be readily derived from the exchangeable IBP and can be written as

$$p(z_{nk} = 1|\mathbf{Z}_{\neg nk}) = \frac{m_{-n,k}}{N},$$

where $m_{-n,k}$ is the number of data points with feature k, not including n, i.e., $m_{-n,k} = \sum_{i \neq n} z_{ik}$. For each data point n, after having sampled all elements z_{nk} for the K_+ nonzero columns in $\mathbf{Z}$, the algorithm samples from a distribution (where the prior is a Poisson distribution with mean α/N) a number of new features necessary to explain that data point.

Although MCMC methods perform exact inference, they typically suffer from high computational complexity. To solve this limitation, variational inference algorithms can be applied instead at a lower computational cost, at the expense of performing approximate inference (Jordan et al., 1999). A variational inference algorithm for the IBP under the standard Gaussian observation model is presented by Doshi-Velez et al. (2009). This algorithm makes use of the stick breaking construction of the IBP, summarized above.

3. Observation Model

Unlike the standard Gaussian observation model, let us consider discrete observations, that is, each element $x_{nd} \in \{1, \dots, R_d\}$, where this finite set contains the indexes to all the possible values of x_{nd}. For simplicity and without loss of generality, we consider that $R_d = R$, but the following results can be readily extended to a different cardinality per input dimension, as well as mixing continuous variables with discrete variables, since given the latent feature matrix $\mathbf{Z}$ the columns of $\mathbf{X}$ are assumed to be independent.

We introduce the $K \times R$ matrices $\mathbf{B}^d$ and the length-R row vectors $\mathbf{b}_0^d$ to model the probability distribution over $\mathbf{X}$, such that $\mathbf{B}^d$ links the latent features with the d^{th} column of the observation matrix $\mathbf{X}$, denoted by $\mathbf{x}_{\bullet d}$, and $\mathbf{b}_0^d$ is included to model the bias term in the distribution over the data points. This bias term plays the role of a latent variable that is always active. For a categorical observation space, if we do not have a bias term and all latent variables are inactive, the model assumes that all the outcomes are independent and equally likely, which is not a suitable assumption in most cases. In our application, the bias term is used to model the people that do not suffer from any disorder and it captures the baseline diagnosis in the general population. Additionally, this bias term simplifies the inference since the latent features of those subjects that are not diagnosed any disorder do not need to be sampled.

Hence, we assume that the probability of each element x_{nd} taking value r ($r = 1, \dots, R$), denoted by π_{nd}^r, is given by the multiple-logistic function, i.e.,

$$\pi_{nd}^r = p(x_{nd} = r | \mathbf{z}_{n\bullet}, \mathbf{B}^d, \mathbf{b}_0^d) = \frac{\exp\left(\mathbf{z}_{n\bullet}\mathbf{b}_{\bullet r}^d + b_{0r}^d\right)}{\sum_{r'=1}^{R} \exp\left(\mathbf{z}_{n\bullet}\mathbf{b}_{\bullet r'}^d + b_{0r'}^d\right)}, \tag{2}$$

where $\mathbf{b}_{\bullet r}^d$ denotes the r^{th} column of $\mathbf{B}^d$ and b_{0r}^d denotes the r^{th} element of vector $\mathbf{b}_0^d$. Note that the matrices $\mathbf{B}^d$ are used to weight differently the contribution of each latent feature to every component d, similarly as in the standard Gaussian observation model in Griffiths and Ghahramani (2011). We assume that the mixing vectors $\mathbf{b}_{\bullet r}^d$ are Gaussian distributed with zero mean and covariance matrix $\mathbf{\Sigma}_b = \sigma_B^2 \mathbf{I}$, and the elements b_{0r}^d are also Gaussian distributed with zero mean and variance σ_B^2. The corresponding graphical model is shown in Figure 3.

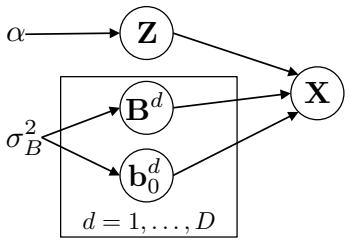


Figure 3: Graphical probabilistic model of the IBP with discrete observations.

The choice of the observation model in Eq. (2), which combines the multiple-logistic function with Gaussian parameters, is based on the fact that it induces dependencies among

the probabilities π_{nd}^r that cannot be captured with other distributions, such as the Dirichlet distribution (Blei and Lafferty, 2007). Furthermore, this multinomial-logistic normal distribution has been widely used to define probability distributions over discrete random variables (Williams and Barber, 1998; Blei and Lafferty, 2007).

We consider that elements x_{nd} are independent given the latent feature matrix $\mathbf{Z}$, the weighting matrices $\mathbf{B}^d$ and the weighting vectors $\mathbf{b}_0^d$. Then, the likelihood for any matrix $\mathbf{X}$ can be expressed as

$$p(\mathbf{X}|\mathbf{Z}, \mathbf{B}^1, \ldots, \mathbf{B}^D, \mathbf{b}_0^1, \ldots, \mathbf{b}_0^D) = \prod_{n=1}^{N} \prod_{d=1}^{D} p(x_{nd}|\mathbf{z}_{n\bullet}, \mathbf{B}^d, \mathbf{b}_0^d) = \prod_{n=1}^{N} \prod_{d=1}^{D} \pi_{nd}^{x_{nd}}. \tag{3}$$

3.1 Laplace Approximation for Gibbs Sampling Inference

In Section 2, the (heuristic) Gibbs sampling algorithm for posterior inference over the latent variables of the IBP, detailed in Griffiths and Ghahramani (2011), has been briefly reviewed. To sample from Eq. (1), we need to integrate out $\mathbf{B}^d$ and $\mathbf{b}_0^d$ in (3), as sequentially sampling from the posterior distribution of these variables is intractable, for which an approximation is required. We rely on the Laplace approximation to integrate out the parameters $\mathbf{B}^d$ and $\mathbf{b}_0^d$ for simplicity and ease of implementation. We first consider the finite form of the proposed model, where K is bounded.

We can simplify the notation in Eqs. 2 and 3 by considering an extended latent feature matrix $\mathbf{Z}$ of size $N \times (K+1)$, in which the elements of the first column are equal to one, and D extended weighting matrices $\mathbf{B}^d$ of size $(K+1) \times R$, in which the first row equals the vector $\mathbf{b}_0^d$. With these definitions, Eq. (2) can be rewritten as

$$\pi_{nd}^r = p(x_{nd} = r|\mathbf{z}_{n\bullet}, \mathbf{B}^d) = \frac{\exp\left(\mathbf{z}_{n\bullet}\mathbf{b}_{\bullet r}^d\right)}{\sum_{r'=1}^{R} \exp\left(\mathbf{z}_{n\bullet}\mathbf{b}_{\bullet r'}^d\right)}.$$

Unless otherwise specified, we use the simplified notation throughout this section. For this reason, the index k over the latent variables takes the values in $\{0, 1, \ldots, K\}$, with $z_{n0} = 1$ for all n.

Recall that our model assumes independence among the observations given the hidden latent variables. Then, the posterior $p(\mathbf{B}^1, \ldots, \mathbf{B}^D|\mathbf{X}, \mathbf{Z})$ factorizes as

$$p(\mathbf{B}^1, \ldots, \mathbf{B}^D|\mathbf{X}, \mathbf{Z}) = \prod_{d=1}^{D} p(\mathbf{B}^d|\mathbf{x}_{\bullet d}, \mathbf{Z}) = \prod_{d=1}^{D} \frac{p(\mathbf{x}_{\bullet d}|\mathbf{B}^d, \mathbf{Z})p(\mathbf{B}^d)}{p(\mathbf{x}_{\bullet d}|\mathbf{Z})}.$$

Hence, we only need to deal with each term $p(\mathbf{B}^d|\mathbf{x}_{\bullet d}, \mathbf{Z})$ individually. The marginal likelihood $p(\mathbf{x}_{\bullet d}|\mathbf{Z})$, which we are interested in, can be obtained as

$$p(\mathbf{x}_{\bullet d}|\mathbf{Z}) = \int p(\mathbf{x}_{\bullet d}|\mathbf{B}^d, \mathbf{Z})p(\mathbf{B}^d)d\mathbf{B}^d. \tag{4}$$

Although the prior $p(\mathbf{B}^d)$ is Gaussian, due to the non-conjugacy with the likelihood term, the computation of this integral, as well as the computation of the posterior $p(\mathbf{B}^d|\mathbf{x}_{\bullet d}, \mathbf{Z})$, turns out to be intractable.

Following a similar procedure as in Gaussian processes for multiclass classification (Williams and Barber, 1998), we approximate the posterior $p(\mathbf{B}^d|\mathbf{x}_{\bullet d}, \mathbf{Z})$ as a Gaussian distribution using Laplace's method. In order to obtain the parameters of the Gaussian distribution, we define $f(\mathbf{B}^d)$ as the un-normalized log-posterior of $p(\mathbf{B}^d|\mathbf{x}_{\bullet d}, \mathbf{Z})$, i.e.,

$$f(\mathbf{B}^d) = \log p(\mathbf{x}_{\bullet d}|\mathbf{B}^d, \mathbf{Z}) + \log p(\mathbf{B}^d). \tag{5}$$

As proven in Appendix A, the function $f(\mathbf{B}^d)$ is a strictly concave function of $\mathbf{B}^d$ and therefore it has a unique maximum, which is reached at $\mathbf{B}^d_{\text{MAP}}$, denoted by the subscript 'MAP' (*maximum a posteriori*) because it coincides with the mean of the Gaussian distribution in the Laplace's approximation. We resort to Newton's method to compute $\mathbf{B}^d_{\text{MAP}}$.

We stack the columns of $\mathbf{B}^d$ into $\boldsymbol{\beta}^d$, i.e., $\boldsymbol{\beta}^d = \mathbf{B}^d(:)$ for avid Matlab users. The posterior $p(\mathbf{B}^d|\mathbf{x}_{\bullet d}, \mathbf{Z})$ can be approximated as

$$p(\boldsymbol{\beta}^d|\mathbf{x}_{\bullet d}, \mathbf{Z}) \approx \mathcal{N}\left(\boldsymbol{\beta}^d \,\middle|\, \boldsymbol{\beta}^d_{\text{MAP}}, (-\nabla\nabla f)|_{\boldsymbol{\beta}^d_{\text{MAP}}}\right),$$

where $\boldsymbol{\beta}^d_{\text{MAP}}$ contains all the columns of $\mathbf{B}^d_{\text{MAP}}$ stacked into a vector and $\nabla\nabla f$ is the Hessian of $f(\boldsymbol{\beta}^d)$. Hence, by taking the second-order Taylor series expansion of $f(\boldsymbol{\beta}^d)$ around its maximum, the computation of the marginal likelihood in (4) results in a Gaussian integral, whose solution can be expressed as

$$\begin{aligned}\log p(\mathbf{x}_{\bullet d}|\mathbf{Z}) \approx & -\frac{1}{2\sigma_B^2}\,\text{trace}\left\{(\mathbf{B}^d_{\text{MAP}})^\top \mathbf{B}^d_{\text{MAP}}\right\} \\ & -\frac{1}{2}\log\left|\mathbf{I}_{R(K+1)} + \sigma_B^2 \sum_{n=1}^{N}\left(\text{diag}(\widehat{\boldsymbol{\pi}}_{nd}) - (\widehat{\boldsymbol{\pi}}_{nd})^\top \widehat{\boldsymbol{\pi}}_{nd}\right) \otimes (\mathbf{z}_{n\bullet}^\top \mathbf{z}_{n\bullet})\right| + \log p(\mathbf{x}_{\bullet d}|\mathbf{B}^d_{\text{MAP}}, \mathbf{Z}),\end{aligned} \tag{6}$$

where $\widehat{\boldsymbol{\pi}}_{nd}$ is the vector $\boldsymbol{\pi}_{nd} = \left[\pi^1_{nd}, \pi^2_{nd}, \ldots, \pi^R_{nd}\right]$ evaluated at $\mathbf{B}^d = \mathbf{B}^d_{\text{MAP}}$, and $\text{diag}(\widehat{\boldsymbol{\pi}}_{nd})$ is a diagonal matrix with the values of $\widehat{\boldsymbol{\pi}}_{nd}$ as its diagonal elements.

Similarly as in Griffiths and Ghahramani (2011), it is straightforward to prove that the limit of Eq. (6) is well-defined if $\mathbf{Z}$ has an unbounded number of columns, that is, as $K \to \infty$. The resulting expression for the marginal likelihood $p(\mathbf{x}_{\bullet d}|\mathbf{Z})$ can be readily obtained from Eq. (6) by replacing K by K_+, $\mathbf{Z}$ by the submatrix containing only the non-zero columns of $\mathbf{Z}$, and $\mathbf{B}^d_{\text{MAP}}$ by the submatrix containing the K_++1 corresponding rows.

3.2 Speeding Up the Matrix Inversion

In this section, we propose a method that reduces the complexity of computing the inverse of the Hessian for Newton's method (as well as its determinant) from $\mathcal{O}(R^3K_+^3 + NR^2K_+^2)$ to $\mathcal{O}(RK_+^3 + NR^2K_+^2)$, effectively accelerating the inference procedure for large values of R.

Let us denote with $\mathbf{Z}$ the matrix that contains only the $K_+ + 1$ non-zero columns of the extended full IBP matrix. The inverse of the Hessian for Newton's method, as well as its determinant in (6), can be efficiently carried out if we rearrange the inverse of $\nabla\nabla f$ as follows:

$$(-\nabla\nabla f)^{-1} = \left(\mathbf{D} - \sum_{n=1}^{N}\mathbf{v}_n\mathbf{v}_n^\top\right)^{-1},$$

where $\mathbf{v}_n = (\boldsymbol{\pi}_{nd})^\top \otimes \mathbf{z}_{n\bullet}^\top$ and $\mathbf{D}$ is a block-diagonal matrix, in which each diagonal submatrix is given by

$$\mathbf{D}_r = \frac{1}{\sigma_B^2}\mathbf{I}_{K_+ +1} + \mathbf{Z}^\top \operatorname{diag}(\boldsymbol{\pi}_{\bullet d}^r)\,\mathbf{Z}, \tag{7}$$

with $\boldsymbol{\pi}_{\bullet d}^r = \left[\ \pi_{1d}^r, \ldots, \pi_{Nd}^r\ \right]^\top$. Since $\mathbf{v}_n\mathbf{v}_n^\top$ is a rank-one matrix, we can apply the Woodbury identity (Woodbury, 1949) N times to invert the matrix $-\nabla\nabla f$, similar to the RLS (Recursive Least Squares) updates (Haykin, 2002). At each iteration $n = 1, \ldots, N$, we compute

$$(\mathbf{D}^{(n)})^{-1} = \left(\mathbf{D}^{(n-1)} - \mathbf{v}_n\mathbf{v}_n^\top\right)^{-1} = (\mathbf{D}^{(n-1)})^{-1} + \frac{(\mathbf{D}^{(n-1)})^{-1}\mathbf{v}_n\mathbf{v}_n^\top(\mathbf{D}^{(n-1)})^{-1}}{1 - \mathbf{v}_n^\top(\mathbf{D}^{(n-1)})^{-1}\mathbf{v}_n}. \tag{8}$$

For the first iteration, we define $\mathbf{D}^{(0)}$ as the block-diagonal matrix $\mathbf{D}$, whose inverse matrix involves computing the R matrix inversions of size $(K_+ + 1) \times (K_+ + 1)$ of the matrices in (7), which can be efficiently solved applying the Matrix Inversion Lemma. After N iterations of (8), it turns out that $(-\nabla\nabla f)^{-1} = (\mathbf{D}^{(N)})^{-1}$.

In practice, there is no need to iterate over all observations, since all subjects sharing the same latent feature vector $\mathbf{z}_{n\bullet}$ and observation x_{nd} can be grouped together, therefore requiring (at most) $R2^{K_+}$ iterations instead of N. In our applications, it provides significant savings in run-time complexity, since $R2^{K_+} \ll N$.

For the determinant in (6), similar recursions can be applied using the Matrix Determinant Lemma (Harville, 1997), which states that $|\mathbf{D} + \mathbf{v}\mathbf{u}^\top| = (1 + \mathbf{v}^\top\mathbf{D}\mathbf{u})|\mathbf{D}|$, and $|\mathbf{D}^{(0)}| = \prod_{r=1}^R |\mathbf{D}_r|$.

4. Variational Inference

Variational inference provides a complementary (and less expensive in terms of computational complexity) alternative to MCMC methods as a general source of approximation methods for inference in large-scale statistical models (Jordan et al., 1999). In this section, we adapt the infinite variational approach for the linear-Gaussian model with respect to a full IBP prior introduced by Doshi-Velez et al. (2009) to the model proposed in Section 3. This approach assumes the (truncated) stick-breaking construction for the IBP in Section 2.1, which bounds the number of columns of the IBP matrix by a finite (but large enough) value, K. Then, in the truncated stick-breaking process, $\omega_k = \prod_{i=1}^k v_i$ for $k \leq K$ and zero otherwise.

The hyperparameters of the model are contained in the set $\mathcal{H} = \{\alpha, \sigma_B^2\}$ and, similarly, $\Psi = \{\mathbf{Z}, \mathbf{B}^1, \ldots, \mathbf{B}^D, \mathbf{b}_0^1, \ldots, \mathbf{b}_0^D, v_1, \ldots, v_K\}$ denotes the set of unobserved variables in the model. Under the truncated stick-breaking construction for the IBP, the joint probability distribution over all the variables $p(\Psi, \mathbf{X}|\mathcal{H})$ can be factorized as

$$\begin{aligned} p(\Psi, \mathbf{X}|\mathcal{H}) = &\prod_{k=1}^K \left(p(v_k|\alpha) \prod_{n=1}^N p(z_{nk}|\{v_i\}_{i=1}^k) \right) \prod_{d=1}^D \left(p(\mathbf{b}_0^d|\sigma_B^2) \prod_{k=1}^K p(\mathbf{b}_{k\bullet}^d|\sigma_B^2) \right) \\ &\times \prod_{n=1}^N \prod_{d=1}^D p(x_{nd}|\mathbf{z}_{n\bullet}, \mathbf{B}^d, \mathbf{b}_0^d), \end{aligned}$$

where $\mathbf{b}^d_{k\bullet}$ is the k^{th} row of matrix $\mathbf{B}^d$.

We approximate $p(\Psi|\mathbf{X},\mathcal{H})$ with the variational distribution $q(\Psi)$ given by

$$q(\Psi) = \prod_{k=1}^{K}\left(q(v_k|\tau_{k1},\tau_{k2})\prod_{n=1}^{N}q(z_{nk}|\nu_{nk})\right)\prod_{k=0}^{K}\prod_{r=1}^{R}\prod_{d=1}^{D}q(b^d_{kr}|\phi^d_{kr},(\sigma^d_{kr})^2),$$

where the elements of matrix $\mathbf{B}^d$ are denoted by b^d_{kr}, and

$$\begin{aligned}
&q(v_k|\tau_{k1},\tau_{k2}) = \text{Beta}(\tau_{k1},\tau_{k2}),\\
&q(b^d_{kr}|\phi^d_{kr},(\sigma^d_{kr})^2) = \mathcal{N}(\phi^d_{kr},(\sigma^d_{kr})^2),\\
&q(z_{nk}|\nu_{nk}) = \text{Bernoulli}(\nu_{nk}).
\end{aligned}$$

Inference involves optimizing the variational parameters of $q(\Psi)$ to minimize the Kullback-Leibler divergence from $q(\Psi)$ to $p(\Psi|\mathbf{X},\mathcal{H})$, i.e., $D_{KL}(q||p)$. This optimization is equivalent to maximizing a lower bound on the evidence $p(\mathbf{X}|\mathcal{H})$, since

$$\begin{aligned}
\log p(\mathbf{X}|\mathcal{H}) &= \mathbb{E}_q\left[\log p(\Psi,\mathbf{X}|\mathcal{H})\right] + H[q] + D_{KL}(q||p)\\
&\geqslant \mathbb{E}_q\left[\log p(\Psi,\mathbf{X}|\mathcal{H})\right] + H[q],
\end{aligned} \tag{9}$$

where $\mathbb{E}_q[\cdot]$ denotes the expectation with respect to the distribution $q(\Psi)$, $H[q]$ is the entropy of distribution $q(\Psi)$ and

$$\begin{aligned}
\mathbb{E}_q\left[\log p(\Psi,\mathbf{X}|\mathcal{H})\right] = &\sum_{k=1}^{K}\mathbb{E}_q\left[\log p(v_k|\alpha)\right] + \sum_{d=1}^{D}\sum_{k=1}^{K}\mathbb{E}_q\left[\log p(\mathbf{b}^d_{k\bullet}|\sigma^2_B)\right] + \sum_{d=1}^{D}\mathbb{E}_q\left[\log p(\mathbf{b}^d_0|\sigma^2_B)\right]\\
&+\sum_{k=1}^{K}\sum_{n=1}^{N}\mathbb{E}_q\left[\log p(z_{nk}|\{v_i\}_{i=1}^k)\right] + \sum_{n=1}^{N}\sum_{d=1}^{D}\mathbb{E}_q\left[\log p(x_{nd}|\mathbf{z}_{n\bullet},\mathbf{B}^d,\mathbf{b}^d_0)\right].
\end{aligned} \tag{10}$$

The derivation of the lower bound in (9) is straightforward, with the exception of the terms $\mathbb{E}_q\left[\log p(z_{nk}|\{v_i\}_{i=1}^k)\right]$ and $\mathbb{E}_q\left[\log p(x_{nd}|\mathbf{z}_{n\bullet},\mathbf{B}^d,\mathbf{b}^d_0)\right]$ in (10), which have no closed-form solution, so we instead need to bound them. Deriving these bounds leads to a new bound $\mathcal{L}(\mathcal{H},\mathcal{H}_q)$, which can be obtained in closed-form, such that $\log p(\mathbf{X}|\mathcal{H}) \geq \mathcal{L}(\mathcal{H},\mathcal{H}_q)$, being $\mathcal{H}_q$ the full set of variational parameters. The final expression for $\mathcal{L}(\mathcal{H},\mathcal{H}_q)$, as well as the details on the derivation of the bound, are provided in Appendix B.

In order to maximize the lower bound $\mathcal{L}(\mathcal{H},\mathcal{H}_q)$, we need to optimize with respect to the value of the variational parameters. To this end, we can iteratively maximize the bound with respect to each variational parameter by taking the derivative of $\mathcal{L}(\mathcal{H},\mathcal{H}_q)$ and setting it to zero. This procedure readily leads to the following fixed-point equations:

1. For the variational Beta distribution $q(v_k|\tau_{k1},\tau_{k2})$,

$$\begin{aligned}
\tau_{k1} &= \alpha + \sum_{m=k}^{K}\left(\sum_{n=1}^{N}\nu_{nm}\right) + \sum_{m=k+1}^{K}\left(N - \sum_{n=1}^{N}\nu_{nm}\right)\left(\sum_{i=k+1}^{m}\lambda_{mi}\right),\\
\tau_{k2} &= 1 + \sum_{m=k}^{K}\left(N - \sum_{n=1}^{N}\nu_{nm}\right)\lambda_{mk}.
\end{aligned}$$

2. For the Bernoulli distribution $q(z_{nk}|\nu_{nk})$,

$$\nu_{nk} = \frac{1}{1+\exp(-A_{nk})},$$

where

$$\begin{aligned} A_{nk} = &\sum_{i=1}^{k}\left[\psi(\tau_{i1}) - \psi(\tau_{i1}+\tau_{i2})\right] - \left[\sum_{m=1}^{k}\lambda_{km}\psi(\tau_{m2}) + \sum_{m=1}^{k-1}\left(\sum_{n=m+1}^{k}\lambda_{kn}\right)\psi(\tau_{m1})\right. \\ &\left. - \sum_{m=1}^{k}\left(\sum_{n=m}^{k}\lambda_{kn}\right)\psi(\tau_{m1}+\tau_{m2}) - \sum_{m=1}^{k}\lambda_{km}\log(\lambda_{km})\right] \\ &+ \sum_{d=1}^{D}\left(\phi^d_{kx_{nd}} - \xi_{nd}\sum_{r=1}^{R}\left[\exp\left(\phi^d_{0r} + \frac{1}{2}(\sigma^d_{0r})^2\right)\left(1 - \exp\left(\phi^d_{kr} + \frac{1}{2}(\sigma^d_{kr})^2\right)\right)\times\right.\right. \\ &\left.\left.\times\prod_{k'\neq k}\left(1 - \nu_{nk'} + \nu_{nk'}\exp\left(\phi^d_{k'r} + \frac{1}{2}(\sigma^d_{k'r})^2\right)\right)\right]\right), \end{aligned}$$

and $\psi(\cdot)$ stands for the digamma function (Abramowitz and Stegun, 1972, p. 258–259).

3. For the feature assignments, which are Bernoulli distributed given the feature probabilities, we have lower bounded $\mathbb{E}_q\left[\log p(z_{nk}|\{v_i\}_{i=1}^k)\right]$ by using the multinomial approach in Doshi-Velez et al. (2009) (see Appendix B for further details). This approximation introduces the auxiliary multinomial distribution $\boldsymbol{\lambda}_k = [\lambda_{k1}, \ldots, \lambda_{kk}]$, where each λ_{ki} can be updated as

$$\lambda_{ki} \propto \exp\left(\psi(\tau_{i2}) + \sum_{m=1}^{i-1}\psi(\tau_{m1}) - \sum_{m=1}^{i}\psi(\tau_{m1}+\tau_{m2})\right),$$

where the proportionality ensures that $\boldsymbol{\lambda}_k$ is a valid distribution.

4. The maximization with respect to the variational parameters ϕ^d_{kr}, ϕ^d_{0r}, $(\sigma^d_{kr})^2$, and $(\sigma^d_{0r})^2$ has no analytical solution, and therefore, we need to resort to a numerical method to find the maximum, such as Newton's method or conjugate gradient algorithm, for which the first and the second derivatives[1] (given in Appendix C) are required.

5. Finally, we lower bound the likelihood term $\mathbb{E}_q\left[\log p(x_{nd}|\mathbf{z}_{n\bullet}, \mathbf{B}^d, \mathbf{b}^d_0)\right]$ by resorting to a first-order Taylor series expansion around the auxiliary variables ξ^{-1}_{nd} for $n = 1, \ldots, N$ and $d = 1, \ldots, D$ (see Appendix B for further details), which are optimized by the expression

$$\xi_{nd} = \left[\sum_{r=1}^{R}\exp\left(\phi^d_{0r} + \frac{1}{2}(\sigma^d_{0r})^2\right)\prod_{k=1}^{K}\left(1 - \nu_{nk} + \nu_{nk}\exp\left(\phi^d_{kr} + \frac{1}{2}(\sigma^d_{kr})^2\right)\right)\right]^{-1}.$$

1. Note that the second derivatives are strictly negative and, therefore, the maximum with respect to each parameter is unique.

5. Experiments

In this section, we first use a toy example to show how our model with discrete observations works and then we turn to two experiments over the NESARC database.

5.1 Inference over Synthetic Images

We generate an illustrative example inspired by the example in Griffiths and Ghahramani (2011) to show that the proposed model works as expected. We define four base black-and-white images, shown in Figure 4a, that can be present with probability 0.3, independently of the others. These base images are combined to create a binary composite image. We also multiply each white pixel independently with equiprobable binary noise, hence each white pixel in the composite image can be turned black 50% of the times, while black pixels always remain black. We generate 200 observations to learn the IBP model (several examples can be found in Figure 4c). The Gibbs sampler has been initialized with $K_+ = 2$, setting each $z_{nk} = 1$ with probability 1/2, and setting the hyperparameters to $\alpha = 0.5$ and $\sigma_B^2 = 1$.

After 350 iterations, the Gibbs sampler returns four latent features. Each of the four features recovers one of the base images with a different ordering, which is inconsequential. In Figure 4b, we have plotted the posterior probability for each pixel being white, when only one of the components is active. As expected, the black pixels are known to be black (almost zero probability of being white) and the white pixels have about a 50/50 chance of being black or white, due to the multiplicative noise. The Gibbs sampler has used as many as eleven hidden features, as shown in Figure 4e, but after less than 50 iterations, the first four features represent the base images and the others just lock on to a noise pattern, which eventually fades away.

In Figure 4d, we depict the posterior probability of pixels being white for the four images in Figure 4c, given the inferred latent feature vectors for these observations. Note that the model behaves as expected and properly captures the generative process, even for those observations which do not possess any latent features, for which the vectors $\mathbf{b}_0^d$ do not provide significant information about the black-or-white probabilities.

5.2 Comorbidity Analysis of Psychiatric Disorders

In the present study, our objective is to provide an alternative to the factor analysis approach used by Blanco et al. (2013) with the IBP for discrete observations introduced in the present paper. We build an unsupervised model taking the 20 disorders used by Blanco et al. (2013) as input data, drawn from the NESARC data.

The NESARC database was designed to estimate the prevalence of psychiatric disorders, as well as their associated features and level of disability. The NESARC had two waves of interviews (first wave in 2001-2002 and second wave in 2004-2005). For the following experimental results, we only use the data from the first wave, for which 43,093 people were selected to represent the U.S. population of 18 years of age and older. Through 2,991 entries, the NESARC collects data on the background of participants, alcohol and other drug use and use disorders, and other mental disorders. Public use data are currently available for this wave of data collection.[2]

2. See http://aspe.hhs.gov/hsp/06/catalog-ai-an-na/nesarc.htm

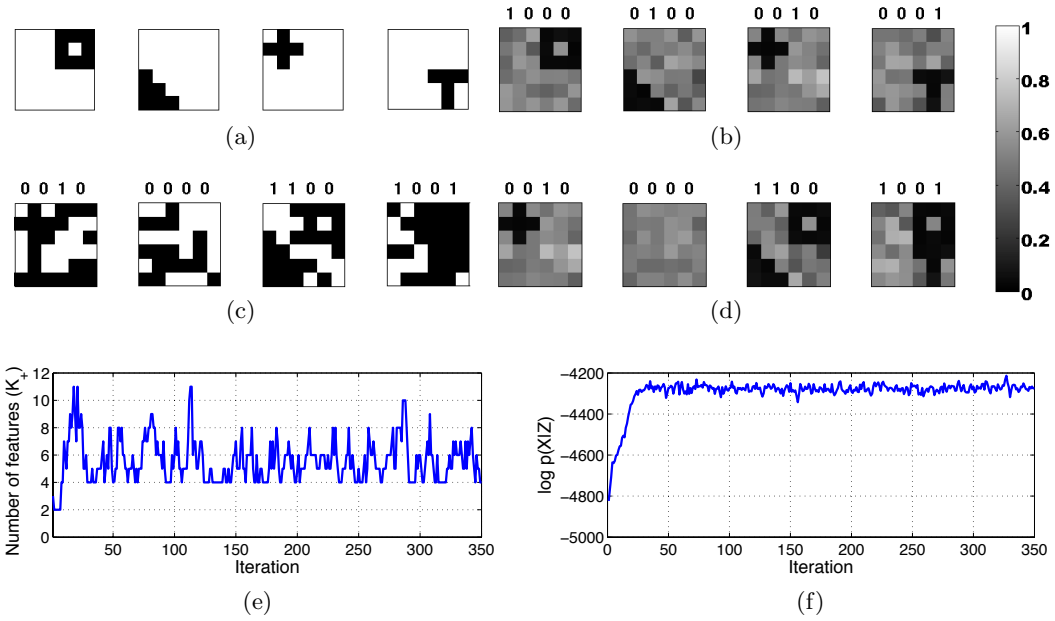


Figure 4: Experimental results of the infinite binary multinomial-logistic model over the image data set. (a) The four base images used to generate the 200 observations. (b) Probability of each pixel being white, when a single feature is active (ordered to match the images on the left), computed using the matrices $\mathbf{B}^d_{\text{MAP}}$. (c) Four data points generated as described in the text. The numbers above each figure indicate which features are present in that image. (d) Probabilities of each pixel being white after 350 iterations of the Gibbs sampler inferred for the four data points on (c). The numbers above each figure show the inferred value of $\mathbf{z}_{n\bullet}$ for these data points. (e) The number of latent features K_+ and (f) the approximate log of $p(\mathbf{X}|\mathbf{Z})$ over the 200 iterations of the Gibbs sampler.

The 20 disorders include substance use disorders (alcohol abuse and dependence, drug abuse and dependence and nicotine dependence), mood disorders (major depressive disorder (MDD), bipolar disorder and dysthymia), anxiety disorders (panic disorder, social anxiety disorder (SAD), specific phobia and generalized anxiety disorder (GAD)), pathological gambling (PG) and seven personality disorders (avoidant, dependent, obsessive-compulsive (OC), paranoid, schizoid, histrionic and antisocial personality disorders (PDs)).

We run the Gibbs sampler over $3,500$ randomly chosen subjects out of the 43,093 participants in the survey, having initialized the sampler with an active feature, i.e., $K_+ = 1$, having set $z_{nk} = 1$ randomly with probability 0.5, and fixing $\alpha = 1$ and $\sigma_B^2 = 1$. After convergence, we run an additional Gibbs sampler with 10 iterations for each of the remaining subjects in the database, restricted to their latent features (that is, we fix the latent features learned for the $3,500$ subjects to sample the feature vector of each subject). Then, we run additional iterations of the Gibbs sampler over the whole database, finally obtaining three latent features. In order to speed up the sampling procedure, we do not sample the

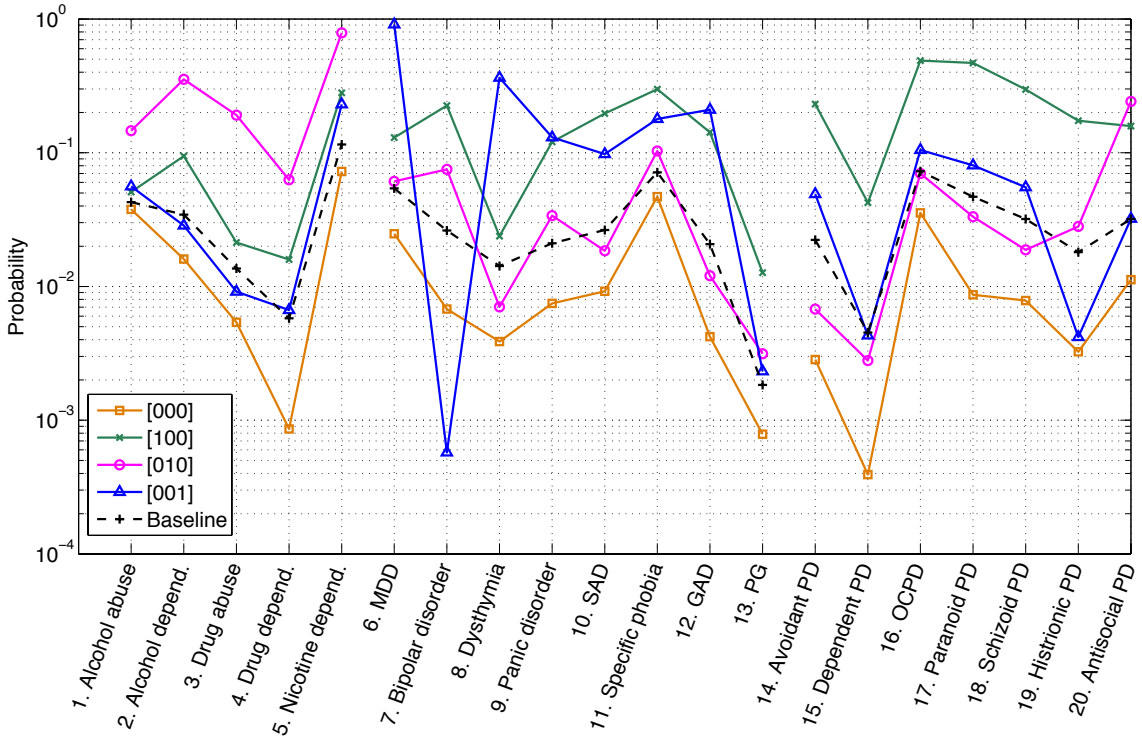


Figure 5: Probabilities of suffering from the 20 considered disorders. These probabilities have been obtained using the matrices $\mathbf{B}^d_{\text{MAP}}$, when none or a single latent feature is active. The legend shows the latent feature vector corresponding to each curve. The baseline has been obtained taking into account the $43,093$ subjects in the database.

rows of $\mathbf{Z}$ corresponding to those subjects who do not suffer from any of the 20 disorders, but instead fix these latent features to zero. The idea is that the $\mathbf{b}^d_0$ terms must capture the general population that does not suffer from any psychiatric disorder, and we use the active components of the matrix $\mathbf{Z}$ to characterize the disorders.

To examine the three latent features, we plot in Figure 5 the posterior probability of having each of the considered disorders, when none or one of the latent features is active. As expected, for those subjects who do not possess any feature, the probability of having any of the disorders is below the baseline level (due to the contribution of the vectors $\mathbf{b}^d_0$), defined as the empirical probability in the full sample, that is, taking into account the $43,093$ participants. Feature 1 increases the probability of having all the disorders, and thus seems to represent a general psychopathology factor, although it may particularly increase the risk of personality disorders. Feature 2 models substance use disorders and antisocial personality disorder, consistent with the externalizing factor identified in previous studies of the structure of psychiatric disorders (Krueger, 1999; Kendler et al., 2003; Vollebergh et al., 2001; Blanco et al., 2013). Feature 3 models mood or anxiety disorders, and thus seems to represent the internalizing factor also identified in previous studies.

Feature vector	1 x x	x 1 x	x x 1
Empirical Probability	0.0748	0.0330	0.0227

(a)

Feature vector	1 1 x	1 x 1	x 1 1
Empirical Probability	0.0028	0.0012	0.0009
Product Probability	0.0025	0.0017	0.0007

(b)

Table 1: Probabilities of possessing at least (a) one latent feature, or (b) two latent features, as given in the patterns shown in the heading rows. The symbol 'x' denotes either 0 or 1. The 'empirical probability' rows contain the probabilities extracted directly from the inferred IBP matrix $\mathbf{Z}$, while the 'product probability' row shows the product of the corresponding two latent feature probabilities given in (a).

Thus, in accord to previous results from the studies on the latent structure of the comorbidity of psychiatric disorders, detailed in Section 1, we find that the patterns of comorbidity of common psychiatric disorders can be well described by a small number of latent features. In addition, nosologically related disorders, such as social anxiety disorder and avoidant personality disorder, tend to be modeled by similar features. As found in previous results (Blanco et al., 2013), no disorder is perfectly aligned along one single latent feature, therefore suggesting that disorders can develop through multiple etiological paths. For instance, the risk of nicotine dependence may be particularly high in individuals with a propensity towards externalization or internalization.

In Table 1a, we first show the empirical probability of possessing each latent feature, that is, the number of subjects in the database that possess each latent feature divided by the total number of subjects. We also show in Table 1b the probability of possessing at least two features as the product of the probabilities in Table 1a (Product Probability), and also the empirical probability. We include Table 1 to show that the three features are nearly independent of one another, since the probability of possessing any two particular features is close to the product of the probabilities of possessing them individually. The differences in Table 1b are not statistically significant. Then, besides explicitly capturing the probability of each disorder, our model also provides a way to measure independence among the latent features. Note that although the proposed model assumes that the latent features are independent *a priori*, we could have found that the empirical probability does not correspond to the product one. Therefore, the independence among the three latent features follows the model's assumption and, from a psychiatric perspective, it also shows that the three factors (internalizing, externalizing and general psychopathology factor) are independent one another, that is, suffering from one group of disorders does not imply an increased probability of suffering from any other group of disorders.

Finally, we remark that we have also applied the variational inference algorithm to study the comorbidity patterns of psychiatric disorders but, since both algorithms (the variational and the Gibbs sampler) infer the same three latent features, we only plot the results for the

Gibbs sampling algorithm in this section and apply the variational inference algorithm in next section.

5.3 Comorbidity Analysis of Personality Disorders

In order to identify the seven personality disorders studied in the previous section, psychiatrists have established specific diagnostic criteria for each of them. These criteria correspond to affirmative responses to one or several questions in the NESARC survey and this correspondence is shown in Appendix D. Then, there exists a set of criteria to identify if a subject presents any of the following personality disorders: avoidant, dependent, obsessive-compulsive, paranoid, schizoid, histrionic and antisocial. In the present analysis, we consider as input data the fulfillment of the 52 criteria (i.e., $R = 2$) corresponding to all the disorders for the 43,093 subjects and we apply the variational inference algorithm truncated to $K = 25$ features, as detailed in Section 4, to find the latent structure of the data.

In order to properly initialize the huge amount of variational parameters, we have previously run six Gibbs samplers over the data but taking only the criteria corresponding to the avoidant PD and another PD (that is, the seven criteria for the avoidant PD and the seven for the dependent PD, the criteria for the avoidant PD with the eight for the OCPD, etc.) for $10,000$ randomly chosen subjects. After running the six Gibbs samplers, we obtain 18 latent features that we group in a unique matrix $\mathbf{Z}$ to obtain the weighting matrices $\mathbf{B}^d_{\text{MAP}}$, which are used to initialize some parameters ν_{nk} and ϕ^d_{kr}. We do this because the variational algorithm is sensitive to the starting point and a random initialization would not produce good solutions.

We run enough iterations of the variational algorithm to ensure convergence of the variational lower bound (the lower bound at each iteration is shown in Figure 6). We construct a binary matrix $\mathbf{Z}$ by setting each element $z_{nk} = 1$ if $\nu_{nk} > 0.5$. We flip (changing zeros by ones, and vice versa) those features possessed by more than 80% of the subjects, obtaining only 10 latent features possessed by more than 50 subjects among the $43,093$ in the database and then recomputing the weighting matrices. In Table 2, we show the probability of occurrence of each feature (top row), as well as the probability of having active only one single feature (bottom row). We also show the 'empirical' and the 'product' probabilities of possessing at least two latent features in Table 3, and the probabilities of possessing at least two features given that one of them is active in Table 4.

Features	1	2	3	4	5	6	7	8	9	10
Total	43.45	19.01	15.28	13.99	11.76	8.97	7.54	6.91	1.86	1.43
Single feature	13.48	3.62	2.22	1.34	2.27	0.49	0.76	1.07	0	0

Table 2: Probabilities (%) of possessing (top row) at least one latent feature, or (bottom row) a single feature.

In Figure 7, we plot the probability of meeting each criterion in the general population (dashed line) and the probability of meeting each criterion for those subjects that do not have any active feature in our model (solid line). There are $15,185$ subjects (35.2% of the

Features	1	2	3	4	5	6	7	8	9	10
1		9.92	8.96	8.48	5.67	7.22	4.92	3.85	1.46	1.42
2	8.26		4.43	4.54	3.67	1.90	1.43	2.08	0.71	0.21
3	6.64	2.90		3.29	2.18	3.00	2.02	1.58	0.54	0.20
4	6.08	2.66	2.14		2.79	1.91	2.39	1.40	1.25	0.03
5	5.11	2.23	1.80	1.65		1.31	1.35	0.85	0.57	0.00
6	3.90	1.71	1.37	1.26	1.05		1.10	0.80	0.44	0.14
7	3.28	1.43	1.15	1.06	0.89	0.68		0.65	0.28	0.00
8	3.00	1.31	1.06	0.97	0.81	0.62	0.52		0.51	0.07
9	0.81	0.35	0.28	0.26	0.22	0.17	0.14	0.13		0.00
10	0.62	0.27	0.22	0.20	0.17	0.13	0.11	0.10	0.03	

Table 3: Probabilities (%) of possessing at least two latent features. The elements above the diagonal correspond to the 'empirical probability', that is, extracted directly from the inferred IBP matrix $\mathbf{Z}$, and the elements below the diagonal correspond to the 'product probability' of the corresponding two latent feature probabilities given in the first row of Table 2.

population) which do not present any active feature, and for these people the probability of meeting any criterion is reduced significantly.

We have found results that are in accordance with previous studies and at the same time provide new information to understand personality disorders. Out of the 10 features, 6 of them directly describe personality disorders. Feature 1 increases the probability of fulfilling the criteria for OCPD, Feature 3 increases the probability of fulfilling the criteria for antisocial, Feature 4 increases the probability of fulfilling the criteria for paranoid, Feature 5 increases the probability of meeting the criteria for schizoid, Feature 8 increases the probability of fulfilling the criteria for histrionic and Feature 7 increases the probability of meeting the criteria for avoidant and dependent. In Figure 8, we plot the probability ratio between the probability of meeting each criterion when a single feature is active with respect to the probability of meeting each criterion in the general population (baseline in

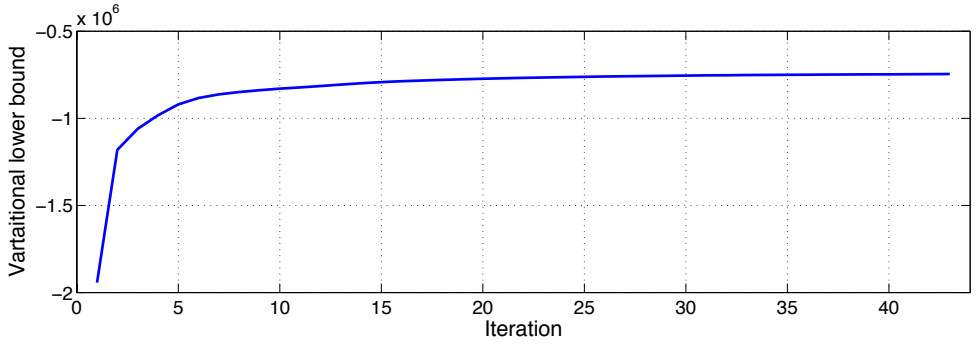


Figure 6: Variational lower bound $\mathcal{L}(\mathcal{H}, \mathcal{H}_q)$ at each iteration.

k_1 \ k_2	1	2	3	4	5	6	7	8	9	10
1	100	22.83	20.63	19.53	13.05	16.62	11.33	8.85	3.37	3.27
2	52.19	100	23.33	23.90	19.32	10.00	7.51	10.95	3.75	1.09
3	58.68	29.03	100	21.54	14.29	19.66	13.25	10.34	3.51	1.29
4	60.63	32.47	23.52	100	19.97	13.65	17.05	10.02	8.92	0.20
5	48.22	31.25	18.57	23.77	100	11.11	11.49	7.24	4.88	0.00
6	80.47	21.18	33.47	21.29	14.56	100	12.23	8.92	4.86	1.53
7	65.26	18.92	26.83	31.63	17.91	14.55	100	8.65	3.66	0.03
8	55.62	30.11	22.86	20.28	12.32	11.58	9.43	100	7.39	1.07
9	78.46	38.23	28.77	67.00	30.76	23.41	14.82	27.40	100	0.12
10	99.19	14.40	13.75	1.94	0.00	9.55	0.16	5.18	0.16	100

Table 4: Probabilities (%) of possessing at least features k_1 and k_2 given that k_1 is active, i.e., $\left(\sum_{n=1}^{N} z_{nk_1} z_{nk_2}\right) / \left(\sum_{n=1}^{N} z_{nk_1}\right)$.

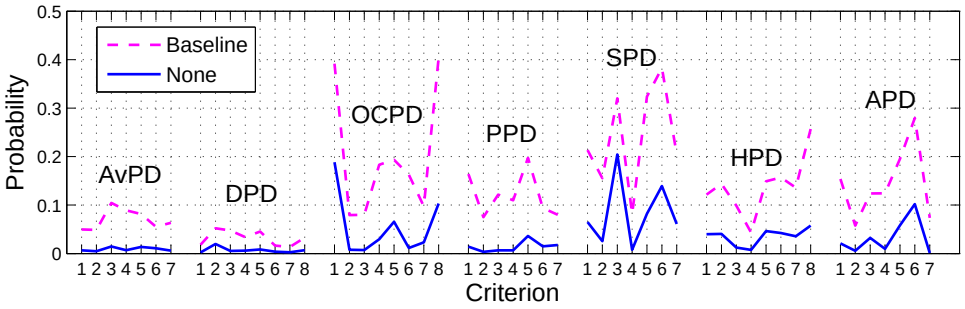


Figure 7: Probability of meeting each criterion. The probabilities when no latent feature is active (solid curve) have been obtained using the matrices $\mathbf{B}^d_{\text{MAP}}$, while the baseline (dashed curve) has been obtained taking into account the 43,093 subjects in the database.
(AvPD=Avoidant PD, DPD=Dependent PD, OCPD=Obsessive-compulsive PD, PPD=Paranoid PD, SPD=Schizoid PD, HPD=Histrionic PD, APD=Antisocial PD)

Figure 7). So, if the ratio is above one, it means that the feature increases the probability of meeting that criterion with respect to the general population. In all these plots, we also show the probability ratio between not having any active feature and the general population, which serves as a reference for a low probability of fulfilling a criterion. Note that the scale on the vertical axis may be different through all the figures for a better display. In Figure 8, we can see that only the criteria for one of the personality disorders is systematically above one, when one feature is active, except for Feature 7 that increases the probability for both avoidant and dependent. In the figure, we can also notice that when one feature is active

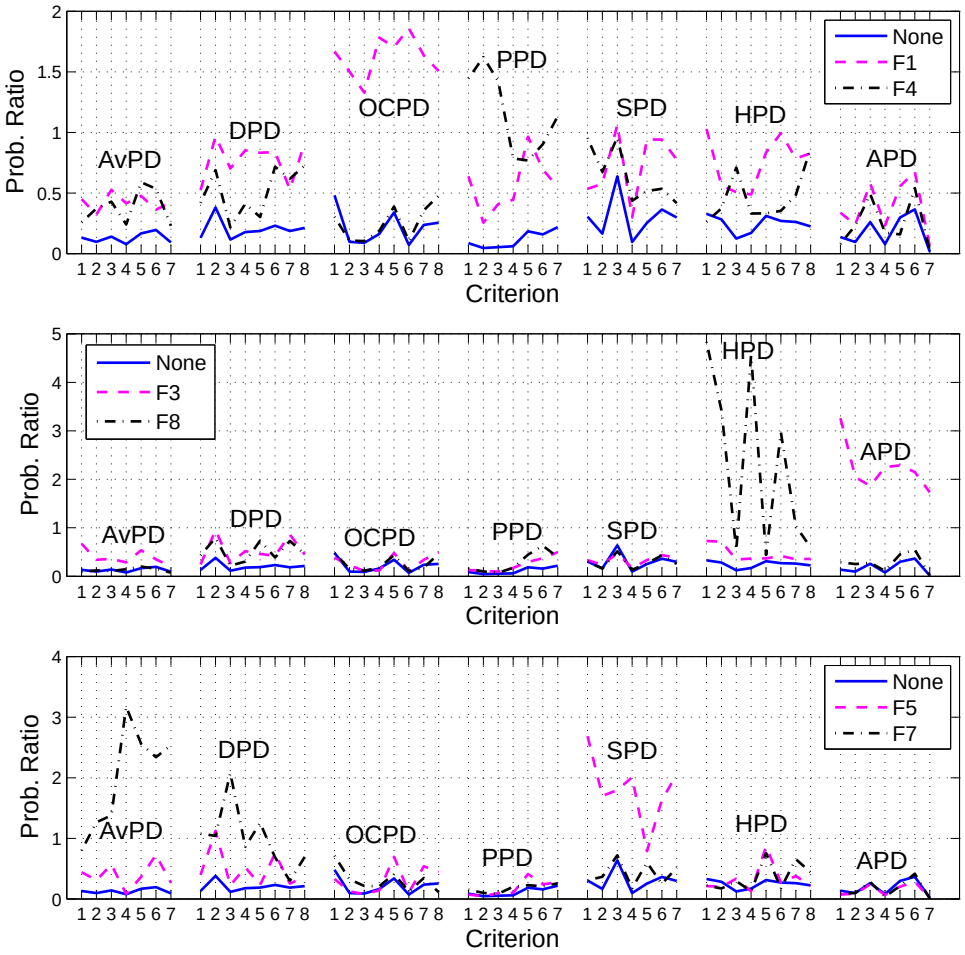


Figure 8: Probability ratio of meeting each criterion, with respect to the baseline. These probabilities have been obtained using the matrices $\mathbf{B}^d_{\text{MAP}}$, when none or a single feature is active (the legend shows the active latent features).

the probability of the criteria for the other disorders is above the probability for the subjects that do not have any active feature, although lower than the general population (above the solid line and below one). It partially shows the comorbidity pattern for each personality disorder. For example, Feature 1, besides increasing the probability of meeting the criteria for OCPD, also increases the probability of meeting criterion 3 for schizoid and criterion 1 for histrionic. It is also important to point out that Feature 8 increases significantly the probability of meeting criteria 1, 2, 4 and 6 for histrionic (and mildly for criterion 7), but it does not affect criteria 3, 5 and 8, although the probability of meeting these criteria are increased by Feature 4 (paranoid) and Feature 5 (schizoid). In a way, it indicates that criteria 3 and 8 are more related to paranoid disorder and criterion 5 to schizoid disorder.

As seen in Figure 9, Features 2 and 6 mainly reduce the probability of meeting the criteria for dependent PD. Feature 2 also reduces criteria 4-7 for avoidant and mildly increases

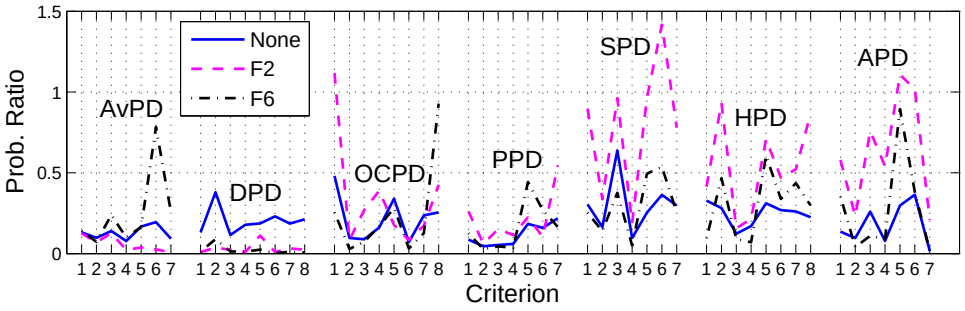


Figure 9: Probability ratio of meeting each criterion, with respect to the baseline. These probabilities have been obtained using the matrices $\mathbf{B}^d_{\text{MAP}}$, when none or a single feature is active (the legend shows the active latent features).

criterion 1 for OCPD, criterion 6 for schizoid and criteria 5 and 6 for antisocial. Feature 6 also reduces some criteria below the probability for the subjects with no active features. But for most of the criteria the probability ratio moves between one and the ratio for the subjects with no active feature. When these features appear by themselves, the subjects might be similar to the subjects without any active feature, they become relevant when they appear together with other features. These features are less likely to be isolated features than the previous ones, as reported in Table 2. For example, Feature 2 appears frequently with Features 1, 3, 4 and 5, as shown in Table 4, and the probability ratios are plotted in Figure 10 and compared to the probability ratio when each feature is not accompanied by Feature 2. We can see that when we add Feature 2 to Feature 1, the comorbidity pattern changes significantly and it results in subjects with higher probabilities of meeting the criteria for every other disorder except avoidant and dependent. Additionally, when we add Feature 2 to Feature 5, we can see that meeting the criteria for schizoid is even more probable, together with criterion 5 for histrionic.

Either Feature 1 or Features 1 and 3 typically accompany Feature 6, and Feature 6 is seldom seen by itself (see Tables 2 and 5). In Figure 11, we show the probability ratio when Feature 1 is active and when Features 1 and 3 are active, as reference, and when we add Feature 6 to them. Adding Feature 6 mainly reduces the probability of meeting the criteria for dependent. It is also relevant to point out that Features 1 and 3 increase the probability of meeting the criteria 5 and 6 for paranoid, while Feature 4 mainly increased the probability of meeting the criteria 1-4 for paranoid personality disorder, as shown in Figure 8.

Feature 9 is similar to Feature 7, as it captures an increase in the probability of meeting the criteria for avoidant and dependent, but it never appears isolated and most times it appears together with Features 1 and 4.

Feature 10 never appears isolated and it mainly appears only with Feature 1. This feature by itself only indicates that the probability of all the criteria should be much lower than the subjects with no active features, except for antisocial, which behaves as the subjects with no active features. When we add Feature 1 to Feature 10, we get that the probability of meeting the criteria for OCDP goes to that of the subject with no active features, as can

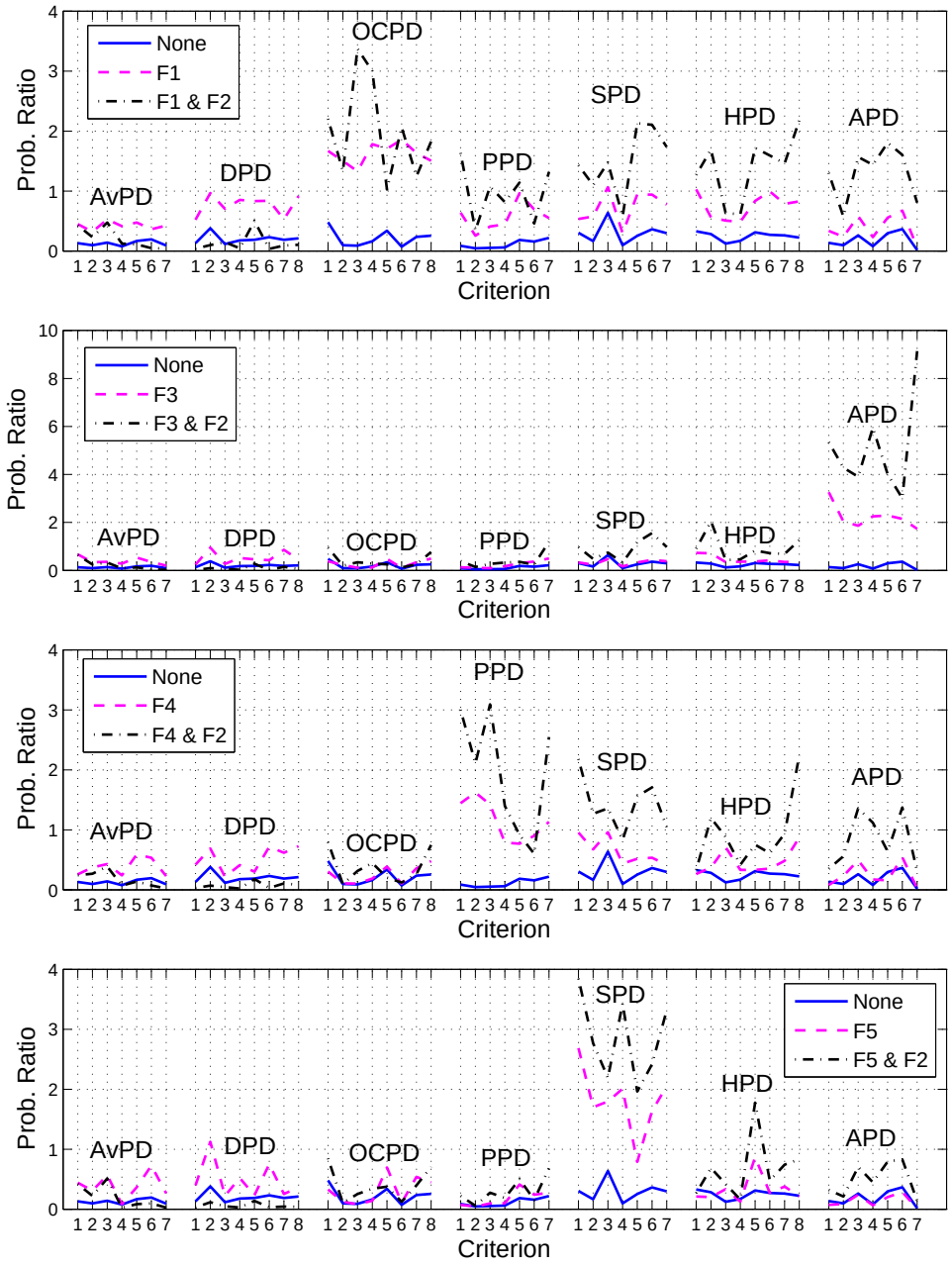


Figure 10: Probability ratio of meeting each criterion, with respect to the baseline. These probabilities have been obtained using the matrices $\mathbf{B}^d_{\text{MAP}}$, when none, a single or two features are active (the legend shows the active latent features).

be seen in Figure 12. For us this is a spurious feature that is equivalent to not having any active feature and that the variational algorithm has not been able to eliminate. This is always a risk when working with flexible models, like BNP, in which a spurious component

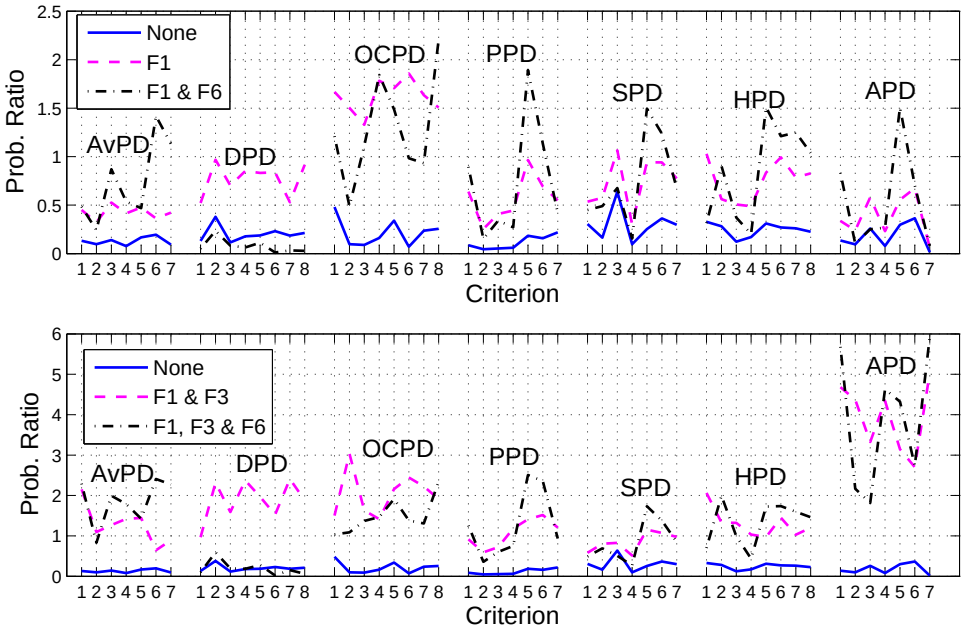


Figure 11: Probability ratio of meeting each criterion, with respect to the baseline. These probabilities have been obtained using the matrices $\mathbf{B}^d_{\text{MAP}}$, when none, a single or several features are active (the legend shows the active latent features).

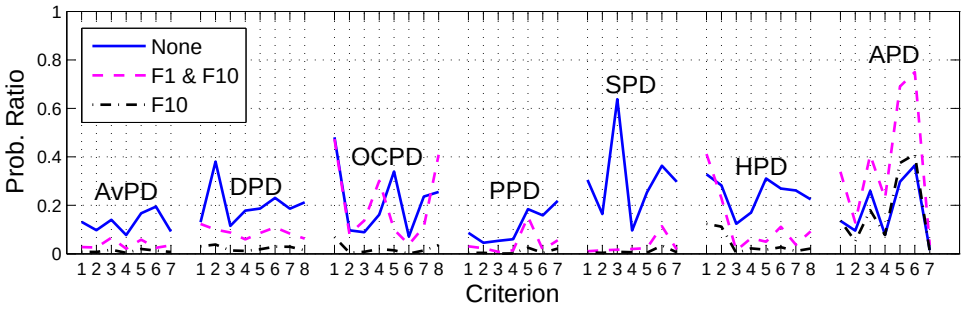


Figure 12: Probability ratio of meeting each criterion, with respect to the baseline. These probabilities have been obtained using the matrices $\mathbf{B}^d_{\text{MAP}}$, when none, a single or two features are active (the legend shows the active latent features).

might appear when it should not. These components can be eliminated by common sense in most cases or by further analysis by experts (psychiatric experts in our case). But it can also indicate an unknown component that can point towards a new research direction previously unknown, which is one of the attractive features of using generative models.

Besides the comorbidity patterns shown by the individual features that we have already reported, we can also see that almost all the features are positively correlated. In Table 3, we

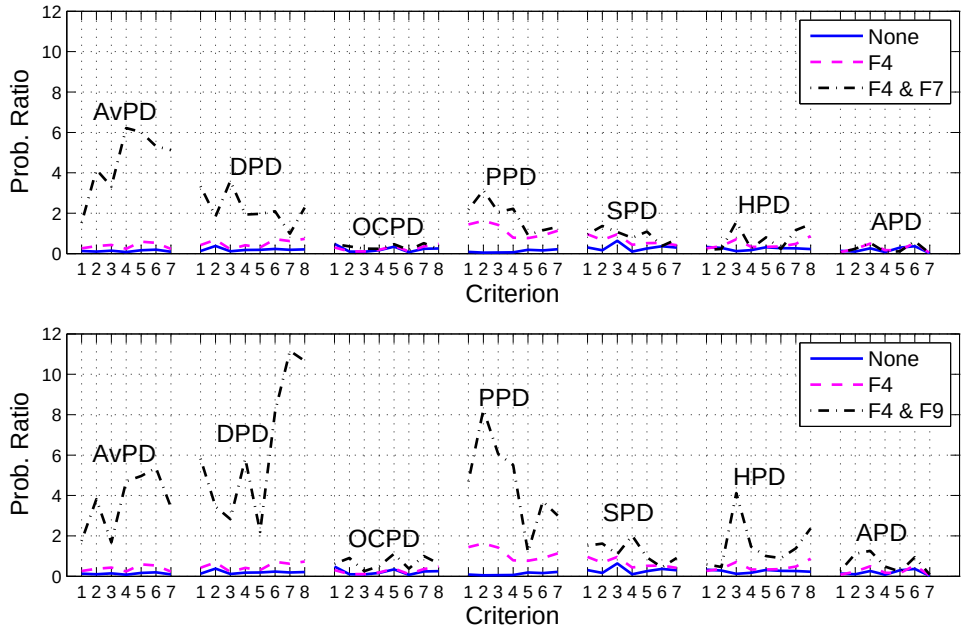


Figure 13: Probability ratio of meeting each criterion, with respect to the baseline. These probabilities have been obtained using the matrices $\mathbf{B}^d_{\text{MAP}}$, when none, a single or two features are active (the legend shows the active latent features).

show the probability that any two features appear together (upper triangular sub-matrix) and the joint probability that we should observe if the features were independent (lower triangular sub-matrix). Ignoring Feature 10, all of the other features are positively correlated, except Features 2 and 7 and Features 8 and 5 that seem uncorrelated (the differences are not statistically significant). Most of the features are strongly correlated and the differences in Table 3 correspond to several standard deviations higher (between 3 and 42) that we should expect from independent random observations. For example, the correlation between Features 4 and 9 and Features 4 and 7 is quite high and both show subjects with higher probability of meeting the criteria for avoidant, dependent and paranoid. The difference between Features 7 and 9 is given by the criteria 1-4 for paranoid PD, that are significantly increased by Feature 9 and slightly by Feature 7, as it can be seen in Figure 13. Finally, it is worth mentioning that Feature 4 (paranoid) is the most highly correlated feature with all the others, so we can say that anyone suffering from paranoid PD has a higher comorbidity with any other personality disorder.

6. Conclusions

In this paper, we have proposed a new model that combines the IBP with discrete observations using the multinomial-logit distribution. We have used the Laplace approximation to integrate out the weighting factors, which allows us to efficiently run the Gibbs sampler.

	# Occurrences	Features									
		1	2	3	4	5	6	7	8	9	10
1	15185	0	0	0	0	0	0	0	0	0	0
2	5811	1	0	0	0	0	0	0	0	0	0
3	1561	0	1	0	0	0	0	0	0	0	0
4	1389	1	1	0	0	0	0	0	0	0	0
5	1021	1	0	1	0	0	0	0	0	0	0
6	977	0	0	0	0	1	0	0	0	0	0
7	958	1	0	0	0	0	1	0	0	0	0
8	956	0	0	1	0	0	0	0	0	0	0
9	946	1	0	0	1	0	0	0	0	0	0
10	687	1	0	0	0	1	0	0	0	0	0
11	576	0	0	0	1	0	0	0	0	0	0
12	553	1	0	0	0	0	0	1	0	0	0
13	495	0	1	0	0	1	0	0	0	0	0
14	486	1	0	0	0	0	0	0	1	0	0
15	460	0	0	0	0	0	0	0	1	0	0
16	451	0	1	1	0	0	0	0	0	0	0
17	438	1	0	0	0	0	0	0	0	0	1
18	414	1	0	1	0	0	1	0	0	0	0
19	385	0	1	0	1	0	0	0	0	0	0
20	370	1	1	0	1	0	0	0	0	0	0

Table 5: List of the 20 most common feature patterns.

We have also derived a variational inference algorithm, which allows dealing with larger databases and provides accurate results.

We have applied our model to the NESARC database to find out the hidden features that characterize the psychiatric disorders. First, we have used the Gibbs sampler to extract the latent structure behind 20 of the most common psychiatric disorders. As a result, we have found that the comorbidity patterns of these psychiatric disorders can be described by only three latent features, which mainly model the internalizing disorders, the externalizing disorders, and a general psychopathology factor. Additionally, we have applied the variational inference algorithm to analyze the relation among the 52 criteria defined by the psychiatrists to diagnose each of the seven personality disorders (that is, externalizing disorders). We have obtained that for most of the disorders, a latent feature appears to model all the criteria that characterize that particular disorder. In this experiment, we have also seen that avoidant and dependent PDs are jointly modeled by four features, and that paranoid disorder is the most highly correlated PD with all the others.

Acknowledgments

Francisco J. R. Ruiz is supported by an FPU fellowship from the Spanish Ministry of Education (AP2010-5333), Isabel Valera is supported by the *Plan Regional-Programas I+D* of *Comunidad de Madrid* (AGES-CM S2010/BMD-2422), Fernando Pérez-Cruz has been partially supported by a Salvador de Madariaga grant, and Carlos Blanco acknowledges NIH grants (DA019606 and DA023200) and the New York State Psychiatric Institute for their support. The authors also acknowledge the support of *Ministerio de Ciencia e Innovación* of Spain (projects DEIPRO TEC2009-14504-C02-00, ALCIT TEC2012-38800-C03-01, and program Consolider-Ingenio 2010 CSD2008-00010 COMONSENS). This work was also supported by the European Union 7th Framework Programme through the Marie Curie Initial Training Network "Machine Learning for Personalized Medicine" MLPM2012, Grant No. 316861.

Appendix A. Laplace Approximation Details

In this section we provide the necessary details for the implementation of the Laplace approximation proposed in Section 3.1. The expression in (5) can be rewritten as

$$f(\mathbf{B}^d) = \operatorname{trace}\left\{\mathbf{M}^{d\top}\mathbf{B}^d\right\} - \sum_{n=1}^{N}\log\left(\sum_{r=1}^{R}\exp(\mathbf{z}_{n\bullet}\mathbf{b}^d_{\bullet r})\right) - \frac{1}{2\sigma_B^2}\operatorname{trace}\left\{\mathbf{B}^{d\top}\mathbf{B}^d\right\} - \frac{R(K+1)}{2}\log(2\pi\sigma_B^2),$$

where $(\mathbf{M}^d)_{kr}$ counts the number of data points for which $x_{nd} = r$ and $z_{nk} = 1$, namely, $(\mathbf{M}^d)_{kr} = \sum_{n=1}^{N}\delta(x_{nd} = r)z_{nk}$, where $\delta(\cdot)$ is the Kronecker delta function. By definition, $(\mathbf{M}^d)_{0r} = \sum_{n=1}^{N}\delta(x_{nd} = r)$.

By defining $(\boldsymbol{\rho}^d)_{kr} = \sum_{n=1}^{N} z_{nk}\pi_{nd}^r$, the gradient of $f(\mathbf{B}^d)$ can be derived as

$$\nabla f = \mathbf{M}^d - \boldsymbol{\rho}^d - \frac{1}{\sigma_B^2}\mathbf{B}^d.$$

To compute the Hessian, it is easier to define the gradient ∇f as a vector, instead of a matrix, and hence we stack the columns of $\mathbf{B}^d$ into $\boldsymbol{\beta}^d$, i.e., $\boldsymbol{\beta}^d = \mathbf{B}^d(:)$ for avid Matlab users. The Hessian matrix can now be readily computed taking the derivatives of the gradient, yielding

$$\begin{aligned}\nabla\nabla f &= -\frac{1}{\sigma_B^2}\mathbf{I}_{R(K+1)} + \nabla\nabla\log p(\mathbf{x}_{\bullet d}|\boldsymbol{\beta}^d, \mathbf{Z}) \\ &= -\frac{1}{\sigma_B^2}\mathbf{I}_{R(K+1)} - \sum_{n=1}^{N}\left(\operatorname{diag}(\boldsymbol{\pi}_{nd}) - (\boldsymbol{\pi}_{nd})^\top\boldsymbol{\pi}_{nd}\right)\otimes(\mathbf{z}_{n\bullet}^\top\mathbf{z}_{n\bullet}),\end{aligned}$$

where $\operatorname{diag}(\boldsymbol{\pi}_{nd})$ is a diagonal matrix with the values of the vector $\boldsymbol{\pi}_{nd} = \left[\pi_{nd}^1, \pi_{nd}^2, \ldots, \pi_{nd}^R\right]$ as its diagonal elements.

Finally, note that, since $p(\mathbf{x}_{\bullet d}|\boldsymbol{\beta}^d, \mathbf{Z})$ is a log-concave function of $\boldsymbol{\beta}^d$ (Boyd and Vandenberghe, 2004, p. 87), $-\nabla\nabla f$ is a positive definite matrix, which guarantees that the maximum of $f(\boldsymbol{\beta}^d)$ is unique.

Appendix B. Lower Bound Derivation

In this section we derive the lower bound $\mathcal{L}(\mathcal{H}, \mathcal{H}_q)$ on the evidence $p(\mathbf{X}|\mathcal{H})$. From Eq. (9),

$$\begin{aligned}\log p(\mathbf{X}|\mathcal{H}) &= \mathbb{E}_q\left[\log p(\Psi, \mathbf{X}|\mathcal{H})\right] + H[q] + D_{KL}(q||p)\\ &\geqslant \mathbb{E}_q\left[\log p(\Psi, \mathbf{X}|\mathcal{H})\right] + H[q].\end{aligned}$$

The expectation $\mathbb{E}_q\left[\log p(\Psi, \mathbf{X}|\mathcal{H})\right]$ can be derived as

$$\begin{aligned}\mathbb{E}_q\left[\log p(\Psi, \mathbf{X}|\mathcal{H})\right] = &\sum_{k=1}^{K} \underbrace{\mathbb{E}_q\left[\log p(v_k|\alpha)\right]}_{1} + \sum_{d=1}^{D}\sum_{k=1}^{K} \underbrace{\mathbb{E}_q\left[\log p(\mathbf{b}_{k\bullet}^d|\sigma_B^2)\right]}_{2} + \sum_{d=1}^{D} \underbrace{\mathbb{E}_q\left[\log p(\mathbf{b}_0^d|\sigma_B^2)\right]}_{3}\\ &+ \sum_{k=1}^{K}\sum_{n=1}^{N} \underbrace{\mathbb{E}_q\left[\log p(z_{nk}|\{v_i\}_{i=1}^k)\right]}_{4} + \sum_{n=1}^{N}\sum_{d=1}^{D} \underbrace{\mathbb{E}_q\left[\log p(x_{nd}|\mathbf{z}_{n\bullet}, \mathbf{B}^d, \mathbf{b}_0^d)\right]}_{5},\end{aligned} \tag{11}$$

where each term can be computed as shown below:

1. For the Beta distribution over v_k,

$$\mathbb{E}_q\left[\log p(v_k|\alpha)\right] = \log(\alpha) + (\alpha - 1)\left[\psi(\tau_{k1}) - \psi(\tau_{k1} + \tau_{k2})\right].$$

2. For the Gaussian distribution over vectors $\mathbf{b}_{k\bullet}^d$,

$$\mathbb{E}_q\left[\log p(\mathbf{b}_{k\bullet}^d|\sigma_B^2)\right] = -\frac{R}{2}\log(2\pi\sigma_B^2) - \frac{1}{2\sigma_B^2}\left(\sum_{r=1}^{R}(\phi_{kr}^d)^2 + \sum_{r=1}^{R}(\sigma_{kr}^d)^2\right).$$

3. For the Gaussian distribution over $\mathbf{b}_0^d$,

$$\mathbb{E}_q\left[\log p(\mathbf{b}_0^d|\sigma_B^2)\right] = -\frac{R}{2}\log(2\pi\sigma_B^2) - \frac{1}{2\sigma_B^2}\left(\sum_{r=1}^{R}(\phi_{0r}^d)^2 + \sum_{r=1}^{R}(\sigma_{0r}^d)^2\right).$$

4. For the feature assignments, which are Bernoulli distributed given the feature probabilities, we have

$$\begin{aligned}\mathbb{E}_q\left[\log p(z_{nk}|\{v_i\}_{i=1}^k)\right] =&(1-\nu_{nk})\mathbb{E}_q\left[\log\left(1 - \prod_{i=1}^{k} v_i\right)\right]\\ &+ \nu_{nk}\sum_{i=1}^{k}\left[\psi(\tau_{i1}) - \psi(\tau_{i1} + \tau_{i2})\right],\end{aligned}$$

where the expectation $\mathbb{E}_q\left[\log\left(1-\prod_{i=1}^{k} v_i\right)\right]$ has no closed-form solution. We can instead lower bound it by using the multinomial approach (Doshi-Velez et al., 2009). Under this approach, we introduce an auxiliary multinomial distribution $\boldsymbol{\lambda}_k = [\lambda_{k1}, \ldots, \lambda_{kk}]$ in the expectation and apply Jensen's inequality, yielding

$$\mathbb{E}_q\left[\log\left(1-\prod_{i=1}^{k} v_i\right)\right] \geq \sum_{m=1}^{k} \lambda_{km}\psi(\tau_{m2}) + \sum_{m=1}^{k-1}\left(\sum_{n=m+1}^{k}\lambda_{kn}\right)\psi(\tau_{m1}) - \sum_{m=1}^{k}\left(\sum_{n=m}^{k}\lambda_{kn}\right)\psi(\tau_{m1}+\tau_{m2}) - \sum_{m=1}^{k}\lambda_{km}\log(\lambda_{km}),$$

which holds for any distribution represented by the probabilities $\lambda_{k1}, \ldots, \lambda_{kk}$, for $1 \leq k \leq K$. Then,

$$\begin{aligned}\mathbb{E}_q\left[\log p(z_{nk}|\{v_i\}_{i=1}^{k})\right] \geq (1-\nu_{nk})\Bigg[&\sum_{m=1}^{k} \lambda_{km}\psi(\tau_{m2}) + \sum_{m=1}^{k-1}\left(\sum_{n=m+1}^{k}\lambda_{kn}\right)\psi(\tau_{m1}) \\ &- \sum_{m=1}^{k}\left(\sum_{n=m}^{k}\lambda_{kn}\right)\psi(\tau_{m1}+\tau_{m2}) - \sum_{m=1}^{k}\lambda_{km}\log(\lambda_{km})\Bigg] \\ &+ \nu_{nk}\sum_{i=1}^{k}\left[\psi(\tau_{i1}) - \psi(\tau_{i1}+\tau_{i2})\right].\end{aligned}$$

5. For the likelihood term, we can write

$$\mathbb{E}_q\left[\log p(x_{nd}|\mathbf{z}_{n\bullet}, \mathbf{B}^d, \mathbf{b}_0^d)\right] = \phi_{0x_{nd}}^d + \sum_{k=1}^{K}\nu_{nk}\phi_{kx_{nd}}^d - \mathbb{E}_q\left[\log\left(\sum_{r=1}^{R}\exp(\mathbf{z}_{n\bullet}\mathbf{b}_{\bullet r}^d + b_{0r}^d)\right)\right],$$

where the logarithm can be upper bounded by its first-order Taylor series expansion around the auxiliary variable ξ_{nd}^{-1} (for $n = 1, \ldots, N$ and $d = 1, \ldots, D$) (Blei and Lafferty, 2007; Bouchard, 2007), yielding

$$\log\left(\sum_{r=1}^{R}\exp(\mathbf{z}_{n\bullet}\mathbf{b}_{\bullet r}^d + b_{0r}^d)\right) \leq \xi_{nd}\left(\sum_{r=1}^{R}\exp(\mathbf{z}_{n\bullet}\mathbf{b}_{\bullet r}^d + b_{0r}^d)\right) - \log(\xi_{nd}) - 1.$$

The main advantage of this bound lies on the fact that it allows us to compute the expectation of the bound for the Gaussian distribution, since it involves the moment generating functions of the distributions $q(\mathbf{b}_{\bullet r}^d)$ and $q(b_{0r}^d)$. Then, we can lower bound the likelihood term as

$$\begin{aligned}\mathbb{E}_q\left[\log p(x_{nd}|\mathbf{z}_{n\bullet}, \mathbf{B}^d, \mathbf{b}_0^d)\right] \geq{}& \phi_{0x_{nd}}^d + \sum_{k=1}^{K}\nu_{nk}\phi_{kx_{nd}}^d + \log(\xi_{nd}) + 1 \\ &- \xi_{nd}\sum_{r=1}^{R}\left[\exp\left(\phi_{0r}^d + \frac{1}{2}(\sigma_{0r}^d)^2\right)\prod_{k=1}^{K}\left(1-\nu_{nk}+\nu_{nk}\exp\left(\phi_{kr}^d + \frac{1}{2}(\sigma_{kr}^d)^2\right)\right)\right].\end{aligned}$$

Substituting the previous results in (11), we obtain

$$
\begin{aligned}
\mathbb{E}_q\left[\log p(\Psi, \mathbf{X}|\mathcal{H})\right] \geq & \sum_{k=1}^{K}\left[\log(\alpha) + (\alpha-1)\left(\psi(\tau_{k1}) - \psi(\tau_{k1}+\tau_{k2})\right)\right] \\
& - \frac{R(K+1)D}{2}\log(2\pi\sigma_B^2) - \frac{1}{2\sigma_B^2}\sum_{k=0}^{K}\sum_{d=1}^{D}\sum_{r=1}^{R}\left((\phi_{kr}^d)^2 + \left(\sigma_{kr}^d\right)^2\right) \\
& + \sum_{n=1}^{N}\sum_{k=1}^{K}\Bigg[\nu_{nk}\sum_{i=1}^{k}\left[\psi(\tau_{i1}) - \psi(\tau_{i1}+\tau_{i2})\right] \\
& \qquad + (1-\nu_{nk})\left(\sum_{m=1}^{k}\lambda_{km}\psi(\tau_{m2}) + \sum_{m=1}^{k-1}\left(\sum_{n=m+1}^{k}\lambda_{kn}\right)\psi(\tau_{m1})\right. \\
& \qquad \left. - \sum_{m=1}^{k}\left(\sum_{n=m}^{k}\lambda_{kn}\right)\psi(\tau_{m1}+\tau_{m2}) - \sum_{m=1}^{k}\lambda_{km}\log(\lambda_{km})\right)\Bigg] \\
& + \sum_{n=1}^{N}\sum_{d=1}^{D}\Bigg[\phi_{0x_{nd}}^d + \sum_{k=1}^{K}\nu_{nk}\phi_{kx_{nd}}^d + \log(\xi_{nd}) + 1 \\
& - \xi_{nd}\sum_{r=1}^{R}\left[\exp\left(\phi_{0r}^d + \frac{1}{2}(\sigma_{0r}^d)^2\right)\prod_{k=1}^{K}\left(1 - \nu_{nk} + \nu_{nk}\exp\left(\phi_{kr}^d + \frac{1}{2}\left(\sigma_{kr}^d\right)^2\right)\right)\right]\Bigg].
\end{aligned}
$$

Additionally, the entropy of the distribution $q(\Psi)$ is given by

$$
\begin{aligned}
H[q] &= \mathbb{E}_q\left[\log q(\Psi)\right] \\
&= \sum_{k=1}^{K}\mathbb{E}_q\left[\log q(v_k|\tau_{k1},\tau_{k2})\right] + \sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{k=0}^{K}\mathbb{E}_q\left[\log q(b_{kr}^d|\phi_{kr}^d,(\sigma_{kr}^d)^2)\right] + \sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}_q\left[\log q(z_{nk}|\nu_{nk})\right] \\
&= \sum_{k=1}^{K}\left[\log\left(\frac{\Gamma(\tau_{k1})\Gamma(\tau_{k2})}{\Gamma(\tau_{k1}+\tau_{k2})}\right) - (\tau_{k1}-1)\psi(\tau_{k1}) - (\tau_{k2}-1)\psi(\tau_{k2}) + (\tau_{k1}+\tau_{k2}-2)\psi(\tau_{k1}+\tau_{k2})\right] \\
&\quad + \sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{k=0}^{K}\frac{1}{2}\log(2\pi e(\sigma_{kr}^d)^2) + \sum_{n=1}^{N}\sum_{k=1}^{K}\left[-\nu_{nk}\log(\nu_{nk}) - (1-\nu_{nk})\log(1-\nu_{nk})\right].
\end{aligned}
$$

Finally, we obtain the lower bound on the evidence $p(\mathbf{X}|\mathcal{H})$ as

$$
\begin{aligned}
&\log p(\mathbf{X}|\mathcal{H}) \geq \mathbb{E}_q\left[\log p(\Psi, \mathbf{X}|\mathcal{H})\right] + H[q] \\
&\geq \sum_{k=1}^{K}\left[\log(\alpha) + (\alpha-1)\left(\psi(\tau_{k1}) - \psi(\tau_{k1}+\tau_{k2})\right)\right] \\
&\quad - \frac{R(K+1)D}{2}\log(2\pi\sigma_B^2) - \frac{1}{2\sigma_B^2}\sum_{k=0}^{K}\sum_{d=1}^{D}\sum_{r=1}^{R}\left((\phi_{kr}^d)^2 + \left(\sigma_{kr}^d\right)^2\right) \\
&\quad + \sum_{n=1}^{N}\sum_{k=1}^{K}\Bigg[\nu_{nk}\sum_{i=1}^{k}\left[\psi(\tau_{i1}) - \psi(\tau_{i1}+\tau_{i2})\right] \\
&\qquad\qquad + (1-\nu_{nk})\left(\sum_{m=1}^{k}\lambda_{km}\psi(\tau_{m2}) + \sum_{m=1}^{k-1}\left(\sum_{n=m+1}^{k}\lambda_{kn}\right)\psi(\tau_{m1})\right. \\
&\qquad\qquad \left. - \sum_{m=1}^{k}\left(\sum_{n=m}^{k}\lambda_{kn}\right)\psi(\tau_{m1}+\tau_{m2}) - \sum_{m=1}^{k}\lambda_{km}\log(\lambda_{km})\right)\Bigg] \\
&\quad + \sum_{n=1}^{N}\sum_{d=1}^{D}\Bigg[\phi_{0x_{nd}}^d + \sum_{k=1}^{K}\nu_{nk}\phi_{kx_{nd}}^d + \log(\xi_{nd}) + 1 \\
&\quad - \xi_{nd}\sum_{r=1}^{R}\left[\exp\left(\phi_{0r}^d + \frac{1}{2}(\sigma_{0r}^d)^2\right)\prod_{k=1}^{K}\left(1-\nu_{nk}+\nu_{nk}\exp\left(\phi_{kr}^d + \frac{1}{2}\left(\sigma_{kr}^d\right)^2\right)\right)\right]\Bigg] \\
&\quad + \sum_{k=1}^{K}\left[\log\left(\frac{\Gamma(\tau_{k1})\Gamma(\tau_{k2})}{\Gamma(\tau_{k1}+\tau_{k2})}\right) - (\tau_{k1}-1)\psi(\tau_{k1}) - (\tau_{k2}-1)\psi(\tau_{k2}) + (\tau_{k1}+\tau_{k2}-2)\psi(\tau_{k1}+\tau_{k2})\right] \\
&\quad + \sum_{d=1}^{D}\sum_{r=1}^{R}\sum_{k=0}^{K}\frac{1}{2}\log(2\pi e(\sigma_{kr}^d)^2) + \sum_{n=1}^{N}\sum_{k=1}^{K}\left[-\nu_{nk}\log(\nu_{nk}) - (1-\nu_{nk})\log(1-\nu_{nk})\right] \\
&= \mathcal{L}(\mathcal{H}, \mathcal{H}_q),
\end{aligned}
$$

where $\mathcal{H}_q = \{\tau_{k1}, \tau_{k2}, \lambda_{km}, \xi_{nd}, \nu_{nk}, \phi_{kr}^d, \phi_{0r}^d, (\sigma_{kr}^d)^2, (\sigma_{0r}^d)^2\}$ (for $k = 1, \ldots, K$, $m = 1, \ldots, k$, $d = 1, \ldots, D$, and $n = 1, \ldots, N$) represents the set of the variational parameters.

Appendix C. Derivatives for Newton's Method

- For the parameters of the Gaussian distribution $q(b_{kr}^d|\phi_{kr}^d, (\sigma_{kr}^d)^2)$ for $k = 1, \ldots, K$,

$$
\begin{aligned}
\frac{\partial}{\partial\phi_{kr}^d}\mathcal{L}(\mathcal{H}, \mathcal{H}_q) = -\frac{1}{\sigma_B^2}\phi_{kr}^d + \sum_{n=1}^{N}\Bigg[&\nu_{nk}\delta(x_{nd}=r) - \nu_{nk}\xi_{nd}\exp\left(\phi_{0r}^d + \frac{1}{2}(\sigma_{0r}^d)^2\right)\exp\left(\phi_{kr}^d + \frac{1}{2}(\sigma_{kr}^d)^2\right) \\
&\times\prod_{k'\neq k}\left(1-\nu_{nk'}+\nu_{nk'}\exp\left(\phi_{k'r}^d + \frac{1}{2}(\sigma_{k'r}^d)^2\right)\right)\Bigg].
\end{aligned}
$$

$$\frac{\partial^2}{\partial(\phi_{kr}^d)^2}\mathcal{L}(\mathcal{H},\mathcal{H}_q) = -\frac{1}{\sigma_B^2} - \sum_{n=1}^{N}\left[\nu_{nk}\xi_{nd}\exp\left(\phi_{0r}^d + \frac{1}{2}(\sigma_{0r}^d)^2\right)\exp\left(\phi_{kr}^d + \frac{1}{2}(\sigma_{kr}^d)^2\right)\right.$$
$$\left.\times\prod_{k'\neq k}\left(1-\nu_{nk'}+\nu_{nk'}\exp\left(\phi_{k'r}^d+\frac{1}{2}(\sigma_{k'r}^d)^2\right)\right)\right].$$

$$\frac{\partial}{\partial(\sigma_{kr}^d)^2}\mathcal{L}(\mathcal{H},\mathcal{H}_q) = -\frac{1}{2\sigma_B^2} + \frac{1}{2}(\sigma_{kr}^d)^{-2} - \frac{1}{2}\sum_{n=1}^{N}\left[\nu_{nk}\xi_{nd}\exp\left(\phi_{0r}^d + \frac{1}{2}(\sigma_{0r}^d)^2\right)\exp\left(\phi_{kr}^d + \frac{1}{2}(\sigma_{kr}^d)^2\right)\right.$$
$$\left.\times\prod_{k'\neq k}\left(1-\nu_{nk'}+\nu_{nk'}\exp\left(\phi_{k'r}^d+\frac{1}{2}(\sigma_{k'r}^d)^2\right)\right)\right].$$

$$\frac{\partial^2}{(\partial(\sigma_{kr}^d)^2)^2}\mathcal{L}(\mathcal{H},\mathcal{H}_q) = -\frac{1}{2}(\sigma_{kr}^d)^{-4} - \frac{1}{4}\sum_{n=1}^{N}\left[\nu_{nk}\xi_{nd}\exp\left(\phi_{0r}^d + \frac{1}{2}(\sigma_{0r}^d)^2\right)\exp\left(\phi_{kr}^d + \frac{1}{2}(\sigma_{kr}^d)^2\right)\right.$$
$$\left.\times\prod_{k'\neq k}\left(1-\nu_{nk'}+\nu_{nk'}\exp\left(\phi_{k'r}^d+\frac{1}{2}(\sigma_{k'r}^d)^2\right)\right)\right].$$

- For the parameters of the Gaussian distribution $q(b_{0r}^d|\phi_{0r}^d,(\sigma_{0r}^d)^2)$,

$$\frac{\partial}{\partial\phi_{0r}^d}\mathcal{L}(\mathcal{H},\mathcal{H}_q)$$
$$= -\frac{1}{\sigma_B^2}\phi_{0r}^d + \sum_{n=1}^{N}\left[\delta(x_{nd}=r) - \xi_{nd}\exp\left(\phi_{0r}^d+\frac{1}{2}(\sigma_{0r}^d)^2\right)\prod_{k=1}^{K}\left(1-\nu_{nk}+\nu_{nk}\exp\left(\phi_{kr}^d+\frac{1}{2}(\sigma_{kr}^d)^2\right)\right)\right].$$

$$\frac{\partial^2}{(\partial\phi_{0r}^d)^2}\mathcal{L}(\mathcal{H},\mathcal{H}_q)$$
$$= -\frac{1}{\sigma_B^2} - \sum_{n=1}^{N}\left[\xi_{nd}\exp\left(\phi_{0r}^d+\frac{1}{2}(\sigma_{0r}^d)^2\right)\prod_{k=1}^{K}\left(1-\nu_{nk}+\nu_{nk}\exp\left(\phi_{kr}^d+\frac{1}{2}(\sigma_{kr}^d)^2\right)\right)\right].$$

$$\frac{\partial}{\partial(\sigma_{0r}^d)^2}\mathcal{L}(\mathcal{H},\mathcal{H}_q)$$
$$= -\frac{1}{2\sigma_B^2} + \frac{1}{2}(\sigma_{0r}^d)^{-2} - \frac{1}{2}\sum_{n=1}^{N}\left[\xi_{nd}\exp\left(\phi_{0r}^d+\frac{1}{2}(\sigma_{0r}^d)^2\right)\prod_{k=1}^{K}\left(1-\nu_{nk}+\nu_{nk}\exp\left(\phi_{kr}^d+\frac{1}{2}(\sigma_{kr}^d)^2\right)\right)\right].$$

$$\frac{\partial^2}{(\partial(\sigma_{0r}^d)^2)^2}\mathcal{L}(\mathcal{H},\mathcal{H}_q)$$
$$= -\frac{1}{2}(\sigma_{0r}^d)^{-4} - \frac{1}{4}\sum_{n=1}^{N}\left[\xi_{nd}\exp\left(\phi_{0r}^d+\frac{1}{2}(\sigma_{0r}^d)^2\right)\prod_{k=1}^{K}\left(1-\nu_{nk}+\nu_{nk}\exp\left(\phi_{kr}^d+\frac{1}{2}(\sigma_{kr}^d)^2\right)\right)\right].$$

Appendix D. Correspondence Between Criteria and Questions in NESARC

Question Code	Personality disorder and criterion
S10Q1A1-S10Q1B7	Avoidant (1 question for each diagnostic criterion)
S10Q1A8-S10Q1B15	Dependent (1 question for each diagnostic criterion)
S10Q1A16-S10Q1B17	OCPD criterion 1
S10Q1A18-S10Q1B23	OCPD criteria 2-7
S10Q1A24-S10Q1B25	OCPD criterion 8
S10Q1A26-S10Q1B29	Paranoid criteria 1-4
S10Q1A30-S10Q1A31	Paranoid criterion 5
S10Q1A32-S10Q1B33	Paranoid criteria 6-7
S10Q1A45-S10Q1B46	Schizoid criterion 1
S10Q1A47-S10Q1B48	Schizoid criteria 2-3
S10Q1A50-S10Q1B50	Schizoid criterion 4
S10Q1A43-S10Q1B43	Schizoid criterion 5
S10Q1A51-S10Q1B52	Schizoid criterion 6
S10Q1A49-S10Q1B49 or S10Q1A53-S10Q1B53	Schizoid criterion 7
S10Q1A54-S10Q1B54 or S10Q1A56-S10Q1B56	Histrionic criterion 1
S10Q1A58-S10Q1B58 or S10Q1A60-S10Q1B60	Histrionic criterion 2
S10Q1A55-S10Q1B55	Histrionic criterion 3
S10Q1A61-S10Q1B61	Histrionic criterion 4
S10Q1A64-S10Q1B64	Histrionic criterion 5
S10Q1A59-S10Q1B59 or S10Q1A62-S10Q1B62	Histrionic criterion 6
S10Q1A63-S10Q1B63	Histrionic criterion 7
S10Q1A57-S10Q1B57	Histrionic criterion 8
S11Q1A20-S11Q1A25	Antisocial, criterion 1
S11Q1A11- S11Q1A13	Antisocial, criterion 2
S11Q1A8- S11Q1A10	Antisocial, criterion 3
S11Q1A17- S11Q1A18	Antisocial, criterion 4
S11Q1A26- S11Q1A33	Antisocial, criterion 4
S11Q1A14- S11Q1A16	Antisocial, criterion 5
S11Q1A6 and S11Q1A19	Antisocial, criterion 6
S11Q8A-B	Antisocial, criterion 7

Table 6: Correspondence between the criteria for each personality disorder and questions in NESARC.

References

M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, New York, 1972.

D. Aldous. Exchangeability and related topics. In *École d'été de Probabilités de Saint-Flour, XIII—1983*, pages 1–198. Springer, Berlin, 1985.

C. Blanco, R. F. Krueger, D. S. Hasin, S. M. Liu, S. Wang, B. T. Kerridge, T. Saha, and M. Olfson. Mapping common psychiatric disorders: Structure and predictive validity in the National Epidemiologic Survey on Alcohol and Related Conditions. *Journal of the American Medical Association Psychiatry*, 70(2):199–208, 2013.

D. M. Blei and J. D. Lafferty. A correlated topic model of Science. *Annals of Applied Statistics*, 1(1):17–35, August 2007.

G. Bouchard. Efficient bounds for the softmax and applications to approximate inference in hybrid models. *Advances in Neural Information Processing Systems (NIPS), Workshop on Approximate Inference in Hybrid Models*, 2007.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

F. Doshi-Velez, K. T. Miller, J. Van Gael, and Y. W. Teh. Variational inference for the Indian buffet process, 2009.

T. L. Griffiths and Z. Ghahramani. The Indian buffet process: an introduction and review. *Journal of Machine Learning Research*, 12:1185–1224, 2011.

D. A. Harville. *Matrix Algebra From a Statistician's Perspective*. Springer-Verlag, 1997.

S. Haykin. *Adaptive Filter Theory*. Prentice Hall, 2002.

M. I. Jordan. Hierarchical models, nested models and completely random measures. In M.-H. Chen, D. Dey, P. Mueller, D. Sun, and K. Ye, editors, *Frontiers of Statistical Decision Making and Bayesian Analysis: In Honor of James O. Berger*. Springer, New York, (NY), 2010.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, November 1999.

K. S. Kendler, C. A. Prescott, J. Myers, and M. C. Neale. The structure of genetic and environmental risk factors for common psychiatric and substance use disorders in men and women. *Archives of General Psychiatry*, 60(9):929–937, Sep 2003.

R. Kotov, C. J. Ruggero, R. F. Krueger, D. Watson, Q. Yuan, and M. Zimmerman. New dimensions in the quantitative classification of mental illness. *Archives of General Psychiatry*, 68(10):1003–1011, 2011.

R. F. Krueger. The structure of common mental disorders. *Archives of General Psychiatry*, 56(10):921–926, 1999.

F. J. R. Ruiz, I. Valera, C. Blanco, and F. Perez-Cruz. Bayesian nonparametric modeling of suicide attempts. *Advances in Neural Information Processing Systems (NIPS)*, 25: 1862–1870, 2012.

Y. W. Teh, D. Görür, and Z. Ghahramani. Stick-breaking construction for the Indian buffet process. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 11, 2007.

M. Titsias. The infinite gamma-Poisson feature model. *Advances in Neural Information Processing Systems (NIPS)*, 19, 2007.

J. M. Valderas, B. Starfield, B. Sibbald, C. Salisbury, and M. Roland. Defining comorbidity: implications for understanding health and health services. *Annals of Family Medicine*, 7 (4):357–363, 2009.

J. Van Gael, Y. W. Teh, and Z. Ghahramani. The infinite factorial hidden Markov model. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21, 2009.

W. A. Vollebergh, J. Ledema, R.V. Bijl, R. de Graaf, F. Smit, and J. Ormel. The structure and stability of common mental disorders: the NEMESIS study. *Archives of General Psychiatry*, 58(6):597–603, Jun 2001.

C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1342–1351, 1998.

S. Williamson, C. Wang, K. A. Heller, and D. M. Blei. The IBP compound Dirichlet process and its application to focused topic modeling. In *International Conference on Machine Learning (ICML)*, pages 1151–1158, 2010.

J. L. Wolff, B. Starfield, and G. Anderson. Prevalence, expenditures, and complications of multiple chronic conditions in the elderly. *Archives of Internal Medicine*, 162(20): 2269–2276, 2002.

M. A. Woodbury. The stability of out-input matrices. *Mathematical Reviews*, 1949.

Journal of Machine Learning Research 15 (2014) 1249-1280 Submitted 3/13; Revised 10/13; Published 4/14

Robust Near-Separable Nonnegative Matrix Factorization Using Linear Optimization

Nicolas Gillis* NICOLAS.GILLIS@UMONS.AC.BE
Department of Mathematics and Operational Research
Faculté Polytechnique, Université de Mons
Rue de Houdain 9, 7000 Mons, Belgium

Robert Luce† LUCE@MATH.TU-BERLIN.DE
Institut für Mathematik, MA 3-3
Technische Universität Berlin
Straße des 17. Juni 136 - 10623 Berlin, Germany

Editor: Gert Lanckriet

Abstract

Nonnegative matrix factorization (NMF) has been shown recently to be tractable under the *separability assumption*, under which all the columns of the input data matrix belong to the convex cone generated by only a few of these columns. Bittorf, Recht, Ré and Tropp ('Factoring nonnegative matrices with linear programs', NIPS 2012) proposed a linear programming (LP) model, referred to as Hottopixx, which is robust under any small perturbation of the input matrix. However, Hottopixx has two important drawbacks: (i) the input matrix has to be normalized, and (ii) the factorization rank has to be known in advance. In this paper, we generalize Hottopixx in order to resolve these two drawbacks, that is, we propose a new LP model which does not require normalization and detects the factorization rank automatically. Moreover, the new LP model is more flexible, significantly more tolerant to noise, and can easily be adapted to handle outliers and other noise models. Finally, we show on several synthetic data sets that it outperforms Hottopixx while competing favorably with two state-of-the-art methods.

Keywords: nonnegative matrix factorization, separability, linear programming, convex optimization, robustness to noise, pure-pixel assumption, hyperspectral unmixing

1. Introduction

Nonnegative matrix factorization (NMF) is a powerful dimensionality reduction technique as it automatically extracts sparse and meaningful features from a set of nonnegative data vectors: Given n nonnegative m-dimensional vectors gathered in a nonnegative matrix $M \in \mathbb{R}^{m\times n}_+$ and a factorization rank r, NMF computes two nonnegative matrices $W \in \mathbb{R}^{m\times r}_+$ and $H \in \mathbb{R}^{r\times n}_+$ such that $M \approx WH$. In this way, the columns of the matrix W form a basis for the columns of M since $M(:,j) \approx \sum_{k=1}^{r} W(:,k)H(k,j)$ for all j. Moreover, the nonnegativity constraint on the matrices W and H leads these basis elements to represent

∗. This work was carried out when NG was a postdoctoral researcher of the fonds de la recherche scientifique (F.R.S.-FNRS).
†. RL is supported by Deutsche Forschungsgemeinschaft, Cluster of Excellence "UniCat".

common localized features appearing in the data set as no cancellation can happen in the reconstruction of the original data. Unfortunately, NMF is NP-hard in general (Vavasis, 2009), and highly ill-posed; see Gillis (2012) and the references therein. However, if the input data matrix M is *r-separable*, that is, if it can be written as

$$M = W\,[I_r,\, H']\Pi,$$

where I_r is the r-by-r identity matrix, $H' \geq 0$ and Π is a permutation matrix, then the problem can be solved in polynomial time, even if some noise is added to the separable matrix M (Arora et al., 2012a). Algebraically, separability means that there exists a rank-r NMF $(W, H) \geq 0$ of M where each column of W is equal to some column of M. Geometrically, r-separability means that the cone generated by the columns of M has r extreme rays given by the columns of W. Equivalently, if the columns of M are normalized so that their entries sum to one, r-separability means that the convex hull generated by the columns of M has r vertices given by the columns of W; see, e.g., Kumar et al. (2013). The separability assumption is far from being artificial in several applications:

- In text mining, where each column of M corresponds to a word, separability means that, for each topic, there exists a word associated only with that topic; see Arora et al. (2012a,b).

- In hyperspectral imaging, where each column of M equals the spectral signature of a pixel, separability means that, for each constitutive material ("endmember") present in the image, there exists a pixel containing only that material. This assumption is referred to as the *pure-pixel assumption*, and is in general satisfied for high-resolution hyperspectral images; see Bioucas-Dias et al. (2012) and the references therein.

- In blind source separation, where each column of M is a signal measure at a given point in time, separability means that, for each source, there exists a point in time where only that source is active; see Chan et al. (2008); Chen et al. (2011) and the references therein.

Under the separability assumption, NMF reduces to identifying, among the columns of M, the columns of W allowing to reconstruct all columns of M. In fact, given W, the matrix H can be obtained by solving a convex optimization problem $\min_{H\geq 0}\|M - WH\|$.

In this paper, we consider the noisy variant of this problem, referred to as *near-separable NMF*:

> (Near-Separable NMF) *Given a noisy r-separable matrix $\tilde{M} = M + N$ with $M = WH = W[I_r, H']\Pi$ where W and H' are nonnegative matrices, Π is a permutation matrix and N is the noise, find a set $\mathcal{K}$ of r indices such that $\tilde{M}(:, \mathcal{K}) \approx W$.*

Several algorithms have been proposed to solve this problem (Arora et al., 2012a,b; Bittorf et al., 2012; Elhamifar et al., 2012; Esser et al., 2012; Gillis and Vavasis, 2014; Kumar et al., 2013). In this paper, our focus is on the linear programming (LP) model proposed by Bittorf et al. (2012) and referred to as Hottopixx. It is described in the next section.

Remark 1 (Nonnegativity of $\tilde{M}$) *In the formulation of near-separable NMF, the input data matrix $\tilde{M}$ is not necessarily nonnegative since there is no restriction on the noise N. In fact, we will only need to assume that the noise is bounded, but otherwise it is arbitrary; see Section 2.*

1.1 Notation

Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $x \in \mathbb{R}^m$ a vector. We use Matlab-style notation for indexing, for example, $A(i,j)$ denotes the entry of A in the i-th row and j-th column, while $A(:,j) \in \mathbb{R}^m$ denotes the j-th column of A. We use the following notation for various norms:

$$\|x\|_1 = \sum_{i=1}^{m} |x(i)|, \qquad \|A\|_1 = \max_{\|x\|_1\leq 1} \|Ax\|_1 = \max_j \|A(:,j)\|_1,$$
$$\|A\|_s = \sum_{i=1}^{m}\sum_{j=1}^{n} |A(i,j)|, \qquad \|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A(i,j)^2}.$$

1.2 Hottopixx, a Linear Programming Model for Near-Separable NMF

A matrix M is r-separable if and only if

$$\begin{aligned} M = WH &= W[I_r, H']\Pi = [W, WH']\Pi \\ &= [W, WH']\Pi \underbrace{\Pi^{-1}\begin{pmatrix} I_r & H' \\ 0_{(n-r)\times r} & 0_{(n-r)\times(n-r)} \end{pmatrix}\Pi}_{X^0\in\mathbb{R}_+^{n\times n}} = MX^0, \end{aligned} \tag{1}$$

for some permutation Π and some matrices $W, H' \geq 0$. The matrix X^0 is a n-by-n nonnegative matrix with $(n-r)$ zero rows such that $M = MX^0$. Assuming the entries of each column of M sum to one, the entries of each column of W and H' have sum to one as well. Based on these observations, Bittorf et al. (2012) proposed to solve the following optimization problem in order to approximately identify the columns of the matrix W among the columns of the matrix $\tilde{M} = M + N$ where N is the noise with $\|N\|_1 \leq \epsilon$:

$$\begin{aligned} \min_{X\in\mathbb{R}_+^{n\times n}} \quad & p^T \operatorname{diag}(X) \\ \text{such that} \quad & \|\tilde{M} - \tilde{M}X\|_1 \leq 2\epsilon, \\ & \operatorname{tr}(X) = r, \\ & X(i,i) \leq 1 \text{ for all } i, \\ & X(i,j) \leq X(i,i) \text{ for all } i,j, \end{aligned} \tag{2}$$

where p is any n-dimensional vector with distinct entries; see Algorithm 1 (in Bittorf et al., 2012, the authors use the notation $\|\cdot\|_{\infty,1}$ for what we denote by $\|\cdot\|_1$).

Intuitively, the LP model[1] (2) assigns a total weight r to the n diagonal entries of the variable X in such a way that $\tilde{M}$ can be well approximated using nonnegative linear

1. Strictly speaking, (2) is not a linear program but it can be reformulated as one.

Algorithm 1 Hottopixx - Extracting Columns of a Noisy Separable Matrix using Linear Optimization (Bittorf et al., 2012)

Input: A normalized noisy r-separable matrix $\tilde{M} = WH + N \in \mathbb{R}_+^{m \times n}$, the factorization rank r, the noise level $\|N\|_1 \leq \epsilon$ and a vector $p \in \mathbb{R}^n$ with distinct entries.
Output: A matrix $\tilde{W}$ such that $\tilde{W} \approx W$ (up to permutation).

1: Find the optimal solution X^* of (2).
2: Let $\mathcal{K}$ be the index set corresponding to the r largest diagonal entries of X^*.
3: Set $\tilde{W} = \tilde{M}(:, \mathcal{K})$.

combinations of columns of $\tilde{M}$ corresponding to positive diagonal entries of X. Moreover, the weights used in the linear combinations cannot exceed the diagonal entries of X since $X(:, j) \leq \text{diag}(X)$ for all j. There are several drawbacks in using the LP model (2) in practice:

1. The factorization rank r has to be chosen in advance. In practice the true factorization rank is often unknown, and a "good" factorization rank for the application at hand is typically found by trial and error. Therefore the LP above may have to be resolved many times.

2. The columns of the input data matrix have to be normalized in order for their entries to sum to one. This may introduce significant distortions in the data set and lead to poor performance; see Kumar et al. (2013) where some numerical experiments are presented.

3. The noise level $\|N\|_1 \leq \epsilon$ has to be estimated.

4. One has to solve a rather large optimization problem with n^2 variables, so that the model cannot be used directly for huge-scale problems.

It is important to notice that there is no way to getting rid of both drawbacks 2. and 3. In fact, in the noisy case, the user has to indicate either

- The factorization rank r, and the algorithm should find a subset of r columns of $\tilde{M}$ as close as possible to the columns of W, or

- The noise level ϵ, and the algorithm should try to find the smallest possible subset of columns of $\tilde{M}$ allowing to approximate $\tilde{M}$ up to the required accuracy.

1.3 Contribution and Outline of the Paper

In this paper, we generalize Hottopixx in order to resolve drawbacks 1. and 2. above. More precisely, we propose a new LP model which has the following properties:

- Given the noise level ϵ, it detects the number r of columns of W automatically; see Section 2.

- It can be adapted to dealing with outliers; see Section 3.

- It does not require column normalization; see Section 4.
- It is significantly more tolerant to noise than Hottopixx. In fact, we propose a tight robustness analysis of the new LP model proving its superiority (see Theorems 2 and 6). This is illustrated in Section 5 on several synthetic data sets, where the new LP model is shown to outperform Hottopixx while competing favorably with two state-of-the-art methods, namely the successive projection algorithm (SPA) (Araújo et al., 2001; Gillis and Vavasis, 2014) and the fast conical hull algorithm (XRAY) (Kumar et al., 2013).

The emphasis of our work lies in a thorough theoretical understanding of such LP based approaches, and the numerical experiments in Section 5 illustrate the proven robustness properties. An implementation for real-word, large-scale problems is, however, a topic outside the scope of this work (see Section 6).

2. Detecting the Factorization Rank Automatically

In this section, we analyze the following LP model:

$$\begin{aligned} \min_{X \in \mathbb{R}_+^{n \times n}} \quad & p^T \operatorname{diag}(X) \\ \text{such that} \quad & \|\tilde{M} - \tilde{M}X\|_1 \leq \rho\epsilon, \\ & X(i,i) \leq 1 \text{ for all } i, \\ & X(i,j) \leq X(i,i) \text{ for all } i,j, \end{aligned} \tag{3}$$

where p has *positive* entries and $\rho > 0$ is a parameter. We also analyze the corresponding near-separable NMF algorithm (Algorithm 2) with an emphasis on *robustness*. The LP

Algorithm 2 Extracting Columns of a Noisy Separable Matrix using Linear Optimization

Input: A normalized noisy r-separable matrix $\tilde{M} = WH + N \in \mathbb{R}_+^{m \times n}$, the noise level $\|N\|_1 \leq \epsilon$, a parameter $\rho > 0$ and a vector $p \in \mathbb{R}^n$ with positive distinct entries.

Output: An m-by-r matrix $\tilde{W}$ such that $\tilde{W} \approx W$ (up to permutation).

1: Compute an optimal solution X^* of (3).

2: Let $\mathcal{K}$ be the index set corresponding to the diagonal entries of X^* larger than $1 - \frac{\min(1,\rho)}{2}$.

3: $\tilde{W} = \tilde{M}(:, \mathcal{K})$.

model (3) is exactly the same as (2) except that the constraint $\operatorname{tr}(X) = r$ has been removed, and that there is an additional parameter ρ. Moreover, the vector $p \in \mathbb{R}^n$ in the objective function has to be *positive*, or otherwise any diagonal entry of an optimal solution of (3) corresponding to a negative entry of p will be equal to one (in fact, this reduces the objective function the most while minimizing $\|M - MX\|_1$). A natural value for the parameter ρ is two, as in the original LP model (2), so that the matrix X^0 in Equation (1) identifying the set of columns of $\tilde{M}$ corresponding to the columns of W is feasible. However, the model (3) is feasible for any $\rho \geq 0$ since the identity matrix of dimension n (that is, $X = I_n$) is always

feasible. Hence, it is not clear a priori which value of ρ should be chosen. The reason we analyze the LP model (3) for different values of ρ is two-fold:

- First, it shows that the LP model (3) is rather flexible as it is not too sensitive to the right-hand side of the constraint $\|M - MX\|_1 \leq \rho\epsilon$. In other terms, the noise level does not need to be known precisely for the model to make sense. This is a rather desirable property as, in practice, the value of ϵ is typically only known/evaluated approximately.

- Second, we observed that taking ρ smaller than two gives in average significantly better results (see Section 5 for the numerical experiments). Our robustness analysis of Algorithm 2 will suggest that the best choice is to take $\rho = 1$ (see Remark 5).

In this section, we prove that the LP model (3) allows to identifying approximately the columns of the matrix W among the columns of the matrix $\tilde{M}$ *for any* $\rho > 0$, given that the noise level ϵ is sufficiently small (ϵ will depend on the value ρ); see Theorems 2, 6 and 7.

Before stating the robustness results, let us define the conditioning of a nonnegative matrix W whose entries of each column sum to one:

$$\kappa = \min_{1 \leq k \leq r} \min_{x \in \mathbb{R}_+^{r-1}} \|W(:,k) - W(:,\mathcal{K})x\|_1, \qquad \text{where } \mathcal{K} = \{1, 2, \ldots, r\} \backslash \{k\},$$

and the matrix W is said to be κ-*robustly conical*. The parameter $0 \leq \kappa \leq 1$ tells us how well the columns of W are spread in the unit simplex. In particular, if $\kappa = 1$, then W contains the identity matrix as a submatrix (all other entries being zeros) while, if $\kappa = 0$, then at least one of the columns of W belongs to the convex cone generated by the others. Clearly, the better the columns of W are spread across the unit simplex, the less sensitive is the data to noise. For example, $\epsilon < \frac{\kappa}{2}$ is a necessary condition to being able to distinguish the columns of W (Gillis, 2013).

2.1 Robustness Analysis without Duplicates and Near Duplicates

In this section, we assume that the columns of W are isolated (that is, there is no duplicate nor near duplicate of the columns of W in the data set) hence more easily identifiable. This type of margin constraint is typical in machine learning (Bittorf et al., 2012), and is equivalent to bounding the entries of H' in the expression $M = W[I_r, H']\Pi$, see Equation (1). In fact, for any $1 \leq k \leq r$ and $h \in \mathbb{R}_+^r$ with $\max_i h(i) \leq \beta \leq 1$, we have that

$$\begin{aligned} \|W(:,k) - Wh\|_1 &= \|(1 - h(k))W(:,k) - W(:,\mathcal{K})h(\mathcal{K})\|_1 \\ &\geq (1 - \beta) \min_{y \in \mathbb{R}_+^{r-1}} \|W(:,k) - W(:,\mathcal{K})y\|_1 \\ &\geq (1 - \beta)\kappa, \end{aligned}$$

where $\mathcal{K} = \{1, 2, \ldots, r\} \backslash \{k\}$. Hence $\max_{ij} H'_{ij} \leq \beta$ implies that all data points are at distance at least $(1 - \beta)\kappa$ of any column of W. Under this condition, we have the following robustness result:

Theorem 2 *Suppose $\tilde{M} = M + N$ where the entries of each column of M sum to one, $M = WH$ admits a rank-r separable factorization of the form (1) with $\max_{ij} H'_{ij} \leq \beta \leq 1$ and W κ-robustly conical with $\kappa > 0$, and $\|N\|_1 \leq \epsilon$. If*

$$\epsilon \leq \frac{\kappa(1-\beta)\min(1,\rho)}{5(\rho+2)},$$

then Algorithm 2 extracts a matrix $\tilde{W} \in \mathbb{R}^{m\times r}$ satisfying $\|W - \tilde{W}(:,P)\|_1 \leq \epsilon$ for some permutation P.

Proof See Appendix A. ■

Remark 3 (Noiseless case) *When there is no noise (that is, $N = 0$ and $\epsilon = 0$), duplicates and near duplicates are allowed in the data set; otherwise $\epsilon > 0$ implying that $\beta < 1$ hence the columns of W are isolated.*

Remark 4 (A slightly better bound) *The bound on the allowable noise in Theorem 2 can be slightly improved, so that under the same conditions we can allow a noise level of*

$$\epsilon < \frac{\kappa(1-\beta)\min(1,\rho)}{4(\rho+2)+\kappa(1-\beta)\min(1,\rho)}.$$

However, the scope for substantial improvements is limited, as we will show in Theorem 6.

Remark 5 (Best choice for ρ) *Our analysis suggests that the best value for ρ is one. In fact,*

$$\text{argmax}_{\rho\geq 0} \frac{\min(1,\rho)}{(\rho+2)} \;=\; 1.$$

In this particular case, the upper bound on the noise level to guarantee recovery is given by $\epsilon \leq \frac{\kappa(1-\beta)}{15}$ while, for $\rho = 2$, we have $\epsilon \leq \frac{\kappa(1-\beta)}{20}$. The choice $\rho = 1$ is also optimal in the same sense for the bound in the previous remark. We will see in Section 5, where we present some numerical experiments, that choosing $\rho = 1$ works remarkably better than $\rho = 2$.

It was proven by Gillis (2013) that, for Algorithm 1 to extract the columns of W under the same assumptions as in Theorem 2, it is necessary that

$$\epsilon < \frac{\kappa(1-\beta)}{(r-1)(1-\beta)+1} \quad \text{for any } r \geq 3 \text{ and } \beta < 1,$$

while it is sufficient that $\epsilon \leq \frac{\kappa(1-\beta)}{9(r+1)}$. Therefore, if there are no duplicate nor near duplicate of the columns of W in the data set,

> *Algorithm 2 is more robust than Hottopixx (Algorithm 1): in fact, unlike Hottopixx, its bound on the noise to guarantee recovery (up to the noise level) is* independent of the number of columns of W. *Moreover, given the noise level, it detects the number of columns of W automatically.*

The reason for the better performance of Algorithm 2 is the following: for most noisy r-separable matrices $\tilde{M}$, there typically exist matrices X' satisfying the constraints of (3) and such that $\mathrm{tr}(X') < r$. Therefore, the remaining weight $(r - \mathrm{tr}(X'))$ will be assigned by Hottopixx to the diagonal entries of X' corresponding to the smallest entries of p, since the objective is to minimize $p^T \mathrm{diag}(X')$. These entries are unlikely to correspond to columns of W (in particular, if p in chosen by an adversary). We observed that when the noise level ϵ increases, $r - \mathrm{tr}(X')$ increases as well, hence it becomes likely that some columns of W will not be identified.

Example 1 *Let us consider the following simple instance:*

$$M = \underbrace{I_r}_{=W} \underbrace{\left[I_r, \frac{e}{r}\right]}_{=H} \in \mathbb{R}^{r \times (r+1)} \quad \text{and} \quad N = 0,$$

where e is the vector of all ones. We have that $||N||_1 = 0 \leq \epsilon$ for any $\epsilon \geq 0$.

Using $p = [1, 2, \ldots, r, -1]$ in the objective function, the Hottopixx LP (2) will try to put as much weight as possible on the last diagonal entry of X (that is, $X(r+1, r+1)$) which corresponds to the last column of M. Moreover, because W is the identity matrix, no column of W can be used to reconstruct another column of W (this could only increase the error) so that Hottopixx has to assign a weight to the first r diagonal entries of X larger than $(1 - 2\epsilon)$ (in order for the constraint $||M - MX||_1 \leq 2\epsilon$ to be satisfied). The remaining weight of $2r\epsilon$ (the total weight has to be equal to r) can be assigned to the last column of M. Hence, for $1 - 2\epsilon < 2r\epsilon \iff \epsilon > \frac{1}{2(r+1)}$, Hottopixx will fail as it will extract the last column of M.

Let us consider the new LP model (3) with $\rho = 2$. For the same reason as above, it has to assign a weight to the first r diagonal entries of X larger than $(1 - 2\epsilon)$. Because the cost of the last column of M has to be positive (that is, $p(r+1) > 0$), the new LP model (3) will try to minimize the last diagonal entry of X (that is, $X(r+1, r+1)$). Since $M(:, r+1) = \frac{1}{r} We$, $X(r+1, r+1)$ can be taken equal to zero taking $X(1 : r, r+1) = \frac{1-2\epsilon}{r}$. Therefore, for any positive vector p, any r and any $\epsilon < \frac{1}{2}$, the new LP model (3) will identify correctly all columns of W. (For other values of ρ, this will be true for any $\epsilon < \frac{1}{\rho}$.)

This explains why the LP model enforcing the constraint $\mathrm{tr}(X) = r$ is less robust, and why its bound on the noise depends on the factorization rank r. Moreover, the LP (2) is also much more sensitive to the parameter ϵ than the model LP (3):

- For ϵ sufficiently small, it becomes infeasible, while,
- for ϵ too large, the problem described above is worsened: there are matrices X' satisfying the constraints of (3) and such that $\mathrm{tr}(X') \ll r$, hence Hottopixx will perform rather poorly (especially in the worst-case scenario, that is, if the problem is set up by an adversary).

To conclude this section, we prove that the bound on the noise level ϵ to guarantee the recovery of the columns of W by Algorithm 2 given in Theorem 2 is tight up to some constant multiplicative factor.

Theorem 6 *For any fixed $\rho > 0$ and $\beta < 1$, the bound on ϵ in Theorem 2 is tight up to a multiplicative factor. In fact, under the same assumptions on the input matrix $\tilde{M}$, it is necessary that $\epsilon < \frac{\kappa(1-\beta)\min(1,\rho)}{2\rho}$ for Algorithm 2 to extract a matrix $\tilde{W} \in \mathbb{R}^{m\times r}$ satisfying $\|W - \tilde{W}(:,P)\|_1 \leq \epsilon$ for some permutation P.*

Proof See Appendix B. ■

For example, Theorem 6 implies that, for $\rho = 1$, the bound of Theorem 2 is tight up to a factor $\frac{15}{2}$.

2.2 Robustness Analysis with Duplicates and Near Duplicates

In case there are duplicates and near duplicates in the data set, it is necessary to apply a post-processing to the solution of (3). In fact, although we can guarantee that there is a subset of the columns of $\tilde{M}$ close to each column of W whose sum of the corresponding diagonal entries of an optimal solution of (3) is large, there is no guarantee that the weight will be concentrated only in one entry. It is then required to apply some post-processing based on the distances between the data points to the solution of (3) (instead of simply picking the r indices corresponding to its largest diagonal entries) in order to obtain a robust algorithm. In particular, using Algorithm 4 to post-process the solution of (2) leads to a more robust algorithm than Hottopixx (Gillis, 2013). Note that pre-processing would also be possible (Esser et al., 2012; Arora et al., 2012a).

Therefore, we propose to post-process an optimal solution of (3) with Algorithm 4; see Algorithm 3, for which we can prove the following robustness result:

Theorem 7 *Let $M = WH$ be an r-separable matrix whose entries of each column sum to one and of the form (1) with $H \geq 0$ and W κ-robustly conical. Let also $\tilde{M} = M + N$ with $\|N\|_1 \leq \epsilon$. If*

$$\epsilon < \frac{\omega\kappa}{99(r+1)},$$

where $\omega = \min_{i\neq j}\|W(:,i) - W(:,j)\|_1$, then Algorithm 3 extracts a matrix $\tilde{W}$ such that

$$\|W - \tilde{W}(:,P)\|_1 \leq 49(r+1)\frac{\epsilon}{\kappa} + 2\epsilon, \quad \text{for some permutation } P.$$

Proof See Appendix C (for simplicity, we only consider the case $\rho = 2$; the proof can be generalized for other values of $\rho > 0$ in a similar way as in Theorem 2). ■

This robustness result follows directly from (Gillis, 2013, Theorem 5), and is the same as for the algorithm using the optimal solution of (2) post-processed with Algorithm 4. Hence, in case there are duplicates and near duplicates in the data set, we do not know if Algorithm 3 is more robust, although we believe the bound for Algorithm 3 can be improved (in particular, that the dependence in r can be removed), this is a topic for further research.

Remark 8 (Choice of p) *Although Theorem 7 requires the entries of the vector p to be all ones, we recommend to take the entries of p distinct, but close to one. This allows the LP (3) to discriminate better between the duplicates hence Algorithm 3 does not necessarily have to enter the post-processing loop. We suggest to use $p(i) \sim 1 + \mathcal{U}(-\sigma,\sigma)$ for all i, where $\sigma \ll 1$ and $\mathcal{U}(a,b)$ is the uniform distribution in the interval $[a,b]$.*

Algorithm 3 Extracting Columns of a Noisy Separable Matrix using Linear Optimization

Input: A normalized r-separable matrix $\tilde{M} = WH + N$, and the noise level $\|N\|_1 \leq \epsilon$.
Output: An m-by-r matrix $\tilde{W}$ such that $\tilde{W} \approx W$ (up to permutation).

1: Compute the optimal solution X^* of (3) where $p = e$ is the vector of all ones and $\rho = 2$.

2: $K = \text{post-processing}\Big(\tilde{M}, \text{diag}(X^*), \epsilon\Big)$;

3: $\tilde{W} = \tilde{M}(:, \mathcal{K})$;

3. Handling Outliers

Removing the rank constraint has another advantage: it allows to deal with outliers. If the data set contains outliers, the corresponding diagonal entries of an optimal solution X^* of (3) will have to be large (since outliers cannot be approximated well with convex combinations of points in the data set). However, under some reasonable assumptions, outliers are useless to approximate data points, hence off-diagonal entries of the rows of X^* corresponding to outliers will be small. Therefore, one could discriminate between the columns of W and the outliers by looking at the *off-diagonal entries of* X^*. This result is closely related to the one presented by Gillis and Vavasis, 2014 (Section 3). For simplicity, we consider in this section only the case where $\rho = 2$ and assume absence of duplicates and near-duplicates in the data set; the more general case can be treated in a similar way.

Let the columns of $T \in \mathbb{R}^{m \times t}$ be t outliers added to the separable matrix $W[I_r, H']$ along with some noise to obtain

$$\tilde{M} = M + N \quad \text{where} \quad M = [W, T]H = \big[W, T, WH'\big] \begin{bmatrix} I_r & 0_{r\times t} & H' \\ 0_{t\times r} & I_t & 0_{t\times r} \end{bmatrix} \Pi, \tag{4}$$

which is a noisy r-separable matrix containing t outliers. We propose Algorithm 5 to approximately extract the columns of W among the columns of $\tilde{M}$.

In order for Algorithm 5 to extract the correct set of columns of $\tilde{M}$, the off-diagonal entries of the rows corresponding to the columns of T (resp. columns of W) must be small (resp. large). This can be guaranteed using the following conditions (see also Theorem 9 below):

- The angle between the cone generated by the columns of T and the columns space of W is positive. More precisely, we will assume that for all $1 \leq k \leq t$

$$\min_{\substack{x \in \mathbb{R}^t_+, x(k) = 1, \\ y \in \mathbb{R}^r}} \|Tx - Wy\|_1 \ \geq \ \eta > 0. \tag{5}$$

  In fact, if a nonnegative linear combination of outliers (that is, Tx with $x \geq 0$) belongs to the column space of W, then some data points can usually be reconstructed using a non-zero weight for these outliers (it suffices that some data points belong to the convex hull of some columns of W and that linear combination of outliers).

Algorithm 4 Post-Processing - Clustering Diagonal Entries of X^* (Gillis, 2013)

Input: A matrix $\tilde{M} \in \mathbb{R}^{m \times n}$, a vector $x \in \mathbb{R}^n_+$, $\epsilon \geq 0$, and possibly a factorization rank r.

Output: A index set $\mathcal{K}^*$ with r indices so that the columns of $\tilde{M}(:, \mathcal{K}^*)$ are centroids whose corresponding clusters have large weight (the weights of the data points are given by x).

$D(i,j) = \|m_i - m_j\|_1$ for $1 \leq i, j \leq n$;
if r is not part of the input **then**
    $r = \left\lceil \sum_i x(i) \right\rceil$;
else
    $x \leftarrow r \frac{x}{\sum_i x(i)}$;
end if
$\mathcal{K} = \mathcal{K}^* = \left\{ k \mid x(k) > \frac{r}{r+1} \right\}$ and $\nu = \nu^* = \max\left(2\epsilon, \min_{\{(i,j) | D(i,j) > 0\}} D(i,j)\right)$;
while $|\mathcal{K}| < r$ and $\nu < \max_{i,j} D(i,j)$ **do**
    $\mathcal{S}_i = \{ j \mid D(i,j) \leq \nu \}$ for $1 \leq i \leq n$;
    $w(i) = \sum_{j \in \mathcal{S}_i} x(j)$ for $1 \leq i \leq n$;
    $\mathcal{K} = \emptyset$;
    while $\max_{1 \leq i \leq n} w(i) > \frac{r}{r+1}$ **do**
        $k = \operatorname{argmax} w(i)$; $\mathcal{K} \leftarrow \mathcal{K} \cup \{k\}$;
        For all $1 \leq i \leq n$ and $j \in \mathcal{S}_k \cup \mathcal{S}_i$: $w(i) \leftarrow w(i) - x(j)$;
    end while
    if $|\mathcal{K}| > |\mathcal{K}^*|$ **then**
        $\mathcal{K}^* = \mathcal{K}$; $\nu = \nu^*$;
    end if
    $\nu \leftarrow 2\nu$;
end while
% Safety procedure in case the conditions of Theorem 7 are not satisfied:
if $|\mathcal{K}^*| < r$ **then**
    $\mathrm{d} = \max_{i,j} D(i,j)$;
    $\mathcal{S}_i = \{ j \mid D(i,j) \leq \nu^* \}$ for $1 \leq i \leq n$;
    $w(i) = \sum_{j \in \mathcal{S}_i} x(j)$ for $1 \leq i \leq n$;
    $\mathcal{K}^* = \emptyset$;
    while $|\mathcal{K}^*| < r$ **do**
        $k = \operatorname{argmax} w(i)$; $\mathcal{K}^* \leftarrow \mathcal{K}^* \cup \{k\}$;
        For all $1 \leq i \leq n$, and $j \in \mathcal{S}_k \cup \mathcal{S}_i$: $w(i) \leftarrow w(i) - \left(\frac{\mathrm{d} - D(i,j)}{\mathrm{d}}\right)^{0.1} x(j)$;
        $w(k) \leftarrow 0$;
    end while
end if

- The matrix $[W, T]$ is robustly conical, otherwise some columns of T could be reconstructed using other columns of T whose corresponding rows could hence have large off-diagonal entries.

Algorithm 5 Extracting Columns of a Noisy Separable Matrix with Outliers using Linear Optimization

Input: A normalized noisy r-separable matrix $\tilde{M} = [W, T, WH']\Pi + N \in \mathbb{R}^{m \times n}_+$ with outliers, the noise level $\|N\|_1 \leq \epsilon$ and a vector $p \in \mathbb{R}^n$ with positive distinct entries and $\rho = 2$.

Output: An m-by-r matrix $\tilde{W}$ such that $\tilde{W} \approx W$ (up to permutation).

1: Compute the optimal solution X^* of (3) where p has distinct positive entries.
2: Let $\mathcal{K} = \left\{1 \leq k \leq n \mid X^*(k,k) \geq \frac{1}{2} \text{ and } \|X^*(k,:)\|_1 - X^*(k,k) \geq \frac{1}{2}\right\}$.
3: $\tilde{W} = \tilde{M}(:, \mathcal{K})$.

- Each column of W is necessary to reconstruct at least one data point, otherwise the off-diagonal entries of the row of X^* corresponding to that 'useless' column of W will be small, possibly equal to zero, and it cannot be distinguished from an outlier. More formally, for all $1 \leq k \leq r$, there is a least one data point $M(:,j) = WH(:,j) \neq W(:,k)$ such that

$$\min_{x \geq 0, y \geq 0} \|M(:,j) - Tx - W(:,\mathcal{K})y\|_1 \geq \delta, \quad \text{where } \mathcal{K} = \{1, 2, \ldots, r\} \backslash \{k\}. \tag{6}$$

  If Equation (5) holds, this condition is satisfied for example when $\text{conv}(W)$ is a simplex and some points lie inside that simplex (it is actually satisfied if and only if each column of W define with other columns of W a simplex containing at least one data point in its interior).

These conditions allow to distinguish the columns of W from the outliers using off-diagonal entries of an optimal solution X^* of (3):

Theorem 9 *Suppose $\tilde{M} = M + N$ where the entries of each column of M sum to one, $M = [W, T]H$ has the form (4) with $H \geq 0$, $\max_{ij} H'_{ij} \leq \beta \leq 1$ and $[W, T]$ κ-robustly conical, and $\|N\|_1 \leq \epsilon$. Suppose also that M, W and T satisfy Equations (5) and (6) for some $\eta > 0$ and $\delta > 0$. If*

$$\epsilon \leq \frac{\nu(1-\beta)}{20(n-1)} \quad \text{where } \nu = \min(\kappa, \eta, \delta),$$

then Algorithm 5 extracts a matrix $\tilde{W} \in \mathbb{R}^{m \times r}$ satisfying $\|W - \tilde{W}(:,P)\|_1 \leq \epsilon$ for some permutation P.

Proof See Appendix D. ■

Unfortunately, the factor $\frac{1}{n-1}$ is necessary because a row of X^* corresponding to an outlier could potentially be assigned weights proportional to ϵ for all off-diagonal entries. For example, if all data points are perturbed in the direction of an outlier, that is, $N(:,j) = \epsilon T(:,k)$ for all j and for some $1 \leq k \leq t$, then we could have $\sum_{j \neq k} X(k,j) = (n-1)\mathcal{O}(\epsilon)$ hence it is necessary that $\epsilon \leq \mathcal{O}(n^{-1})$ (although it is not likely to happen in practice). A simple way to improve the bound is the following:

- Identify the vertices and outliers using $\mathcal{K} = \left\{1 \leq k \leq n \mid X^*(k,k) \geq \frac{1}{2}\right\}$ (this only requires $\epsilon \leq \frac{\kappa(1-\beta)}{20}$, cf. Theorem 2).
- Solve the linear program $Z^* = \operatorname{argmin}_{Z \geq 0} \|M - M(:,\mathcal{K})Z\|_1$.
- Use the sum of the rows of Z^* (instead of X^*) to identify the columns of W.

Following the same steps as in the proof of Theorem 9, the bound for ϵ for the corresponding algorithm becomes $\epsilon \leq \frac{\nu(1-\beta)}{20(r+t-1)}$.

Remark 10 (Number of outliers) *Algorithm 5 does not require the number of outliers as an input. Moreover, the number of outliers is not limited hence our result is stronger than the one of Gillis and Vavasis (2014) where the number of outliers cannot exceed $m-r$ (because T needs to be full rank, while we only need T to be robustly conical and the cone generated by its columns define a wide angle with the column space of W).*

Remark 11 (Hottopixx and outliers) *Replacing the constraint $\operatorname{tr}(X) = r$ with $\operatorname{tr}(X) = r+t$ (r is the number of columns of W and t is the number of outliers) in the LP model (2) allows to deal with outliers. However, the number of outliers plus the number of columns of W (that is, $r+t$) has to be estimated, which is rather impractical.*

4. Avoiding Column Normalization

In order to use the LP models (2) and (3), normalization must be enforced which may introduce significant distortions in the data set and lead to poor performances (Kumar et al., 2013). If M is r-separable but the entries of each column do not sum to one, we still have that

$$M = W[I_r, H']\Pi = [W, WH']\Pi = [W, WH'] \begin{pmatrix} I_r & H' \\ 0_{(n-r)\times r} & 0_{(n-r)\times(n-r)} \end{pmatrix} \Pi = MX^0.$$

However, the constraints $X(i,j) \leq X(i,i)$ for all i,j in the LP's (2) and (3) are not necessarily satisfied by the matrix X^0, because the entries of H' can be arbitrarily large.

Let us denote $\tilde{M}_o$ the original unnormalized noisy data matrix, and its normalized version $\tilde{M}$, with

$$\tilde{M}(:,j) = \frac{\tilde{M}_o(:,j)}{\|\tilde{M}_o(:,j)\|_1} \quad \text{for all } j.$$

Let us also rewrite the LP (3) in terms of $\tilde{M}_o$ instead of $\tilde{M}$ using the following change of variables

$$X_{ij} = \frac{\|\tilde{M}_o(:,i)\|_1}{\|\tilde{M}_o(:,j)\|_1} Y_{ij} \quad \text{for all } i,\ j.$$

Note that $Y_{ii} = X_{ii}$ for all i. We have for all j that

$$\begin{aligned} \left\| \tilde{M}(:,j) - \sum_i \tilde{M}(:,i) X_{ij} \right\|_1 &= \left\| \frac{\tilde{M}_o(:,j)}{\|\tilde{M}_o(:,j)\|_1} - \sum_j \frac{\tilde{M}_o(:,i)}{\|\tilde{M}_o(:,i)\|_1} \frac{\|\tilde{M}_o(:,i)\|_1}{\|\tilde{M}_o(:,j)\|_1} Y_{ij} \right\|_1 \\ &= \frac{1}{\|\tilde{M}_o(:,j)\|_1} \left\| \tilde{M}_o(:,j) - \sum_j \tilde{M}_o(:,i) Y_{ij} \right\|_1, \end{aligned}$$

which proves that the following LP

$$\min_{Y \in \mathcal{Y}} \quad p^T \operatorname{diag}(Y) \quad \text{such that} \quad \|\tilde{M}_o(:,j) - \tilde{M}_o Y(:,j)\|_1 \leq \rho\epsilon \|\tilde{M}_o(:,j)\|_1 \ \text{ for all } j, \tag{7}$$

where

$$\mathcal{Y} = \{Y \in \mathbb{R}_+^{n \times n} \mid Y(i,i) \leq 1 \ \forall\, i, \ \text{ and } \ \|\tilde{M}_o(:,i)\|_1 Y(i,j) \leq \|\tilde{M}_o(:,j)\|_1 Y(i,i) \ \forall\, i,j\},$$

is equivalent to the LP (3). This shows that the LP (3) looks for an approximation $\tilde{M}_o Y$ of $\tilde{M}_o$ with small *relative error*, which is in general not desirable in practice. For example, a zero column to which some noise is added will have to be approximated rather well, while it does not bring any valuable information. Similarly, the columns of M with large norms will be given relatively less importance while they typically contain a more reliable information (e.g., in document data sets, they correspond to longer documents).

It is now easy to modify the LP (7) to handle other noise models. For example, if the noise added to each column of the input data matrix is independent of its norm, then one should rather use the following LP trying to find an approximation $\tilde{M}_o Y$ of $\tilde{M}_o$ with small *absolute error*:

$$\min_{Y \in \mathcal{Y}} \ p^T \operatorname{diag}(Y) \quad \text{such that} \quad \|\tilde{M}_o - \tilde{M}_o Y\|_1 \leq \rho\epsilon. \tag{8}$$

Remark 12 (Other noise models) *Considering other noise models depending on the problem at hand is also possible: one has to replace the constraint $\|\tilde{M}_o - \tilde{M}_o Y\|_1 \leq \rho\epsilon$ with another appropriate constraint. For example, using any ℓ_q-norm with $q \geq 1$ leads to efficiently solvable convex optimization programs (Glineur and Terlaky, 2004), that is, using*

$$\|\tilde{M}_o(:,j) - \tilde{M}_o Y(:,j)\|_q \leq \rho\epsilon, \quad \text{for all } j.$$

Another possibility is to assume that the noise is distributed among all the entries of the input matrix independently and one could use instead $\sqrt[q]{\sum_{i,j} \left(\tilde{M}_o - \tilde{M}_o Y\right)_{ij}^q} \leq \rho\epsilon$, e.g., $\|\tilde{M}_o - \tilde{M}_o Y\|_F \leq \rho\epsilon$ for Gaussian noise (where $||.||_F$ is the Frobenius norm of a matrix with $q = 2$).

5. Numerical Experiments

In this section, we present some numerical experiments in which we compare our new LP model (8) with Hottopixx and two other state-of-the-art methods. First we describe a practical twist to Algorithm 4, which we routinely apply in the experiments to LP-based solutions.

5.1 Post-Processing of LP solutions

Recall that the LP-based algorithms return a nonnegative matrix X whose diagonal entries indicate the importance of the corresponding columns of the input data matrix $\tilde{M}$. As explained earlier, there are several ways to extract r columns from $\tilde{M}$ using this information, the simplest being to select the columns corresponding to the r largest diagonal entries of

X (Bittorf et al., 2012). Another approach is to take into account the distances between the columns of $\tilde{M}$ and cluster them accordingly; see Algorithm 4. In our experiments we have not observed that one method dominates the other (although in theory, when the noise level is sufficiently small, Algorithm 4 is more robust; see Gillis, 2013). Therefore, the strategy we employ in the experiments below selects the best solution out of the two post-processing strategies based on the residual error, see Algorithm 6.

Algorithm 6 Hybrid Post-Processing for LP-based Near-Separable NMF Algorithms

Input: A matrix $M \in \mathbb{R}^{m \times n}$, a factorization rank r, a noise level ϵ, and a vector of weight $x \in \mathbb{R}^n_+$.

Output: An index set $\mathcal{K}$ such that $\min_{H \geq 0} \|M - M(:, \mathcal{K})H\|_F$ is small.

1: *% Greedy approach*
2: $\mathcal{K}_1$ is the set of the r largest indices of x;
3: *% Clustering using Algorithm 4*
4: $\mathcal{K}_2 = \text{Algorithm 4}\Big(\tilde{M}, x, \epsilon, r\Big)$;
5: *% Select the better of the two*
6: $\mathcal{K} = \text{argmin}_{\mathcal{R} \in \{\mathcal{K}_1, \mathcal{K}_2\}} \min_{H \geq 0} \|M - M(:, \mathcal{R})H\|_F^2$;

5.2 Algorithms

In this section, we compare the following near-separable NMF algorithms:

1. **Hottopixx** (Bittorf et al., 2012). Given the noise level $\|N\|_1$ and the factorization rank r, it computes the optimal solution X^* of the LP (2) (where the input matrix $\tilde{M}$ has to be normalized) and returns the indices obtained using Algorithm 6. The vector p in the objective function was randomly generated using the `randn` function of Matlab. The algorithm of Arora et al. (2012a) was shown to perform worse than Hottopixx (Bittorf et al., 2012) hence we do not include it here (moreover, it requires an additional parameter α related to the conditioning of W which is difficult to estimate in practice).

2. **SPA** (Araújo et al., 2001). The successive projection algorithm (SPA) extracts recursively r columns of the input normalized matrix $\tilde{M}$ as follows: at each step, it selects the column with maximum ℓ_2 norm, and then projects all the columns of $\tilde{M}$ on the orthogonal complement of the extracted column. This algorithm was proved to be robust to noise (Gillis and Vavasis, 2014). (Note that there exist variants where, at each step, the column is selected according to other criteria, e.g., any ℓ_p norm with $1 < p < +\infty$. This particular version of the algorithm using ℓ_2 norm actually dates back from modified Gram-Schmidt with column pivoting, see Gillis and Vavasis, 2014 and the references therein.) SPA was shown to perform significantly better on several synthetic data sets than Hottopixx and several state-of-the-art algorithms from the hyperspectral image community (Gillis and Vavasis, 2014) (these algorithms are based on the pure-pixel assumption which is equivalent to the separability assumption, see Introduction).

3. **XRAY** (Kumar et al., 2013). In Kumar et al. (2013), several fast conical hull algorithms are proposed. We use in this paper the variant referred to as *max*, because it performs in average the best on synthetic data sets. Similarly as SPA, it recursively extracts r columns of the input unnormalized matrix $\tilde{M}_o$: at each step, it selects a column of $\tilde{M}_o$ corresponding to an extreme ray of the cone generated by the columns of $\tilde{M}_o$, and then projects all the columns of $\tilde{M}_o$ on the cone generated by the columns of $\tilde{M}_o$ extracted so far. XRAY was shown to perform much better than Hottopixx and similarly as SPA on synthetic data sets (while performing better than both for topic identification in document data sets as it does not require column normalization). However, it is not known whether XRAY is robust to noise.

4. **LP** (8) with $\rho = 1, 2$. Given the noise level $\|N\|_1$, it computes the optimal solution X^* of the LP (8) and returns the indices obtained with the post-processing described in Algorithm 6. (Note that we have also tried $\rho = \frac{1}{2}$ which performs better than $\rho = 2$ but slightly worse than $\rho = 1$ in average hence we do not display these results here.)

Table 1 gives the following information for the different algorithms: computational cost, memory requirement, parameters and if column normalization of the input matrix is necessary.

	Flops	Memory	Parameters	Normalization
Hottopixx	$\Omega\left(mn^2\right)$	$\mathcal{O}\left(mn + n^2\right)$	$\|N\|_1$, r	Yes
SPA	$2mnr + \mathcal{O}\left(mr^2\right)$	$\mathcal{O}\left(mn\right)$	r	Yes
XRAY	$\mathcal{O}\left(mnr\right)$	$\mathcal{O}\left(mn\right)$	r	No
LP (8)	$\Omega\left(mn^2\right)$	$\mathcal{O}\left(mn + n^2\right)$	$\|N\|_1$	No

Table 1: Comparison of robust algorithms for near-separable NMF for a dense m-by-n input matrix.

The LP have been solved using the IBM ILOG CPLEX Optimizer[2] on a standard Linux box. Because of the greater complexity of the LP-based approaches (formulating (2) and (8) as LP's requires $n^2 + mn$ variables), the size of the input data matrices allowed on a standard machine is limited, roughly $mn^2 \sim 10^6$ (for example, on a two-core machine with 2.99GHz and 2GB of RAM, it already takes about one minute to process a 100-by-100 matrix using CPLEX). In this paper, we mainly focus on the robustness performance of the different algorithms and first compare them on synthetic data sets. We also compare them on the popular swimmer data set. Comparison on large-scale real-world data sets would require dedicated implementations, such as the parallel first-order method proposed by Bittorf et al. (2012) for the LP (2), and is a topic for further research. The code for all algorithms is available at `https://sites.google.com/site/nicolasgillis/code`.

2. The code is available for free for academia at `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

5.3 Synthetic Data Sets

With the algorithms above we have run a benchmark with certain synthetic data sets particularly suited to investigate the robustness behaviour under influence of noise. In all experiments the problem dimensions are fixed to $m = 50$, $n = 100$ and $r = 10$. We conducted our experiments with six different data models. As we will describe next, the models differ in the way the factor H is constructed and the sparsity of the noise matrix N. Given a desired noise level ϵ, the noisy r-separable matrix $\tilde{M} = M + N = WH + N$ is generated as follows:

The entries of W are drawn uniformly at random from the interval $[0, 1]$ (using Matlab's `rand` function). Then each column of W is normalized so that its entries sum to one.

The first r columns of H are always taken as the identity matrix to satisfy the separability assumption. The remaining columns of H and the noise matrix N are generated in two different ways (similar to Gillis and Vavasis, 2014):

1. *Dirichlet.* The remaining 90 columns of H are generated according to a Dirichlet distribution whose r parameters are chosen uniformly in $[0, 1]$ (the Dirichlet distribution generates vectors on the boundary of the unit simplex so that $\|H(:, j)\|_1 = 1$ for all j). Each entry of the noise matrix N is generated following the normal distribution $\mathcal{N}(0, 1)$ (using the `randn` function of Matlab).

2. *Middle Points.* The $\frac{r(r-1)}{2} = 45$ next columns of H resemble all possible equally weighted convex combinations of pairs from the r leading columns of H. This means that the corresponding 45 columns of M are the middle points of pairs of columns of W. The trailing 45 columns of H are generated in the same way as above, using the Dirichlet distribution. No noise is added to the first r columns of M, that is, $N(:, 1 : r) = 0$, while all the other columns are moved toward the exterior of the convex hull of the columns of W using

$$N(:, j) = M(:, j) - \bar{w}, \quad \text{for } r + 1 \leq j \leq n,$$

   where $\bar{w}$ is the average of the columns of W (geometrically, this is the vertex centroid of the convex hull of the columns of W).

We combine these two choices for H and N with three options that control the pattern density of N, thus yielding a total of six different data models:

1. *Dense noise.* Leave the matrix N untouched.

2. *Sparse noise.* Apply a mask to N such that roughly 75% of the entries are set to zero (using the `density` parameter of Matlab's `sprand` function).

3. *Pointwise noise.* Keep only one randomly picked non-zero entry in each nonzero column of N.

Finally we scale the resulting matrix N by a scalar such that $\|N\|_1 = \epsilon$. In order to avoid a bias towards the natural ordering, the columns of $\tilde{M}$ are permuted at random in a last step.

5.3.1 Error Measures and Methodology

Let $\mathcal{K}$ be the set of indices extracted by an algorithm. In our comparisons, we will use the following two error measures:

- *Index recovery*: percentage of correctly extracted indices in $\mathcal{K}$ (recall that we *know* the indices corresponding to the columns of W).
- ℓ_1 *residual norm*: We measure the relative ℓ_1 residual by

$$1 - \min_{H \geq 0} \frac{\|\tilde{M} - \tilde{M}(:, \mathcal{K})H\|_s}{\|\tilde{M}\|_s}. \tag{9}$$

Note that both measures are between zero and one, one being the best possible value, zero the worst.

The aim of the experiments is to display the robustness of the algorithms from Section 5.2 applied to the data sets described in the previous section under increasing noise levels. For each data model, we ran all the algorithms on the same randomly generated data on a predefined range of noise levels ϵ. For each such noise level, 25 data sets were generated and the two measures are averaged over this sample for each algorithm.

5.3.2 Results

Figures 1 and 2 display the results for the three experiments of "Dirichlet" and "Middle Points" types respectively. For comparison purpose, we also display the value of (9) for the true column indices $\mathcal{K}$ of W in M, labeled "true K" in the plots. In all experiments, we observe that

- The new LP model (8) is significantly more robust to noise than Hottopixx, which confirms our theoretical results; see Section 2.1.
- The variant of LP (8) with $\rho = 2$ is less robust than with $\rho = 1$, as suggested by our theoretical findings from Section 2.1.
- SPA and XRAY perform, in average, very similarly.

Comparing the three best algorithms (that is, SPA, XRAY and LP (8) with $\rho = 1$), we have that

- In case of "dense" noise, they give comparable results; although LP (8) with $\rho = 1$ performs slightly worse for the "Dirichlet" type, and slightly better for the "Middle Points" type.
- In case of "sparse" noise, LP (8) with $\rho = 1$ performs consistently better then SPA and XRAY: for all noise levels, it identifies correctly more columns of W and the corresponding NMF's have smaller ℓ_1 residual norms.
- In case of "pointwise" noise, LP (8) with $\rho = 1$ outperforms SPA and XRAY. In particular, for high noise level, it is able to extract correctly almost all columns of

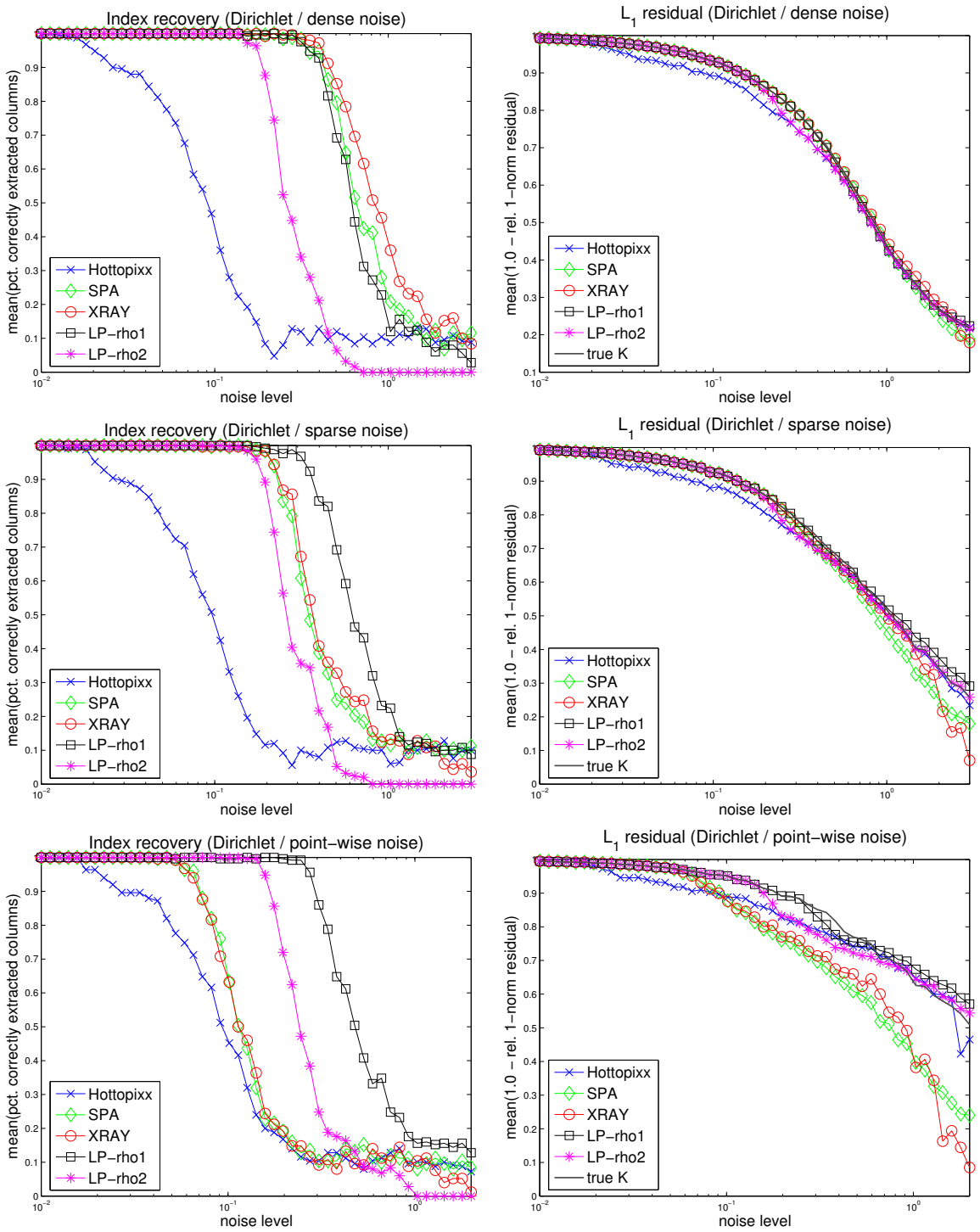


Figure 1: Comparison of near-separable NMF algorithms on "Dirichlet" type data sets. From left to right: index recovery and ℓ_1 residual. From top to bottom: dense noise, sparse noise and pointwise noise.

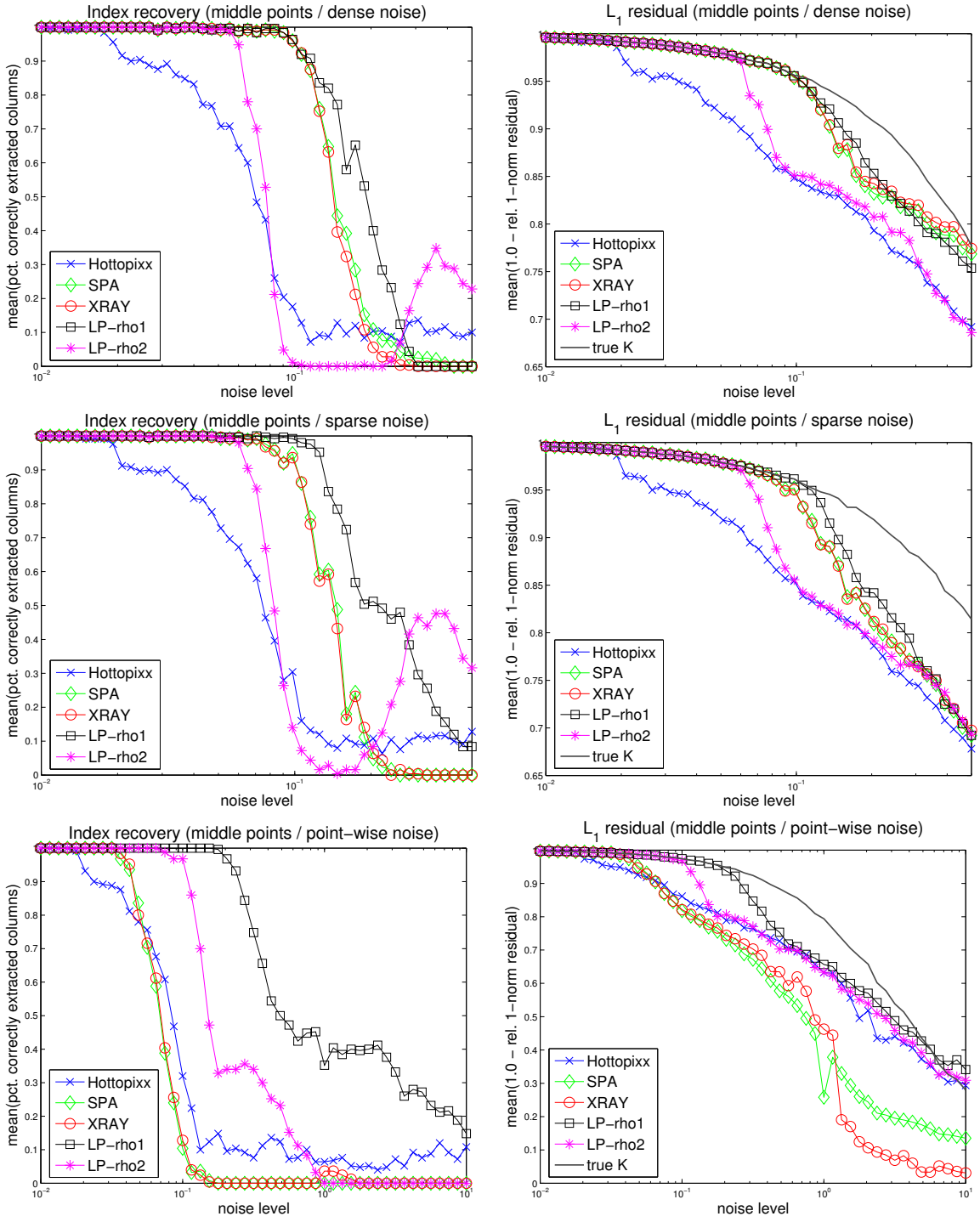


Figure 2: Comparison of near-separable NMF algorithms on "Middle Points" type data sets. From left to right: index recovery and ℓ_1 residual. From top to bottom: dense noise, sparse noise and pointwise noise.

	D/dense	D/sparse	D/pw	MP/dense	MP/sparse	MP/pw
Hottopixx	2.5	2.5	3.6	4.4	4.3	4.2
SPA	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1
XRAY	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1
LP (8), $\rho = 1$	20.5	34.1	39.0	52.5	88.1	41.4
LP (8), $\rho = 2$	10.5	12.3	16.0	32.5	56.9	27.4

Table 2: Average computational time in seconds for the different algorithms and data models. (D stands for Dirichlet, MP for middle points, pw for pointwise.)

W while SPA and XRAY can only extract a few for the "Dirichlet" type (performing as a guessing algorithm since they extract correctly only $r/n = 10\%$ of the columns of W), or none for the "Middle Points" type.

Note that LP (8) with $\rho = 2$ also performs consistently better then SPA and XRAY in case of "pointwise" noise.

Remark 13 *For the "Middle Points" experiments and for large noise levels, the middle points of the columns of W become the vertices of the convex hull of the columns of $\tilde{M}$ (since they are perturbed toward the outside of the convex hull of the columns of W). Hence, near-separable NMF algorithms should not extract any original column of W. However, the index measure for LP* (8) *with $\rho = 2$ increases for larger noise level (although the ℓ_1 residual measure decreases); see Figure 2. It is difficult to explain this behavior because the noise level is very high (close to 100%) hence the separability assumption is far from being satisfied and it is not clear what the LP* (8) *does.*

Table 2 gives the average computational time for a single application of the algorithms to a data set. As expected, the LP-based methods are significantly slower than SPA and XRAY; designing faster solvers is definitely an important topic for further research. Note that the Hottopixx model can be solved about ten times faster on average than the LP model (8), despite the only essential difference being the trace constraint $\mathrm{tr}(X) = r$. It is difficult to explain this behaviour as the number of simplex iterations or geometry of the central path cannot easily be set in relation to the presence or absence of a particular constraint.

Table 3 displays the index recovery robustness: For each algorithm and data model, the maximum noise level $\|N\|_1$ for which the algorithm recovered on average at least 99% of the indices corresponding to the columns of W. In all cases, the LP (8) with $\rho = 1$ is on par or better than all other algorithms.

5.4 Swimmer Data Set

The swimmer data set is a widely used data set for benchmarking NMF algorithms (Donoho and Stodden, 2003). It consists of 256 binary images (20-by-11 pixels) of a body with four limbs which can be each in four different positions; see Figure 3. Let $M \in \{0,1\}^{256\times 220}$ correspond to the swimmer data set where each row corresponds to an image, and each

	D/dense	D/sparse	D/pw	MP/dense	MP/sparse	MP/pw
Hottopixx	0.014	0.018	0.016	0.016	0.018	0.015
SPA	0.220	0.154	0.052	0.077	0.071	0.032
XRAY	**0.279**	0.154	0.052	**0.083**	0.071	0.032
LP (8), $\rho = 1$	**0.279**	**0.195**	**0.197**	**0.083**	**0.098**	**0.178**
LP (8), $\rho = 2$	0.137	0.121	0.141	0.055	0.055	0.075

Table 3: Index recovery robustness: Largest noise level $\|N\|_1$ for which an algorithm achieves almost perfect index recovery (that is, at least 99% on average).

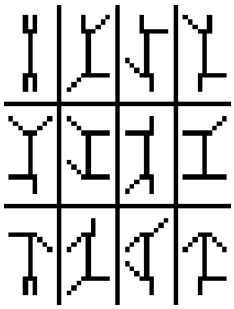


Figure 3: Sample images of the swimmer data set.

column to a pixel. The matrix M is 16-separable: up to permutation, M has the following form

$$M = W \left[I_{16}, I_{16}, I_{16}, \frac{1}{4} E_{16\times 14}, 0_{16\times 158} \right],$$

where $E_{m\times n}$ denotes the m-by-n all-one matrix. In fact, all the limbs are disjoint and contain three pixels (hence each column of W is repeated three times), the body contains fourteen pixels and the remaining 158 background pixels do not contain any information.

Remark 14 (Uniqueness of H) *Note that the weights $\frac{1}{4}E_{16\times 14}$ corresponding to the pixels belonging to the body are not unique. The reason is that the matrix W is not full rank (in fact,* $\text{rank}(W) = 13$*) implying that the convex hull of the columns of W and the origin is not a simplex (that is, $r+1$ vertices in dimension r). Therefore, the convex combination needed to reconstruct a point in the interior of that convex hull is not unique (such as a pixel belonging to the body in this example); see the discussion in Gillis (2012).*

Let us compare the different algorithms on this data set:

- **SPA**. Because the rank of the input matrix M is equal to thirteen, the residual matrix becomes equal to zero after thirteen steps and SPA cannot extract more than thirteen indices hence it fails to decompose M.
- **XRAY**. At the first step, the criterion used by XRAY to identify an extreme ray of the convex hull of the columns of M is maximized by all non-zero columns of M hence

any of them can be extracted. Since there are 48 pixels belonging to a limb and only 14 to the body, XRAY is more likely to extract a pixel on a limb (after which it is able to correctly decompose M). However, if the first pixel extracted by XRAY is a pixel of the body then XRAY requires to be run with $r = 17$ to achieve a perfect decomposition. Therefore, XRAY succeeds on this example only with probability $\frac{48}{62} \sim 77\%$ (given that XRAY picks a column at random among the one maximizing the criterion). We consider here a run where XRAY failed, otherwise it gives the same perfect decomposition as the new LP based approaches; see below.

- **Hottopixx**. With $\epsilon = 0$ in the Hottopixx LP model (2), the columns of W are correctly identified and Hottopixx performs perfectly. However, as soon as ϵ exceeds approximately 0.03, Hottopixx fails in most cases. In particular, if p is chosen such that its smallest entry does not correspond to a columns of W, then it always fails (see also the discussion in Example 1). Even if p is not chosen by an adversary but is randomly generated, this happens with high probability since most columns of M do not correspond to a column of W.

- **LP** (7) **with** $\rho = 1$. For ϵ up to approximately 0.97, the LP model (7) (that is, the new LP model based on relative error) idenfities correctly the columns of W and decomposes M perfectly.

- **LP** (8) **with** $\rho = 1$. For ϵ up to approximately 60, (note that the ℓ_1 norm of the columns of W is equal to 64), the LP model (8) (that is, the new LP model based on absolute error) identifies correctly the columns of W and decomposes M perfectly.

Figure 4 displays the optimal weights corresponding to the columns of M extracted with the different algorithms (that is, the rows of the matrix $H^* = \text{argmin}_{H \geq 0} ||M - M(:, \mathcal{K})H||_F$ where $\mathcal{K}$ is the index set extracted by a given algorithm): the error for SPA is 20.8, for XRAY 12, for Hottopixx 12 and for the new LP models 0. Note that we used $\epsilon = 0.1$ for Hottopixx and the new LP model (a situation in which Hottopixx fails in most cases; see the discussion above—for the particular run shown in Figure 4, Hottopixx extracts a background pixel corresponding to a zero column of M). Note also that we do not display the result for the LP (8) because it gave an optimal solution similar to that of the LP (7). Finally, it is interesting to point out that the nonnegative rank of M is equal to 16 hence the new LP models actually detect the nonnegative rank of M.

6. Conclusion and Further Work

In this paper, we have proposed a new more practical and more robust LP model for near-separable NMF which competes favorably with two state-of-the-art methods (outperforming them in some cases). It would be particularly interesting to investigate the following directions of research:

- Implementation and evaluation of an algorithm to solve (8) for large-sale real-world problems.

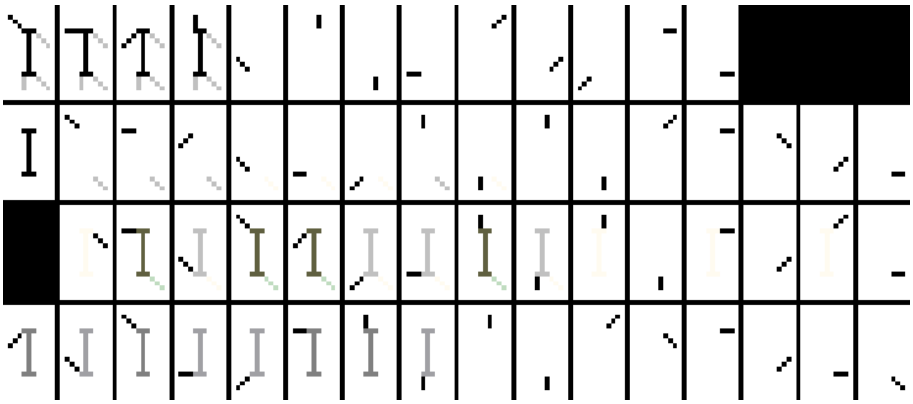


Figure 4: Weights corresponding to the extracted indices by the different algorithms. From to to bottom: SPA, XRAY, Hottopixx ($\epsilon = 0.1$) and the new LP model (7) ($\epsilon = 0.1$).

- Improvement of the theoretical bound on the noise level for Algorithm 3 to extract the right set of columns of the input data matrix in case duplicates and near duplicates are present in the data set (cf. Section 2.2).

- Design of practical and robust near-separable NMF algorithms. For example, would it be possible to design an algorithm as robust as our LP-based approach but computationally more effective (e.g., running in $\mathcal{O}(mnr)$ operations)?

Acknowledgments

The authors would like to thank the reviewers for their feedback which helped improve the paper.

Appendix A. Proof of Theorem 2

The next two lemmas are simple generalizations of Lemmas 2 & 3 in Gillis (2013). Given any feasible solution X of the the linear program (3), the first one shows that the ℓ_1 norm of the error $M - MX$ with respect to the original noiseless data matrix is proportional to ϵ, that is, $\|\tilde{M} - \tilde{M}X\|_1 \leq \mathcal{O}(\epsilon)$. The second one proves that the diagonal entries of X corresponding to the columns of W must be larger than $1 - \mathcal{O}(\epsilon)$.

Lemma 15 *Suppose $\tilde{M} = M + N$ where $\|M(:,j)\|_1 = 1$ for all j and $\|N\|_1 \leq \epsilon < 1$, and suppose X is a feasible solution of (3). Then,*

$$\|X\|_1 \leq 1 + \epsilon \left(\frac{\rho + 2}{1 - \epsilon} \right) \quad \text{and} \quad \|M - MX\|_1 \leq \epsilon \left(\frac{\rho + 2}{1 - \epsilon} \right).$$

Proof First note that $\|\tilde{M}\|_1 \leq \|M\|_1 + \|N\|_1 \leq 1+\epsilon$ and $\|MX\|_1 = \|X\|_1$. By the feasibility of X for (3),

$$\rho\epsilon \geq \|\tilde{M} - \tilde{M}X\|_1 \geq \|\tilde{M}X\|_1 - \|\tilde{M}\|_1 \geq \|MX\|_1 - \|NX\|_1 - (1+\epsilon) \geq \|X\|_1 - \epsilon\|X\|_1 - 1 - \epsilon,$$

hence $\|X\|_1 \leq 1+\epsilon\left(\frac{\rho+2}{1-\epsilon}\right)$, implying that $\|NX\|_1 \leq \|N\|_1\|X\|_1 \leq \epsilon\left(1+\frac{(\rho+2)\epsilon}{1-\epsilon}\right)$. Therefore

$$\rho\epsilon \geq \|\tilde{M} - \tilde{M}X\|_1 = \|M+N-(M+N)X\|_1 \geq \|M - MX\|_1 - \epsilon - \epsilon\left(1+\frac{(\rho+2)\epsilon}{1-\epsilon}\right),$$

from which we obtain $\|M - MX\|_1 \leq \epsilon\left(\rho+2+\frac{(\rho+2)\epsilon}{1-\epsilon}\right) = \epsilon\left(\frac{\rho+2}{1-\epsilon}\right)$ ■

Lemma 16 *Let $\tilde{M} = M + N$ where $||M(:,j)||_1 = 1$ for all j, admits a rank-r separable factorization WH with W κ-robustly conical and $||N||_1 \leq \epsilon < 1$, and has the form (1) with $\max_{i,j} H'_{ij} \leq \beta < 1$ and $W, H \geq 0$. Let also X be any feasible solution of (3), then*

$$X(j,j) \geq 1 - \frac{2\epsilon}{\kappa(1-\beta)}\left(\frac{\rho+2}{1-\epsilon}\right)$$

for all j such that $M(:,j) = W(:,k)$ for some $1 \leq k \leq r$.

Proof The idea of the proof is the following: by assumption, each column of W is isolated from the convex hull of the other columns of M. Therefore, to being able to approximate it up to error $\mathcal{O}(\epsilon)$, its corresponding diagonal entry must be large enough.

Let $\mathcal{K}$ be the set of r indices such that $M(:,\mathcal{K}) = W$. Let also $1 \leq k \leq r$ and denote $j = \mathcal{K}(k)$ so that $M(:,j) = W(:,k)$. By Lemma 15,

$$||W(:,k) - WHX(:,j)||_1 \leq \epsilon\left(\frac{\rho+2}{1-\epsilon}\right). \tag{10}$$

Since $H(k,j) = 1$,

$$\begin{aligned} WHX(:,j) &= W(:,k)H(k,:)X(:,j) + W(:,\mathcal{R})H(\mathcal{R},:)X(:,j) \\ &= W(:,k)\Big(X(j,j) + H(k,\mathcal{J})X(\mathcal{J},j)\Big) + W(:,\mathcal{R})y, \end{aligned}$$

where $\mathcal{R} = \{1,2,\ldots,r\}\backslash\{k\}$, $\mathcal{J} = \{1,2,\ldots,n\}\backslash\{j\}$ and $y = H(\mathcal{R},:)X(:,j) \geq 0$. We have

$$\eta = X(j,j) + H(k,\mathcal{J})X(\mathcal{J},j) \leq X(j,j) + \beta\left(1 + \frac{(\rho+2)\epsilon}{1-\epsilon} - X(j,j)\right), \tag{11}$$

since $||H(k,\mathcal{J})||_\infty \leq \beta$ and $||X(:,j)||_1 \leq 1 + \frac{(\rho+2)\epsilon}{1-\epsilon}$ (Lemma 15). Hence

$$||W(:,k) - WHX(:,j)||_1 \geq (1-\eta)\left\|W(:,k) - W(:,\mathcal{R})\frac{y}{1-\eta}\right\|_1 \geq (1-\eta)\kappa. \tag{12}$$

Combining Equations (10), (11) and (12), we obtain

$$1 - \left(X(j,j) + \beta \left(1 + \frac{(\rho+2)\epsilon}{1-\epsilon} - X(j,j) \right) \right) \leq \frac{\epsilon}{\kappa} \left(\frac{\rho+2}{1-\epsilon} \right)$$

which gives, using the fact that $\kappa, \beta \leq 1$,

$$X(j,j) \geq 1 - \frac{2\epsilon}{\kappa(1-\beta)} \left(\frac{\rho+2}{1-\epsilon} \right).$$

■

If the diagonal entries corresponding to the columns of W of a feasible solution X of (3) are large, then the other diagonal entries will be small. In fact, the columns of M are contained in the convex hull of the columns of W hence can be well approximated with convex combinations of these columns.

Lemma 17 *Let $\tilde{M} = M + N$ where $||M(:,j)||_1 = 1$ for all j, admits a rank-r separable factorization WH and $||N||_1 \leq \epsilon$, and has the form (1). Let $\mathcal{K}$ be the index set with r elements such that $M(:,\mathcal{K}) = W$. Let also X^* be an optimal solution of (3) such that*

$$X^*(k,k) \geq \gamma \quad \text{for all } k \in \mathcal{K}, \tag{13}$$

where $0 \leq \gamma \leq 1$. Then,

$$X^*(j,j) \leq 1 - \min\left(\gamma, \frac{\rho}{2}\right) \quad \text{for all } j \notin \mathcal{K}.$$

Proof Let X be any feasible solution of (3) satisfying (13), and $\alpha = \min\left(\gamma, \frac{\rho}{2}\right)$. Let us show that the jth column of X for some $j \notin \mathcal{K}$ can be modified as follows

$$X(i,j) \leftarrow \begin{cases} 1-\alpha & \text{if } i = j, \\ \alpha H(i,j) & \text{if } i \in \mathcal{K}, \\ 0 & \text{otherwise,} \end{cases}$$

while keeping feasibility. First, $\alpha H(i,j) \leq \gamma \leq X(i,i)$ for all $i \in \mathcal{K}$ hence the condition $X(i,j) \leq X(i,i)$ for all i,j is satisfied while, clearly, $0 \leq X(i,i) \leq 1$ for all i. It remains to show that $||\tilde{M}(:,j) - \tilde{M}X(:,j)||_1 \leq \rho\epsilon$. By assumption, $M(:,j) = WH(:,j) = \alpha WH(:,j) + (1-\alpha)M(:,j)$ hence

$$\begin{aligned} \tilde{M}(:,j) &= \alpha\left(M(:,j) + N(:,j)\right) + (1-\alpha)\tilde{M}(:,j) \\ &= \alpha\left(WH(:,j) + N(:,j)\right) + (1-\alpha)\tilde{M}(:,j). \end{aligned}$$

This gives

$$||\tilde{M}(:,j) - \tilde{M}X(:,j)||_1 = \alpha||M(:,j) + N(:,j) - (W + N(:,\mathcal{K}))H(:,j)||_1 \leq 2\alpha\epsilon \leq \rho\epsilon,$$

since the columns of H sum to one, and $||N||_1 \leq \epsilon$. This result implies that any optimal solution X^* satisfying (13) must satisfy $X^*(j,j) \leq 1 - \alpha$, otherwise we could replace the

jth column of X^* using the construction above and obtain a strictly better solution since the vector p in the objective function only has positive entries. ■

We can now combine Lemmas 16 and 17 to prove robustness of Algorithm 2 when there are no duplicates nor near duplicates of the columns of W in the data set.

Proof [**Proof of Theorem 2**] Let X be an optimal solution of (3). Let us first consider the case $\epsilon = 0$, which is particular because it allows duplicates of the columns of W in the data set and the value of ρ does not influence the analysis since $\rho\epsilon = 0$ for any $\rho > 0$. Let denote

$$\mathcal{K}_k = \{j \mid M(:,j) = W(:,k)\},$$

the set of indices whose corresponding column of M is equal to the kth column of W. By assumption, $\kappa > 0$ hence for all $1 \leq k \leq r$ we have $W(:,k) \notin \text{cone}(W(:,\bar{\mathcal{K}}))$ where $\bar{\mathcal{K}} = \{1,2,\ldots,r\}\backslash\{k\}$. This implies that $\sum_{j\in\mathcal{K}_k} X(j,j) \geq 1$ for all k. Since we are minimizing a positive linear combination of the diagonal entries of X and assigning a weight of one to each cluster $\mathcal{K}_k$ is feasible (see Equation 1), we have $\sum_{j\in\mathcal{K}_k} X(j,j) = 1$. Moreover, assigning all the weight to the index in $\mathcal{K}_k$ with the smallest entry in p minimizes the objective function (and this index is unique since the entries of p are distinct). Finally, for all $1 \leq k \leq r$, there exists a unique j such that $M(:,j) = W(:,k)$ and $X(j,j) = 1$ which gives the result for $\epsilon = 0$.

Otherwise $\epsilon > 0$ and $\beta < 1$, and the result follows from Lemmas 16 and 17: Let $\mathcal{K}$ be the set of r indices such that $M(:,\mathcal{K}) = W$. By Lemma 16, we have

$$X(k,k) \geq 1 - \frac{2\epsilon}{\kappa(1-\beta)}\left(\frac{\rho+2}{1-\epsilon}\right), \quad \text{for all } k \in \mathcal{K},$$

while, by Lemma 17,

$$X(j,j) \leq \max\left(1-\frac{\rho}{2}, \frac{2\epsilon}{\kappa(1-\beta)}\left(\frac{\rho+2}{1-\epsilon}\right)\right), \quad \text{for all } j \notin \mathcal{K}.$$

Therefore, if

$$1 - \frac{2\epsilon}{\kappa(1-\beta)}\left(\frac{\rho+2}{1-\epsilon}\right) \;>\; f \;\geq\; \max\left(1-\frac{\rho}{2}, \frac{2\epsilon}{\kappa(1-\beta)}\left(\frac{\rho+2}{1-\epsilon}\right)\right),$$

where $f = 1 - \frac{\min(1,\rho)}{2} = \max\left(\frac{1}{2}, 1-\frac{\rho}{2}\right)$, then Algorithm 2 extracts the r indices corresponding to the columns of W. The above conditions are satisfied if

$$\frac{2\epsilon}{\kappa(1-\beta)}\left(\frac{\rho+2}{1-\epsilon}\right) < \frac{\rho}{2} \quad \text{and} \quad \frac{2\epsilon}{\kappa(1-\beta)}\left(\frac{\rho+2}{1-\epsilon}\right) < \frac{1}{2},$$

that is, $\frac{\epsilon}{1-\epsilon} < \frac{\kappa(1-\beta)\min(1,\rho)}{4(\rho+2)}$. Taking

$$\epsilon \leq \frac{\kappa(1-\beta)\min(1,\rho)}{5(\rho+2)} < \frac{\kappa(1-\beta)\min(1,\rho)}{\rho+2}\frac{1-\epsilon}{4}$$

gives the results since $\epsilon \leq \frac{1}{5(\rho+2)} < \frac{1}{10}$ for any $\rho > 0$ hence $\frac{1-\epsilon}{4} > \frac{1}{5}$. ■

Appendix B. Proof of Theorem 6

Theorem 6 can be proved using a particular construction.

Proof [**Proof of Theorem 6**] Let us consider

$$W = \begin{pmatrix} \frac{\kappa}{2} I_r \\ (1 - \frac{\kappa}{2}) e_r^T \end{pmatrix}, H = \begin{pmatrix} I_r & \beta I_r + \frac{1-\beta}{r-1}(e_r e_r^T - I_r) \end{pmatrix}, \text{ and } N = 0,$$

where $e_r \in \mathbb{R}^r$ is the all-ones vector, $\frac{1}{r} \leq \beta < 1$ and W is κ-robustly conical with $\kappa > 0$ (Gillis, 2013). Define $p = \begin{pmatrix} Ke_r \\ e_r \end{pmatrix}$ for some large constant K constant. The matrix

$$X = \begin{pmatrix} \left(1 - \frac{\rho\epsilon}{\kappa(1-\beta)}\right) I_r & 0 \\ \frac{\rho\epsilon}{\kappa(1-\beta)} I_r & I_r \end{pmatrix}$$

is a feasible solution of (3) for any $\epsilon \leq \frac{\kappa(1-\beta)}{\rho}$. In fact, for all $1 \leq j \leq r$,

$$||M(:,j) - MX(:,j)||_1 = \frac{\rho\epsilon}{\kappa(1-\beta)} ||M(:,j) - M(:,j+r)||_1 = \rho\epsilon,$$

and it can be easily checked that X satisfies the other constraints. By Lemma 7 of Gillis (2013), for K sufficiently large, any optimal solution X^* of (3) must satisfy

$$\min_{1 \leq k \leq r} X^*(k,k) \leq \max_{1 \leq k \leq r} X(k,k) = 1 - \frac{\rho\epsilon}{\kappa(1-\beta)},$$

(otherwise $p^T \operatorname{diag}(X^*) > p^T \operatorname{diag}(X)$ for K sufficiently large). For the columns of W to be extracted, one requires $X^*(k,k) > 1 - \frac{\min(1,\rho)}{2}$ for all $1 \leq k \leq r$ hence it is necessary that

$$1 - \frac{\rho\epsilon}{\kappa(1-\beta)} > 1 - \frac{\min(1,\rho)}{2} \iff \epsilon < \frac{\kappa(1-\beta)}{2} \frac{\min(1,\rho)}{\rho},$$

for Algorithm 2 to extract the first r columns of M. ■

Appendix C. Proof of Theorem 7

Proof [**Proof of Theorem 7**] The matrix X^0 from Equation (1) is a feasible solution of (3); in fact,

$$||\tilde{M} - \tilde{M}X^0||_1 = ||M + N - (M+N)X^0||_1 \leq ||M - MX^0||_1 + ||N||_1 + ||NX^0||_1 \leq 2\epsilon,$$

since $M = MX^0$, $||N||_1 \leq \epsilon$ and $||NX^0||_1 \leq ||N||_1 ||X^0||_1 \leq \epsilon$ as $||X^0||_1 = 1$. Therefore, since $p = e$, any optimal solution X^* of (3) satisfies

$$\operatorname{tr}(X^*) = p^T \operatorname{diag}(X^*) \leq p^T \operatorname{diag}(X^0) = r.$$

The result then directly follows from Theorem 5 in Gillis (2013). In fact, Algorithm 3 is exactly the same as Algorithm 3 in Gillis (2013) except that the optimal solution of (3) is used instead of (2) while Theorem 5 from Gillis (2013) does not need the entries of p to

be distinct and only the condition $\mathrm{tr}(X) \leq r$ is necessary. Note that Theorem 5 in Gillis (2013) guarantees that there are r disjoint clusters of columns of $\tilde{M}$ around each column of W whose weight is strictly larger $\frac{r}{r+1}$. Therefore, the total weight is strictly larger than $r - \frac{r}{r+1} > r-1$ while it is at most r (since $\mathrm{tr}(X^*) \leq r$) implying that $r = \left\lceil \sum_{i=1}^{n} X^*(i,i) \right\rceil$. ■

Appendix D. Proof of Theorem 9

The proof of Theorem 9 works as follows: Let X be a feasible solution of (3). First, we show that the diagonal entries of X corresponding to the columns of W and T must be large enough (this follows from Theorem 2). Second, we show that the ℓ_1 norm of the rows of X corresponding to the columns of W (resp. T) must be sufficiently large (resp. low) because the columns of W (resp. T) must be used (resp. cannot be used) to reconstruct the other columns of M.

Proof [**Proof of Theorem 9**] In case $\beta = 1$, $\epsilon = 0$ and the proof is similar to that of Theorem 2; the only difference is that the condition from Equation (5) has to be used to show that no weight can be assigned to off-diagonal entries of the rows of an optimal solution of (3) corresponding to the columns of T. Otherwise $\beta < 1$ and there are no duplicate nor near duplicate of the columns of W in the data set.

Let assume without loss of generality that $\tilde{M}$ has the form

$$\tilde{M} = [T, W, WH'] + N,$$

that is, the first t columns correspond to T and the r next ones to W. Let then X be an optimal solution of (3).

Since $[W, T]$ is κ-robustly conical, Theorem 2 applies (as if the columns of T were not outliers) and, for all $1 \leq k \leq r + t$,

$$X(k,k) \geq 1 - \frac{8\epsilon}{\kappa(1-\beta)(1-\epsilon)} \geq \frac{1}{2},$$

while $X(j,j) \leq \frac{8\epsilon}{\kappa(1-\beta)(1-\epsilon)} \leq \frac{1}{2}$ for all $j > r + t$, since $\epsilon \leq \frac{\nu(1-\beta)}{20(n-1)}$ where $\nu = \min(\kappa, \eta, \delta)$. Therefore, only the first $r+t$ indices can potentially be extracted by Algorithm 5. It remains to bound above (resp. below) the off-diagonal entries of the rows of X corresponding to T (resp. W).

By Lemma 16 (see also Gillis, 2013, Lemma 2), we have for all $1 \leq j \leq n$

$$||M(:,j) - MX(:,j)||_1 \leq \frac{4\epsilon}{1-\epsilon} \quad \text{and} \quad ||X(:,j)||_1 \leq 1 + \frac{4\epsilon}{1-\epsilon}.$$

Using the fact that $[W, T]$ is κ-robustly conical, for all $1 \leq k \leq t$, we have

$$||T(:,k) - MX(:,k)||_1 \geq (1 - X(k,k)) \min_{x \geq 0, y \geq 0} ||T(:,k) - T(:,\bar{\mathcal{K}})x - Wy||_1 \geq (1 - X(k,k))\kappa,$$

implying that for all $1 \leq k \leq t$

$$X(k,k) \geq 1 - \frac{4\epsilon}{\kappa(1-\epsilon)} \geq \frac{1}{2},$$

since $\frac{4}{1-\epsilon} \leq 5$ because $\epsilon \leq \frac{1}{20}$. Therefore,

$$\sum_{j \neq k} X(j,k) \leq ||X(:,k)||_1 - X(k,k) \leq \frac{4\epsilon}{1-\epsilon} + \frac{4\epsilon}{\kappa(1-\epsilon)} \leq \frac{8\epsilon}{\kappa(1-\epsilon)},$$

as $\kappa, \epsilon \leq 1$. Let $t+1 \leq j \leq n$ and $1 \leq k \leq t$, we have

$$||M(:,j) - MX(:,j)||_1 \geq \min_x \min_{y \geq 0} ||T(:,k) + T(:,\bar{\mathcal{K}})y - Wx||_1 \geq \eta X(k,j),$$

see Equation (5), which implies $X(k,j) \leq \frac{4\epsilon}{\eta(1-\epsilon)}$. Hence, for all $1 \leq k \leq t$, we have

$$\sum_{j \neq k} X(k,j) \leq (t-1)\frac{8\epsilon}{\kappa(1-\epsilon)} + (n-r-t)\frac{4\epsilon}{\eta(1-\epsilon)} \leq \frac{8(n-1)\epsilon}{\nu(1-\epsilon)} \leq \frac{1}{2}.$$

since $\nu = \min(\kappa, \eta, \delta)$. By assumption, for each $t+1 \leq k \leq t+r$, there exists some j satisfying $M(:,j) = WH(:,j) \neq W(:,k)$ and

$$\min_{x \geq 0} ||M(:,j) - W(:,\bar{\mathcal{K}})x||_1 \geq \delta, \quad \text{where } \bar{\mathcal{K}} = \{1,2,\ldots,r\}\backslash\{k\},$$

see Equation (6). For $t+r < j \leq n$, we have $X(j,j) \leq \frac{8\epsilon}{\kappa(1-\beta)(1-\epsilon)}$. Let us denote $\mu = \frac{8(n-r-t)\epsilon}{\kappa(1-\beta)(1-\epsilon)}$ which is an upper bound for the total weight that can be assigned to the columns of M different from W and T. Then, using Equation (6), we have

$$\begin{aligned}\|M(:,j) - MX(:,j)\|_1 &\geq (1-\mu) \min_{y \geq 0} \left\| M(:,j) - \frac{1}{1-\mu} WX(t+1{:}r+t,j) - Ty \right\|_1 \\ &\geq (1-\mu)\left(1 - \frac{X(k,j)}{1-\mu}\right)\delta.\end{aligned}$$

This implies

$$\frac{X(k,j)}{1-\mu} \geq 1 - \frac{4\epsilon}{\delta(1-\mu)(1-\epsilon)}$$

and

$$\begin{aligned}X(k,j) &\geq 1 - \frac{8(n-r-t)\epsilon}{\kappa(1-\beta)(1-\epsilon)} - \frac{4\epsilon}{\delta(1-\epsilon)} \\ &\geq 1 - \frac{8(n-1)\epsilon}{\nu(1-\beta)(1-\epsilon)} \geq \frac{1}{2},\end{aligned}$$

since $\beta \leq 1$ and $\epsilon \leq \frac{\nu(1-\beta)}{20(n-1)}$, and the proof is complete. ■

References

U.M.C. Araújo, B.T.C. Saldanha, R.K.H. Galvão, T. Yoneyama, H.C. Chame, and V. Visani. The successive projections algorithm for variable selection in spectroscopic multicomponent analysis. *Chemometrics and Intelligent Laboratory Systems*, 57(2):65–73, 2001.

S. Arora, R. Ge, R. Kannan, and A. Moitra. Computing a nonnegative matrix factorization – provably. In *Proc. of the 44th Symp. on Theory of Computing*, STOC '12, pages 145–162, 2012a.

S. Arora, R. Ge, and A. Moitra. Learning topic models - going beyond SVD. In *Proc. of the 53rd Annual IEEE Symp. on Foundations of Computer Science*, FOCS '12, pages 1–10, 2012b.

J.M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot. Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(2):354–379, Apr. 2012.

V. Bittorf, B. Recht, E. Ré, and J.A. Tropp. Factoring nonnegative matrices with linear programs. In *Advances in Neural Information Processing Systems (NIPS '12)*, pages 1223–1231, 2012.

T.-H. Chan, W.-K. Ma, C.-Y. Chi, and Y. Wang. A convex analysis framework for blind separation of non-negative sources. *IEEE Trans. on Signal Processing*, 56(10):5120–5134, 2008.

L. Chen, P.L. Choyke, T.-H. Chan, C.-Y. Chi, G. Wang, and Y. Wang. Tissue-specific compartmental analysis for dynamic contrast-enhanced MR imaging of complex tumors. *IEEE Trans. on Medical Imaging*, 30(12):2044–2058, 2011.

D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems (NIPS '03)*, 2003.

E. Elhamifar, G. Sapiro, and R. Vidal. See all by looking at a few: Sparse modeling for finding representative objects. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2012.

E. Esser, M. Moller, S. Osher, G. Sapiro, and J. Xin. A convex model for nonnegative matrix factorization and dimensionality reduction on physical space. *IEEE Trans. on Image Processing*, 21(7):3239–3252, 2012.

N. Gillis. Sparse and unique nonnegative matrix factorization through data preprocessing. *Journal of Machine Learning Research*, 13(Nov):3349–3386, 2012.

N. Gillis. Robustness analysis of Hottopixx, a linear programming model for factoring nonnegative matrices. *SIAM J. Mat. Anal. Appl.*, 34(3):1189–1212, 2013.

N. Gillis and S.A. Vavasis. Fast and robust recursive algorithms for separable nonnegative matrix factorization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(4):698–714, 2014.

F. Glineur and T. Terlaky. Conic formulation for l_p-norm optimization. *Journal of Optimization Theory and Applications*, 122(2):285–307, 2004.

A. Kumar, V. Sindhwani, and P. Kambadur. Fast conical hull algorithms for near-separable non-negative matrix factorization. In *Int. Conf. on Machine Learning (ICML '13)*, volume 28, pages 231–239. 2013.

S.A. Vavasis. On the complexity of nonnegative matrix factorization. *SIAM J. on Optimization*, 20(3):1364–1377, 2009.

Journal of Machine Learning Research 15 (2014) 1281-1316 Submitted 1/13; Revised 1/14; Published 4/14

Follow the Leader If You Can, Hedge If You Must

Steven de Rooij STEVEN.DE.ROOIJ@GMAIL.COM
VU University and University of Amsterdam
Science Park 904, P.O. Box 94323, 1090 GH Amsterdam, the Netherlands

Tim van Erven TIM@TIMVANERVEN.NL
Département de Mathématiques
Université Paris-Sud, 91405 Orsay Cedex, France

Peter D. Grünwald PDG@CWI.NL
Wouter M. Koolen WMKOOLEN@CWI.NL
Leiden University (Grünwald) and Centrum Wiskunde & Informatica (Grünwald and Koolen)
Science Park 123, P.O. Box 94079, 1090 GB Amsterdam, the Netherlands

Editor: Nicolò Cesa-Bianchi

Abstract

Follow-the-Leader (FTL) is an intuitive sequential prediction strategy that guarantees constant regret in the stochastic setting, but has poor performance for worst-case data. Other hedging strategies have better worst-case guarantees but may perform much worse than FTL if the data are not maximally adversarial. We introduce the FlipFlop algorithm, which is the first method that provably combines the best of both worlds. As a stepping stone for our analysis, we develop AdaHedge, which is a new way of dynamically tuning the learning rate in Hedge without using the doubling trick. AdaHedge refines a method by Cesa-Bianchi, Mansour, and Stoltz (2007), yielding improved worst-case guarantees. By interleaving AdaHedge and FTL, FlipFlop achieves regret within a constant factor of the FTL regret, without sacrificing AdaHedge's worst-case guarantees. AdaHedge and FlipFlop do not need to know the range of the losses in advance; moreover, unlike earlier methods, both have the intuitive property that the issued weights are invariant under rescaling and translation of the losses. The losses are also allowed to be negative, in which case they may be interpreted as gains.

Keywords: Hedge, learning rate, mixability, online learning, prediction with expert advice

1. Introduction

We consider sequential prediction in the general framework of Decision Theoretic Online Learning (DTOL) or the *Hedge setting* (Freund and Schapire, 1997), which is a variant of *prediction with expert advice* (Littlestone and Warmuth, 1994; Vovk, 1998; Cesa-Bianchi and Lugosi, 2006). Our goal is to develop a sequential prediction algorithm that performs well not only on adversarial data, which is the scenario most studies worry about, but also when the data are easy, as is often the case in practice. Specifically, with adversarial data, the worst-case regret (defined below) for any algorithm is $\Omega(\sqrt{T})$, where T is the number of predictions to be made. Algorithms such as Hedge, which have been designed to achieve this lower bound, typically continue to suffer regret of order $\sqrt{T}$, even for easy data, where

the regret of the more intuitive but less robust Follow-the-Leader (FTL) algorithm (also defined below) is *bounded*. Here, we present the first algorithm which, up to constant factors, provably achieves both the regret lower bound in the worst case, *and* a regret not exceeding that of FTL. Below, we first describe the Hedge setting. Then we introduce FTL, discuss sophisticated versions of Hedge from the literature, and give an overview of the results and contents of this paper.

1.1 Overview

In the Hedge setting, prediction proceeds in rounds. At the start of each round $t = 1, 2, \ldots$, a learner has to decide on a weight vector $\boldsymbol{w}_t = (w_{t,1}, \ldots, w_{t,K}) \in \mathbb{R}^K$ over K "experts". Each weight $w_{t,k}$ is required to be nonnegative, and the sum of the weights should be 1. Nature then reveals a K-dimensional vector containing the losses of the experts $\boldsymbol{\ell}_t = (\ell_{t,1}, \ldots, \ell_{t,K}) \in \mathbb{R}^K$. Learner's loss is the dot product $h_t = \boldsymbol{w}_t \cdot \boldsymbol{\ell}_t$, which can be interpreted as the expected loss if Learner uses a mixed strategy and chooses expert k with probability $w_{t,k}$. We denote aggregates of per-trial quantities by their capital letter, and vectors are in bold face. Thus, $L_{t,k} = \ell_{1,k} + \ldots + \ell_{t,k}$ denotes the cumulative loss of expert k after t rounds, and $H_t = h_1 + \ldots + h_t$ is Learner's cumulative loss (the *Hedge loss*).

Learner's performance is evaluated in terms of her *regret*, which is the difference between her cumulative loss and the cumulative loss of the best expert:

$$\mathcal{R}_t = H_t - L_t^*, \qquad \text{where } L_t^* = \min_k L_{t,k}.$$

We will always analyse the regret after an arbitrary number of rounds T. We will omit the subscript T for aggregate quantities such as L_T^* or $\mathcal{R}_T$ wherever this does not cause confusion.

A simple and intuitive strategy for the Hedge setting is Follow-the-Leader (FTL), which puts all weight on the expert(s) with the smallest loss so far. More precisely, we will define the weights $\boldsymbol{w}_t$ for FTL to be uniform on the set of leaders $\{k \mid L_{t-1,k} = L_{t-1}^*\}$, which is often just a singleton. FTL works very well in many circumstances, for example in stochastic scenarios where the losses are independent and identically distributed (i.i.d.). In particular, the regret for Follow-the-Leader is bounded by the number of times the leader is overtaken by another expert (Lemma 10), which in the i.i.d. case almost surely happens only a finite number of times (by the uniform law of large numbers), provided the mean loss of the best expert is strictly smaller than the mean loss of the other experts. As demonstrated by the experiments in Section 5, many more sophisticated algorithms can perform significantly worse than FTL.

The problem with FTL is that it breaks down badly when the data are antagonistic. For example, if one out of two experts incurs losses $\frac{1}{2}, 0, 1, 0, \ldots$ while the other incurs opposite losses $0, 1, 0, 1, \ldots$, the regret for FTL at time T is about $T/2$ (this scenario is further discussed in Section 5.1). This has prompted the development of a multitude of alternative algorithms that provide better worst-case regret guarantees.

The seminal strategy for the learner is called *Hedge* (Freund and Schapire, 1997, 1999). Its performance crucially depends on a parameter η called the *learning rate*. Hedge can be interpreted as a generalisation of FTL, which is recovered in the limit for $\eta \to \infty$. In many analyses, the learning rate is changed from infinity to a lower value that optimizes

some upper bound on the regret. Doing so requires precognition of the number of rounds of the game, or of some property of the data such as the eventual loss of the best expert L^*. Provided that the relevant statistic is monotonically nondecreasing in t (such as L_t^*), a simple way to address this issue is the so-called *doubling trick*: setting a budget on the statistic, and restarting the algorithm with a double budget when the budget is depleted (Cesa-Bianchi and Lugosi, 2006; Cesa-Bianchi et al., 1997; Hazan and Kale, 2008); η can then be optimised for each individual block in terms of the budget. Better bounds, but harder analyses, are typically obtained if the learning rate is adjusted each round based on previous observations, see e.g. (Cesa-Bianchi and Lugosi, 2006; Auer et al., 2002).

The Hedge strategy presented by Cesa-Bianchi, Mansour, and Stoltz (2007) is a sophisticated example of such adaptive tuning. The relevant algorithm, which we refer to as CBMS, is defined in (16) in Section 4.2 of their paper. To discuss its guarantees, we need the following notation. Let $\ell_t^- = \min_k \ell_{t,k}$ and $\ell_t^+ = \max_k \ell_{t,k}$ denote the smallest and largest loss in round t, and let $L_t^- = \ell_1^- + \ldots + \ell_t^-$ and $L_t^+ = \ell_1^+ + \ldots + \ell_t^+$ denote the cumulative minimum and maximum loss respectively. Further let $s_t = \ell_t^+ - \ell_t^-$ denote the loss range in trial t and let $S_t = \max\{s_1, \ldots, s_t\}$ denote the largest loss range after t trials. Then, *without prior knowledge* of any property of the data, including T, S and L^*, the CBMS strategy achieves regret bounded by[1]

$$\mathcal{R}^{\text{CBMS}} \leq 4\sqrt{\frac{(L^* - L^-)(L^- + ST - L^*)}{T} \ln K} + \text{lower order terms} \tag{1}$$

(Cesa-Bianchi et al., 2007, Corollary 3). Hence, in the worst case $L^* = L^- + ST/2$ and the bound is of order $S\sqrt{T}$, but when the loss of the best expert $L^* \in [L^-, L^- + ST]$ is close to either boundary the guarantees are much stronger.

The contributions of this work are twofold: first, in Section 2, we develop AdaHedge, which is a refinement of the CBMS strategy. A (very) preliminary version of this strategy was presented at NIPS (Van Erven et al., 2011). Like CMBS, AdaHedge is completely parameterless and tunes the learning rate in terms of a direct measure of past performance. We derive an improved worst-case bound of the following form. Again without any assumptions, we have

$$\mathcal{R}^{\text{ah}} \leq 2\sqrt{S\frac{(L^* - L^-)(L^+ - L^*)}{L^+ - L^-} \ln K} + \text{lower order terms} \tag{2}$$

(see Theorem 8). The parabola under the square root is always smaller than or equal to its CMBS counterpart (since it is nondecreasing in L^+ and $L^+ \leq L^- + ST$); it expresses that the regret is small if $L^* \in [L^-, L^+]$ is close to either boundary. It is maximized in L^* at the midpoint between L^- and L^+, and in this case we recover the worst-case bound of order $S\sqrt{T}$. Like (1), the regret bound (2) is "fundamental", which means that it is invariant under translation of the losses and proportional to their scale. Moreover, not only AdaHedge's regret *bound* is fundamental: the weights issued by the algorithm are themselves invariant

1. As pointed out by a referee, it is widely known that the leading constant of 4 can be improved to $2\sqrt{2} \approx 2.83$ using techniques by Györfi and Ottucsák (2007) that are essentially equivalent to our Lemma 2 below; Gerchinovitz (2011, Remark 2.2) reduced it to approximately 2.63. AdaHedge allows a slight further reduction to 2.

under translation and scaling (see Section 4). The CBMS algorithm and AdaHedge are insensitive to trials in which all experts suffer the same loss, a natural property we call "timelessness". An attractive feature of the new bound (2) is that it expresses this property. A more detailed discussion appears below Theorem 8.

Our second contribution is to develop a second algorithm, called FlipFlop, that retains the worst-case bound (2) (up to a constant factor), but has even better guarantees for easy data: its performance is never substantially worse than that of Follow-the-Leader. At first glance, this may seem trivial to accomplish: simply take both FTL and AdaHedge, and combine the two by using FTL or Hedge recursively. To see why such approaches do not work, suppose that FTL achieves regret $\mathcal{R}^{\text{ftl}}$, while AdaHedge achieves regret $\mathcal{R}^{\text{ah}}$. We would only be able to prove that the regret of the combined strategy compared to the best original expert satisfies $\mathcal{R}^{\text{c}} \leq \min\{\mathcal{R}^{\text{ftl}}, \mathcal{R}^{\text{ah}}\} + \mathcal{G}^{\text{c}}$, where $\mathcal{G}^{\text{c}}$ is the worst-case regret guarantee for the combination method, e.g. (1). In general, either $\mathcal{R}^{\text{ftl}}$ or $\mathcal{R}^{\text{ah}}$ may be close to zero, while at the same time the regret of the combination method, or at least its bound $\mathcal{G}^{\text{c}}$, is proportional to $\sqrt{T}$. That is, the overhead of the combination method will dominate the regret!

The FlipFlop approach we describe in Section 3 circumvents this by alternating between Following the Leader and using AdaHedge in a carefully specified way. For this strategy we can guarantee

$$\mathcal{R}^{\text{ff}} = O(\min\{\mathcal{R}^{\text{ftl}}, \mathcal{G}^{\text{ah}}\}),$$

where $\mathcal{G}^{\text{ah}}$ is the regret guarantee for AdaHedge; Theorem 15 provides a precise statement. Thus, FlipFlop is the first algorithm that provably combines the benefits of Follow-the-Leader with robust behaviour for antagonistic data.

A key concept in the design and analysis of our algorithms is what we call the *mixability gap*, introduced in Section 2.1. This quantity also appears in earlier works, and seems to be of fundamental importance in both the current Hedge setting as well as in stochastic settings. We elaborate on this in Section 6.2 where we provide the big picture underlying this research and we briefly indicate how it relates to practical work such as (Devaine et al., 2013).

1.2 Related Work

As mentioned, AdaHedge is a refinement of the strategy analysed by Cesa-Bianchi et al. (2007), which is itself more sophisticated than most earlier approaches, with two notable exceptions. First, Chaudhuri, Freund, and Hsu (2009) describe a strategy called NormalHedge that can efficiently compete with the best ϵ-quantile of experts; their bound is incomparable with the bounds for CBMS and for AdaHedge. Second, Hazan and Kale (2008) develop a strategy called Variation MW that has especially low regret when the losses of the best expert vary little between rounds. They show that the regret of Variation MW is of order $\sqrt{\text{VAR}_T^{\max} \ln K}$, where $\text{VAR}_T^{\max} = \max_{t \leq T} \sum_{s=1}^{t} \left(\ell_{s,k_t^*} - \frac{1}{t} L_{t,k_t^*}\right)^2$ with k_t^* the best expert after t rounds. This bound dominates our worst-case result (2) (up to a multiplicative constant). As demonstrated by the experiments in Section 5, their method does not achieve the benefits of FTL, however. In Section 5 we also discuss the performance of NormalHedge and Variation MW compared to AdaHedge and FlipFlop.

Other approaches to sequential prediction include Defensive Forecasting (Vovk et al., 2005), and Following the Perturbed Leader (Kalai and Vempala, 2003). These radically different approaches also allow competing with the best ϵ-quantile, as shown by Chernov and Vovk (2010) and Hutter and Poland (2005); the latter also consider nonuniform weights on the experts.

The "safe MDL" and "safe Bayesian" algorithms by Grünwald (2011, 2012) share the present work's focus on the mixability gap as a crucial part of the analysis, but are concerned with the stochastic setting where losses are not adversarial but i.i.d. FlipFlop, safe MDL and safe Bayes can all be interpreted as methods that attempt to choose a learning rate η that keeps the mixability gap small (or, equivalently, that keeps the Bayesian posterior or Hedge weights "concentrated").

1.3 Outline

In the next section we present and analyse AdaHedge and compare its worst-case regret bound to existing results, in particular the bound for CBMS. Then, in Section 3, we build on AdaHedge to develop the FlipFlop strategy. The analysis closely parallels that of AdaHedge, but with extra complications at each of the steps. In Section 4 we show that both algorithms have the property that their behaviour does not change under translation and scaling of the losses. We further illustrate the relationship between the learning rate and the regret, and compare AdaHedge and FlipFlop to existing methods, in experiments with artificial data in Section 5. Finally, Section 6 contains a discussion, with ambitious suggestions for future work.

2. AdaHedge

In this section, we present and analyse the AdaHedge strategy. To introduce our notation and proof strategy, we start with the simplest possible analysis of vanilla Hedge, and then move on to refine it for AdaHedge.

2.1 Basic Hedge Analysis for Constant Learning Rate

Following Freund and Schapire (1997), we define the *Hedge* or *exponential weights* strategy as the choice of weights

$$w_{t,k} = \frac{w_{1,k} e^{-\eta L_{t-1,k}}}{Z_t}, \tag{3}$$

where $\boldsymbol{w}_1 = (1/K, \ldots, 1/K)$ is the uniform distribution, $Z_t = \boldsymbol{w}_1 \cdot e^{-\eta \boldsymbol{L}_{t-1}}$ is a normalizing constant, and $\eta \in (0, \infty)$ is a parameter of the algorithm called the *learning rate*. If $\eta = 1$ and one imagines $L_{t-1,k}$ to be the negative log-likelihood of a sequence of observations, then $w_{t,k}$ is the Bayesian posterior probability of expert k and Z_t is the marginal likelihood of the observations. Like in Bayesian inference, the weights are updated multiplicatively, i.e. $w_{t+1,k} \propto w_{t,k} e^{-\eta \ell_{t,k}}$.

The loss incurred by Hedge in round t is $h_t = \boldsymbol{w}_t \cdot \boldsymbol{\ell}_t$, the cumulative Hedge loss is $H_t = h_1 + \ldots + h_t$, and our goal is to obtain a good bound on H_T. To this end, it turns

out to be technically convenient to approximate h_t by the *mix loss*

$$m_t = -\frac{1}{\eta} \ln(\boldsymbol{w}_t \cdot e^{-\eta \boldsymbol{\ell}_t}), \tag{4}$$

which accumulates to $M_t = m_1 + \ldots + m_t$. This approximation is a standard tool in the literature. For example, the mix loss m_t corresponds to the loss of Vovk's (1998; 2001) Aggregating Pseudo Algorithm, and tracking the evolution of $-m_t$ is a crucial ingredient in the proof of Theorem 2.2 of Cesa-Bianchi and Lugosi (2006).

The definitions may be extended to $\eta = \infty$ by letting η tend to ∞. We then find that $\boldsymbol{w}_t$ becomes a uniform distribution on the set of experts $\{k \mid L_{t-1,k} = L^*_{t-1}\}$ that have incurred smallest cumulative loss before time t. That is, Hedge with $\eta = \infty$ reduces to *Follow-the-Leader*, where in case of ties the weights are distributed uniformly. The limiting value for the mix loss is $m_t = L^*_t - L^*_{t-1}$.

In our approximation of the Hedge loss h_t by the mix loss m_t, we call the approximation error $\delta_t = h_t - m_t$ the *mixability gap*. Bounding this quantity is a standard part of the analysis of Hedge-type algorithms (see, for example, Lemma 4 of Cesa-Bianchi et al. 2007) and it also appears to be a fundamental notion in sequential prediction even when only so-called mixable losses are considered (Grünwald, 2011, 2012); see also Section 6.2. We let $\Delta_t = \delta_1 + \ldots + \delta_t$ denote the cumulative mixability gap, so that the regret for Hedge may be decomposed as

$$\mathcal{R} = H - L^* = M - L^* + \Delta. \tag{5}$$

Here $M - L^*$ may be thought of as the regret under the mix loss and Δ is the cumulative approximation error when approximating the Hedge loss by the mix loss. Throughout the paper, our proof strategy will be to analyse these two contributions to the regret, $M - L^*$ and Δ, separately.

The following lemma, which is proved in Appendix A, collects a few basic properties of the mix loss:

Lemma 1 (Mix Loss with Constant Learning Rate) *For any learning rate $\eta \in (0, \infty]$*

1. *$\ell_t^- \leq m_t \leq h_t \leq \ell_t^+$, so that $0 \leq \delta_t \leq s_t$.*
2. *Cumulative mix loss telescopes: $M = \begin{cases} -\frac{1}{\eta} \ln \left(\boldsymbol{w}_1 \cdot e^{-\eta \boldsymbol{L}} \right) & \text{for } \eta < \infty, \\ L^* & \text{for } \eta = \infty. \end{cases}$*
3. *Cumulative mix loss approximates the loss of the best expert: $L^* \leq M \leq L^* + \frac{\ln K}{\eta}$.*
4. *The cumulative mix loss M is nonincreasing in η.*

In order to obtain a bound for Hedge, one can use the following well-known bound on the mixability gap, which is obtained using Hoeffding's bound on the cumulant generating function (Cesa-Bianchi and Lugosi, 2006, Lemma A.1):

$$\delta_t \leq \frac{\eta}{8} s_t^2, \tag{6}$$

from which $\Delta \leq S^2 T\eta/8$, where (as in the introduction) $S_t = \max\{s_1, \ldots, s_t\}$ is the maximum loss range in the first t rounds. Together with the bound $M - L^* \leq \ln(K)/\eta$ from mix loss property #3 this leads to

$$\mathcal{R} = (M - L^*) + \Delta \leq \frac{\ln K}{\eta} + \frac{\eta S^2 T}{8}. \tag{7}$$

The bound is optimized for $\eta = \sqrt{8\ln(K)/(S^2 T)}$, which equalizes the two terms. This leads to a bound on the regret of $S\sqrt{T\ln(K)/2}$, matching the lower bound on worst-case regret from the textbook by Cesa-Bianchi and Lugosi (2006, Section 3.7). We can use this tuned learning rate if the time horizon T is known in advance. To deal with the situation where T is unknown, either the doubling trick or a time-varying learning rate (see Lemma 2 below) can be used, at the cost of a worse constant factor in the leading term of the regret bound.

In the remainder of this section, we introduce a completely parameterless algorithm called AdaHedge. We then refine the steps of the analysis above to obtain a better regret bound.

2.2 AdaHedge Analysis

In the previous section, we split the regret for Hedge into two parts: $M - L^*$ and Δ, and we obtained a bound for both. The learning rate η was then tuned to equalise these two bounds. The main distinction between AdaHedge and other Hedge approaches is that AdaHedge does not consider an upper bound on Δ in order to obtain this balance: instead it aims to equalize Δ and $\ln(K)/\eta$. As the cumulative mixability gap Δ_t is nondecreasing in t (by mix loss property #1) and can be *observed* on-line, it is possible to adapt the learning rate directly based on Δ_t.

Perhaps the easiest way to achieve this is by using the doubling trick: each subsequent block uses half the learning rate of the previous block, and a new block is started as soon as the observed cumulative mixability gap Δ_t exceeds the bound on the mix loss $\ln(K)/\eta$, which ensures these two quantities are equal at the end of each block. This is the approach taken in an earlier version of AdaHedge (Van Erven et al., 2011). However, we can achieve the same goal much more elegantly, by decreasing the learning rate with time according to

$$\eta_t^{\mathrm{ah}} = \frac{\ln K}{\Delta_{t-1}^{\mathrm{ah}}} \tag{8}$$

(where $\Delta_0^{\mathrm{ah}} = 0$, so that $\eta_1^{\mathrm{ah}} = \infty$). Note that the AdaHedge learning rate does not involve the end time T or any other unobserved properties of the data; all subsequent analysis is therefore valid for all T simultaneously. The definitions (3) and (4) of the weights and the mix loss are modified to use this new learning rate:

$$w_{t,k}^{\mathrm{ah}} = \frac{w_{1,k}^{\mathrm{ah}} e^{-\eta_t^{\mathrm{ah}} L_{t-1,k}}}{\boldsymbol{w}_1^{\mathrm{ah}} \cdot e^{-\eta_t^{\mathrm{ah}} \boldsymbol{L}_{t-1}}} \qquad \text{and} \qquad m_t^{\mathrm{ah}} = -\frac{1}{\eta_t^{\mathrm{ah}}} \ln(\boldsymbol{w}_t^{\mathrm{ah}} \cdot e^{-\eta_t^{\mathrm{ah}} \boldsymbol{\ell}_t}), \tag{9}$$

with $\boldsymbol{w}_1^{\mathrm{ah}} = (1/K, \ldots, 1/K)$ uniform. Note that the multiplicative update rule for the weights no longer applies when the learning rate varies with t; the last three results of Lemma 1 are also no longer valid. Later we will also consider other algorithms to determine

Single round quantities for trial t:		
$\boldsymbol{\ell}_t$		Loss vector
ℓ_t^-	$= \min_k \ell_{t,k}$, $\ell_t^+ = \max_k \ell_{t,k}$	Min and max loss
s_t	$= \ell_t^+ - \ell_t^-$	Loss range
$\boldsymbol{w}_t^{\text{alg}}$	$= e^{-\eta_t^{\text{alg}} \cdot \boldsymbol{L}_{t-1}} / \sum_k e^{-\eta_t^{\text{alg}} L_{t-1,k}}$	Weights played
h_t^{alg}	$= \boldsymbol{w}_t^{\text{alg}} \cdot \boldsymbol{\ell}_t$	Hedge loss
m_t^{alg}	$= -\frac{1}{\eta_t^{\text{alg}}} \ln \left(\boldsymbol{w}_t^{\text{alg}} \cdot e^{-\eta_t^{\text{alg}} \boldsymbol{\ell}_t} \right)$	Mix loss
δ_t^{alg}	$= h_t^{\text{alg}} - m_t^{\text{alg}}$	Mixability gap
v_t^{alg}	$= \text{Var}_{k \sim \boldsymbol{w}_t^{\text{alg}}}[\ell_{t,k}]$	Loss variance
Aggregate quantities after t rounds:		
(The final time T is omitted from the subscript where possible, e.g. $L^* = L_T^*$)		
$\boldsymbol{L}_t$, L_t^-, L_t^+, H_t^{alg}, M_t^{alg}, Δ_t^{alg}, V_t^{alg}		$\sum_{\tau=1}^t$ of $\boldsymbol{\ell}_\tau$, ℓ_τ^-, ℓ_τ^+, h_τ^{alg}, m_τ^{alg}, δ_τ^{alg}, v_τ^{alg}
S_t	$= \max\{s_1, \ldots, s_t\}$	Maximum loss range
L_t^*	$= \min_k L_{t,k}$	Cumulative loss of the best expert
$\mathcal{R}_t^{\text{alg}}$	$= H_t^{\text{alg}} - L_t^*$	Regret
*Algorithms (the "*alg*" in the superscript above):*		
(η)		Hedge with fixed learning rate η
ah		AdaHedge, defined by (8)
ftl		Follow-the-Leader ($\eta^{\text{ftl}} = \infty$)
ff		FlipFlop, defined by (16)

Table 1: Notation

variable learning rates; to avoid confusion the considered algorithm is always specified in the superscript in our notation. See Table 1 for reference. From now on, AdaHedge will be defined as the Hedge algorithm with learning rate defined by (8). For concreteness, a MATLAB implementation appears in Figure 1.

Our learning rate is similar to that of Cesa-Bianchi et al. (2007), but it is less pessimistic as it is based on the mixability gap Δ_t itself rather than its bound, and as such may exploit easy sequences of losses more aggressively. Moreover our tuning of the learning rate simplifies the analysis, leading to tighter results; the essential new technical ingredients appear as Lemmas 3, 5 and 7 below.

We analyse the regret for AdaHedge like we did for a fixed learning rate in the previous section: we again consider $M^{\text{ah}} - L^*$ and Δ^{ah} separately. This time, both legs of the analysis become slightly more involved. Luckily, a good bound can still be obtained with only a small amount of work. First we show that the mix loss is bounded by the mix loss we would have incurred if we would have used the final learning rate η_T^{ah} all along:

Lemma 2 *Let* dec *be any strategy for choosing the learning rate such that $\eta_1 \geq \eta_2 \geq \ldots$ Then the cumulative mix loss for* dec *does not exceed the cumulative mix loss for the strategy that uses the last learning rate η_T from the start: $M^{\text{dec}} \leq M^{(\eta_T)}$.*

```
% Returns the losses of AdaHedge.
% l(t,k) is the loss of expert k at time t
function h = adahedge(l)
    [T, K]  = size(l);
    h       = nan(T,1);
    L       = zeros(1,K);
    Delta   = 0;

    for t = 1:T
        eta = log(K)/Delta;
        [w, Mprev] = mix(eta, L);
        h(t) = w * l(t,:)';
        L = L + l(t,:);
        [~, M] = mix(eta, L);
        delta = max(0, h(t)-(M-Mprev));
        % max clips numeric Jensen violation
        Delta = Delta + delta;
    end
end
```

```
% Returns the posterior weights and mix loss
% for learning rate eta and cumulative loss
% vector L, avoiding numerical instability.
function [w, M] = mix(eta, L)
    mn = min(L);
    if (eta == Inf) % Limit behaviour: FTL
        w = L==mn;
    else
        w = exp(-eta .* (L-mn));
    end
    s = sum(w);
    w = w / s;
    M = mn - log(s/length(L))/eta;
end
```

Figure 1: Numerically robust MATLAB implementation of AdaHedge

This lemma was first proved in its current form by Kalnishkan and Vyugin (2005, Lemma 3), and an essentially equivalent bound was introduced by Györfi and Ottucsák (2007) in the proof of their Lemma 1. Related techniques for dealing with time-varying learning rates go back to Auer et al. (2002).

Proof Using mix loss property #4, we have

$$M_T^{\mathrm{dec}} = \sum_{t=1}^{T} m_t^{\mathrm{dec}} = \sum_{t=1}^{T} \left(M_t^{(\eta_t)} - M_{t-1}^{(\eta_t)} \right) \leq \sum_{t=1}^{T} \left(M_t^{(\eta_t)} - M_{t-1}^{(\eta_{t-1})} \right) = M_T^{(\eta_T)},$$

which was to be shown. ■

We can now show that the two contributions to the regret are still balanced.

Lemma 3 *The AdaHedge regret is* $\mathcal{R}^{\mathrm{ah}} = M^{\mathrm{ah}} - L^* + \Delta^{\mathrm{ah}} \leq 2\Delta^{\mathrm{ah}}$.

Proof As $\delta_t^{\mathrm{ah}} \geq 0$ for all t (by mix loss property #1), the cumulative mixability gap Δ_t^{ah} is nondecreasing. Consequently, the AdaHedge learning rate η_t^{ah} as defined in (8) is nonincreasing in t. Thus Lemma 2 applies to M^{ah}; together with mix loss property #3 and (8) this yields

$$M^{\mathrm{ah}} \leq M^{(\eta_T^{\mathrm{ah}})} \leq L^* + \frac{\ln K}{\eta_T^{\mathrm{ah}}} = L^* + \Delta_{T-1}^{\mathrm{ah}} \leq L^* + \Delta_T^{\mathrm{ah}}.$$

Substitution into the trivial decomposition $\mathcal{R}^{\mathrm{ah}} = M^{\mathrm{ah}} - L^* + \Delta^{\mathrm{ah}}$ yields the result. ■

The remaining task is to establish a bound on Δ^{ah}. As before, we start with a bound on the mixability gap in a single round, but rather than (6), we use Bernstein's bound on the mixability gap in a single round to obtain a result that is expressed in terms of the variance of the losses, $v_t^{\mathrm{ah}} = \mathrm{Var}_{k \sim \boldsymbol{w}_t^{\mathrm{ah}}}[\ell_{t,k}] = \sum_k w_{t,k}^{\mathrm{ah}} (\ell_{t,k} - h_t^{\mathrm{ah}})^2$.

Lemma 4 (Bernstein's Bound) *Let $\eta_t = \eta_t^{\text{alg}} \in (0, \infty)$ denote the finite learning rate chosen for round t by any algorithm "*alg*". The mixability gap δ_t^{alg} satisfies*

$$\delta_t^{\text{alg}} \leq \frac{g(s_t\eta_t)}{s_t} v_t^{\text{alg}}, \quad \textit{where} \quad g(x) = \frac{e^x - x - 1}{x}. \tag{10}$$

Further, $v_t^{\text{alg}} \leq (\ell_t^+ - h_t^{\text{alg}})(h_t^{\text{alg}} - \ell_t^-) \leq s_t^2/4$.

Proof This is Bernstein's bound (Cesa-Bianchi and Lugosi, 2006, Lemma A.5) on the cumulant generating function, applied to the random variable $(\ell_{t,k} - \ell_t^-)/s_t \in [0,1]$ with k distributed according to $\boldsymbol{w}_t^{\text{alg}}$. ■

Bernstein's bound is more sophisticated than Hoeffding's bound (6), because it expresses that the mixability gap δ_t is small not only when η_t is small, but also when all experts have approximately the same loss, or when the weights $\boldsymbol{w}_t$ are concentrated on a single expert.

The next step is to use Bernstein's inequality to obtain a bound on the cumulative mixability gap Δ^{ah}. In the analysis of Cesa-Bianchi et al. (2007) this is achieved by first applying Bernstein's bound for each individual round, and then using a telescoping argument to obtain a bound on the sum. With our learning rate (8) it is convenient to reverse these steps: we first telescope, which can now be done with equality, and subsequently apply Bernstein's inequality in a stricter way.

Lemma 5 *AdaHedge's cumulative mixability gap satisfies*

$$\left(\Delta^{\text{ah}}\right)^2 \leq V^{\text{ah}} \ln K + (\tfrac{2}{3}\ln K + 1) S \Delta^{\text{ah}}.$$

Proof In this proof we will omit the superscript "ah". Using the definition of the learning rate (8) and $\delta_t \leq s_t$ (from mix loss property #1), we get

$$\begin{aligned} \Delta^2 &= \sum_{t=1}^{T} \left(\Delta_t^2 - \Delta_{t-1}^2\right) = \sum_t \left((\Delta_{t-1} + \delta_t)^2 - \Delta_{t-1}^2\right) = \sum_t \left(2\delta_t \Delta_{t-1} + \delta_t^2\right) \\ &= \sum_t \left(2\delta_t \frac{\ln K}{\eta_t} + \delta_t^2\right) \leq \sum_t \left(2\delta_t \frac{\ln K}{\eta_t} + s_t\delta_t\right) \leq 2\ln K \sum_t \frac{\delta_t}{\eta_t} + S\Delta. \end{aligned} \tag{11}$$

The inequalities in this equation replace a δ_t term by S, which is of no concern: the resulting term $S\Delta$ adds at most $2S$ to the regret bound. We will now show

$$\frac{\delta_t}{\eta_t} \leq \tfrac{1}{2} v_t + \tfrac{1}{3} s_t \delta_t. \tag{12}$$

This supersedes the bound $\delta_t/\eta_t \leq (e-2)v_t$ for $\eta_t s_t \leq 1$ used by Cesa-Bianchi et al. (2007). Even though at first sight circular, the form (12) has two major advantages. First, inclusion of the overhead $\frac{1}{3}s_t\delta_t$ will only affect smaller order terms of the regret, but admits a reduction of the leading constant to the optimal factor $\frac{1}{2}$. This gain directly percolates to our regret bounds below. Second, (12) holds for unbounded η_t, which simplifies tuning considerably.

First note that (12) is clearly valid if $\eta_t = \infty$. Assuming that η_t is finite, we can obtain this result by rewriting Bernstein's bound (10) as follows:

$$\tfrac{1}{2} v_t \geq \delta_t \cdot \frac{s_t}{2g(s_t\eta_t)} = \frac{\delta_t}{\eta_t} - s_t f(s_t\eta_t)\delta_t, \quad \text{where} \quad f(x) = \frac{e^x - \frac{1}{2}x^2 - x - 1}{xe^x - x^2 - x}.$$

Remains to show that $f(x) \leq 1/3$ for all $x \geq 0$. After rearranging, we find this to be the case if

$$(3 - x)e^x \leq \tfrac{1}{2}x^2 + 2x + 3.$$

Taylor expansion of the left-hand side around zero reveals that $(3 - x)e^x = \frac{1}{2}x^2 + 2x + 3 - \frac{1}{6}x^3 u e^u$ for some $0 \leq u \leq x$, from which the result follows. The proof is completed by plugging (12) into (11) and finally relaxing $s_t \leq S$. ■

Combination of these results yields the following natural regret bound, analogous to Theorem 5 of Cesa-Bianchi et al. (2007).

Theorem 6 *AdaHedge's regret is bounded by*

$$\mathcal{R}^{\text{ah}} \leq 2\sqrt{V^{\text{ah}} \ln K} + S(\tfrac{4}{3}\ln K + 2).$$

Proof Lemma 5 is of the form

$$(\Delta^{\text{ah}})^2 \leq a + b\Delta^{\text{ah}}, \tag{13}$$

with a and b nonnegative numbers. Solving for Δ^{ah} then gives

$$\Delta^{\text{ah}} \leq \tfrac{1}{2}b + \tfrac{1}{2}\sqrt{b^2 + 4a} \leq \tfrac{1}{2}b + \tfrac{1}{2}(\sqrt{b^2} + \sqrt{4a}) = \sqrt{a} + b,$$

which by Lemma 3 implies that

$$\mathcal{R}^{\text{ah}} \leq 2\sqrt{a} + 2b.$$

Plugging in the values $a = V^{\text{ah}} \ln K$ and $b = S(\frac{2}{3}\ln K + 1)$ from Lemma 5 completes the proof. ■

This first regret bound for AdaHedge is difficult to interpret, because the cumulative loss variance V^{ah} depends on the actions of the AdaHedge strategy itself (through the weights $\boldsymbol{w}_t^{\text{ah}}$). Below, we will derive a regret bound for AdaHedge that depends only on the data. However, AdaHedge has one important property that is captured by this first result that is no longer expressed by the worst-case bound we will derive below. Namely, if the data are easy in the sense that there is a clear best expert, say k^*, then the weights played by AdaHedge will concentrate on that expert. If $w_{t,k^*}^{\text{ah}} \to 1$ as t increases, then the loss variance must decrease: $v_t^{\text{ah}} \to 0$. Thus, Theorem 6 suggests that the AdaHedge regret may be bounded if the weights concentrate on the best expert sufficiently quickly. This indeed turns out to be the case: we can prove that the regret is bounded for the stochastic setting where the loss vectors $\boldsymbol{\ell}_t$ are independent, and $\mathrm{E}[L_{t,k^*} - L_{t,k}] = \Omega(t^\beta)$ for all $k \neq k^*$ and any $\beta > 1/2$. This is an important feature of AdaHedge when it is used as a stand-alone algorithm, and Van Erven et al. (2011) provide a proof for the previous version of the

strategy. See Section 5.4 for an example of concentration of the AdaHedge weights. Here we will not pursue this further, because the Follow-the-Leader strategy also incurs bounded loss in that case; we rather focus attention on how to successfully compete with FTL in Section 3.

We now proceed to derive a bound that depends only on the data, using an approach similar to the one taken by Cesa-Bianchi et al. (2007). We first bound the cumulative loss variance as follows:

Lemma 7 *Assume $L^* \leq H$. The cumulative loss variance for AdaHedge satisfies*

$$V^{\mathrm{ah}} \leq S\frac{(L^+ - L^*)(L^* - L^-)}{L^+ - L^-} + 2S\Delta.$$

In the degenerate case $L^- = L^+$ the fraction reads $0/0$, but since we then have $V^{\mathrm{ah}} = 0$, from here on we define the ratio to be zero in that case, which is also its limiting value.

Proof We omit all "ah" superscripts. By Lemma 4 we have $v_t \leq (\ell_t^+ - h_t)(h_t - \ell_t^-)$. Now

$$\begin{aligned} V = \sum_{t=1}^{T} v_t \leq \sum_t (\ell_t^+ - h_t)(h_t - \ell_t^-) &\leq S\sum_t \frac{(\ell_t^+ - h_t)(h_t - \ell_t^-)}{s_t} \\ = ST\sum_t \frac{1}{T}\frac{(\ell_t^+ - h_t)(h_t - \ell_t^-)}{(\ell_t^+ - h_t) + (h_t - \ell_t^-)} &\leq S\frac{(L^+ - H)(H - L^-)}{L^+ - L^-}, \end{aligned} \tag{14}$$

where the last inequality is an instance of Jensen's inequality applied to the function B defined on the domain $x, y \geq 0$ by $B(x,y) = \frac{xy}{x+y}$ for $xy > 0$ and $B(x,y) = 0$ for $xy = 0$ to ensure continuity. To verify that B is jointly concave, we will show that the Hessian is negative semi-definite on the interior $xy > 0$. Concavity on the whole domain then follows from continuity. The Hessian, which turns out to be the rank one matrix

$$\nabla^2 B(x,y) = -\frac{2}{(x+y)^3}\begin{pmatrix} y \\ -x \end{pmatrix}\begin{pmatrix} y \\ -x \end{pmatrix}^{\mathsf{T}},$$

is negative semi-definite since it is a negative scaling of a positive outer product.

Subsequently using $H \geq L^*$ (by assumption) and $H \leq L^* + 2\Delta$ (by Lemma 3) yields

$$\frac{(L^+ - H)(H - L^-)}{L^+ - L^-} \leq \frac{(L^+ - L^*)(L^* + 2\Delta - L^-)}{L^+ - L^-} \leq \frac{(L^+ - L^*)(L^* - L^-)}{L^+ - L^-} + 2\Delta$$

as desired. ■

This can be combined with Lemmas 5 and 3 to obtain our first main result:

Theorem 8 (AdaHedge Worst-Case Regret Bound) *AdaHedge's regret is bounded by*

$$\mathcal{R}^{\mathrm{ah}} \leq 2\sqrt{S\frac{(L^+ - L^*)(L^* - L^-)}{L^+ - L^-}\ln K} + S(\tfrac{16}{3}\ln K + 2). \tag{15}$$

Proof If $H^{\text{ah}} < L^*$, then $\mathcal{R}^{\text{ah}} < 0$ and the result is clearly valid. But if $H^{\text{ah}} \geq L^*$, we can bound V^{ah} using Lemma 7 and plug the result into Lemma 5 to get an inequality of the form (13) with $a = S(L^+ - L^*)(L^* - L^-)/(L^+ - L^-)$ and $b = S(\frac{8}{3}\ln K + 1)$. Following the steps of the proof of Theorem 6 with these modified values for a and b we arrive at the desired result. ■

This bound has several useful properties:

1. It is always smaller than the CBMS bound (1), with a leading constant that has been reduced from the previously best-known value of 2.63 to 2. To see this, note that (15) increases to (1) if we replace L^+ by the upper bound $L^- + ST$. It can be substantially stronger than (1) if the range of the losses s_t is highly variable.

2. The bound is "fundamental", a concept discussed in detail by Cesa-Bianchi et al. (2007): it is invariant to translations of the losses and proportional to their scale. It is therefore valid for arbitrary loss ranges, regardless of sign. In fact, not just the bound, but AdaHedge itself is fundamental in this sense: see Section 4 for a discussion and proof.

3. The regret is small when the best expert either has a very low loss, or a very high loss. The latter is important if the algorithm is to be used for a scenario in which we are provided with a sequence of gain vectors $\boldsymbol{g}_t$ rather than losses: we can transform these gains into losses using $\boldsymbol{\ell}_t = -\boldsymbol{g}_t$, and then run AdaHedge. The bound then implies that we incur small regret if the best expert has very small cumulative gain relative to the minimum gain.

4. The bound is not dependent on the number of trials but only on the losses; it is a "timeless" bound as discussed below.

2.3 What are Timeless Bounds?

All bounds presented for AdaHedge (and FlipFlop) are "timeless". We call a regret bound *timeless* if it does not change under insertion of additional trials where all experts are assigned the same loss. Intuitively, the prediction task does not become more difficult if nature should insert same-loss trials. Since these trials do nothing to differentiate between the experts, they can safely be ignored by the learner without affecting her regret; in fact, many Hedge strategies, including Hedge with a fixed learning rate, FTL, AdaHedge and CBMS already have the property that their future behaviour does not change under such insertions: they are robust against such time dilation. If any strategy does not have this property by itself, it can easily be modified to ignore equal-loss trials.

It is easy to imagine practical scenarios where this robustness property would be important. For example, suppose you hire a number of experts who continually monitor the assets in your portfolio. Usually they do not recommend any changes, but occasionally, when they see a rare opportunity or receive subtle warning signs, they may urge you to trade, resulting in a potentially very large gain or loss. It seems only beneficial to poll the experts often, and there is no reason why the many resulting equal-loss trials should complicate the learning task.

The oldest bounds for Hedge scale with $\sqrt{T}$ or $\sqrt{L^*}$, and are thus not timeless. From the results above we can obtain fundamental and timeless variants with, for parameterless algorithms, the best known leading constants (the first item below follows Corollary 1 of Cesa-Bianchi et al. 2007):

Corollary 9 *The AdaHedge regret satisfies the following inequalities:*

$$\mathcal{R}^{\mathrm{ah}} \leq \sqrt{\textstyle\sum_{t=1}^{T} s_t^2 \ln K} + S(\tfrac{4}{3}\ln K + 2) \qquad \text{(analogue of traditional } T\text{-based bounds)},$$

$$\mathcal{R}^{\mathrm{ah}} \leq 2\sqrt{S(L^* - L^-)\ln K} + S(\tfrac{16}{3}\ln K + 2) \qquad \text{(analogue of traditional } L^*\text{-based bounds)},$$

$$\mathcal{R}^{\mathrm{ah}} \leq 2\sqrt{S(L^+ - L^*)\ln K} + S(\tfrac{16}{3}\ln K + 2) \qquad \text{(symmetric bound, useful for gains)}.$$

Proof We could get a bound that depends only on the loss ranges s_t by substituting the worst case $L^* = (L^+ + L^-)/2$ into Theorem 8, but a sharper result is obtained by plugging the inequality $v_t \leq s_t^2/4$ from Lemma 4 directly into Theorem 6. This yields the first item above. The other two inequalities follow easily from Theorem 8. ■

In the next section, we show how we can compete with FTL while at the same time maintaining all these worst-case guarantees up to a constant factor.

3. FlipFlop

AdaHedge balances the cumulative mixability gap Δ^{ah} and the mix loss regret $M^{\mathrm{ah}} - L^*$ by reducing η_t^{ah} as necessary. But, as we observed previously, if the data are not hopelessly adversarial we might not need to worry about the mixability gap: as Lemma 4 expresses, δ_t^{ah} is also small if the variance v_t^{ah} of the loss under the weights $w_{t,k}^{\mathrm{ah}}$ is small, which is the case if the weight on the best expert $\max_k w_{t,k}^{\mathrm{ah}}$ becomes close to one.

AdaHedge is able to exploit such a lucky scenario to an extent: as explained in the discussion that follows Theorem 6, if the weight of the best expert goes to one quickly, AdaHedge will have a small cumulative mixability gap, and therefore, by Lemma 3, a small regret. This happens, for example, in the stochastic setting with independent, identically distributed losses, when a single expert has the smallest expected loss. Similarly, in the experiment of Section 5.4, the AdaHedge weights concentrate sufficiently quickly for the regret to be bounded.

There is the potential for a nasty feedback loop, however. Suppose there are a small number of difficult early trials, during which the cumulative mixability gap increases relatively quickly. AdaHedge responds by reducing the learning rate (8), with the effect that the weights on the experts become more uniform. As a consequence, the mixability gap in future trials may be larger than what it would have been if the learning rate had stayed high, leading to further unnecessary reductions of the learning rate, and so on. The end result may be that AdaHedge behaves as if the data are difficult and incurs substantial regret, even in cases where the regret of Hedge with a fixed high learning rate, or of Follow-the-Leader, is bounded! Precisely this phenomenon occurs in the experiment in Section 5.2 below: AdaHedge's regret is close to the worst-case bound, whereas FTL hardly incurs any regret at all.

It appears, then, that we must *either* hope that the data are easy enough that we can make the weights concentrate quickly on a single expert, by not reducing the learning rate at all; *or* we fear the worst and reduce the learning rate as much as we need to be able to provide good guarantees. We cannot really interpolate between these two extremes: an intermediate learning rate may not yield small regret in favourable cases and may at the same time destroy any performance guarantees in the worst case.

It is unclear a priori whether we can get away with keeping the learning rate high, or that it is wiser to play it safe using AdaHedge. The most extreme case of keeping the learning rate high, is the limit as η tends to ∞, for which Hedge reduces to Follow-the-Leader. In this section we work out a strategy that combines the advantages of FTL and AdaHedge: it retains AdaHedge's worst-case guarantees up to a constant factor, but its regret is also bounded by a constant times the regret of FTL (Theorem 15). Perhaps surprisingly, this is not easy to achieve. To see why, imagine a scenario where the average loss of the best expert is substantial, whereas the *regret* of either Follow-the-Leader or AdaHedge, is small. Since our combination has to guarantee a similarly small regret, it has only a very limited margin for error. We cannot, for example, simply combine the two algorithms by recursively plugging them into Hedge with a fixed learning rate, or into AdaHedge: the performance guarantees we have for those methods of combination are too weak. Even if both FTL and AdaHedge yield small regret on the original problem, choosing the actions of FTL for some rounds and those of AdaHedge for the other rounds may fail if we do it naively, because the regret is not necessarily increasing, and we may end up picking each algorithm precisely in those rounds where the other one is better.

Luckily, alternating between the optimistic FTL strategy and the worst-case-proof AdaHedge does turn out to be possible if we do it in a careful way. In this section we explain the appropriate strategy, called FlipFlop (superscript: "ff"), and show that it combines the desirable properties of both FTL and AdaHedge.

3.1 Exploiting Easy Data by Following the Leader

We first investigate the potential benefits of FTL over AdaHedge. Lemma 10 below identifies the circumstances under which FTL will perform well, which is when the number of leader changes is small. It also shows that the regret for FTL is equal to its cumulative mixability gap when FTL is interpreted as a Hedge strategy with infinite learning rate.

Lemma 10 *Let c_t be an indicator for a leader change at time t: define $c_t = 1$ if there exists an expert k such that $L_{t-1,k} = L^*_{t-1}$ while $L_{t,k} \neq L^*_t$, and $c_t = 0$ otherwise. Let $C_t = c_1 + \ldots + c_t$ be the cumulative number of leader changes. Then the FTL regret satisfies*

$$\mathcal{R}^{\text{ftl}} = \Delta^{(\infty)} \leq S\,C.$$

Proof We have $M^{(\infty)} = L^*$ by mix loss property #3, and consequently $\mathcal{R}^{\text{ftl}} = \Delta^{(\infty)} + M^{(\infty)} - L^* = \Delta^{(\infty)}$.

To bound $\Delta^{(\infty)}$, notice that, for any t such that $c_t = 0$, all leaders remained leaders and incurred identical loss. It follows that $m_t^{(\infty)} = L^*_t - L^*_{t-1} = h_t^{(\infty)}$ and hence $\delta_t^{(\infty)} = 0$. By

bounding $\delta_t^{(\infty)} \leq S$ for all other t we obtain

$$\Delta^{(\infty)} = \sum_{t=1}^{T} \delta_t^{(\infty)} = \sum_{t\colon c_t=1} \delta_t^{(\infty)} \leq \sum_{t\colon c_t=1} S = S\,C,$$

as required. ■

We see that the regret for FTL is bounded by the number of leader changes. This quantity is both fundamental and timeless. It is a natural measure of the difficulty of the problem, because it remains small whenever a single expert makes the best predictions on average, even in the scenario described above, in which AdaHedge gets caught in a feedback loop. One example where FTL outperforms AdaHedge is when the losses for two experts are $(1,0)$ on the first round, and keep alternating according to $(1,0)$, $(0,1)$, $(1,0)$, ... for the remainder of the rounds. Then the FTL regret is only $1/2$, whereas AdaHedge's performance is close to the worst-case bound (because its weights $\boldsymbol{w}_t^{\mathrm{ah}}$ converge to $(1/2,1/2)$, for which the bound (6) on the mixability gap is tight). This scenario is illustrated further in the experiments, Section 5.2.

3.2 FlipFlop

FlipFlop is a Hedge strategy in the sense that it uses exponential weights defined by (9), but the learning rate η_t^{ff} now alternates between infinity, such that the algorithm behaves like FTL, and the AdaHedge value, which decreases as a function of the mixability gap accumulated over the rounds where AdaHedge is used. In Definition 11 below, we will specify the "flip" regime $\overline{R}_t$, which is the subset of times $\{1,\ldots,t\}$ where we follow the leader by using an infinite learning rate, and the "flop" regime $\underline{R}_t = \{1,\ldots,t\} \setminus \overline{R}_t$, which is the set of times where the learning rate is determined by AdaHedge (mnemonic: the position of the bar refers to the value of the learning rate). We accumulate the mixability gap, the mix loss and the variance for these two regimes separately:

$$\overline{\Delta}_t = \sum_{\tau\in\overline{R}_t} \delta_\tau^{\mathrm{ff}}; \qquad \overline{M}_t = \sum_{\tau\in\overline{R}_t} m_\tau^{\mathrm{ff}}; \qquad \text{(flip)}$$

$$\underline{\Delta}_t = \sum_{\tau\in\underline{R}_t} \delta_\tau^{\mathrm{ff}}; \qquad \underline{M}_t = \sum_{\tau\in\underline{R}_t} m_\tau^{\mathrm{ff}}; \qquad \underline{V}_t = \sum_{\tau\in\underline{R}_t} v_\tau^{\mathrm{ff}}. \qquad \text{(flop)}$$

We also change the learning rate from its definition for AdaHedge in (8) to the following, which differentiates between the two regimes of the strategy:

$$\eta_t^{\mathrm{ff}} = \begin{cases} \eta_t^{\mathrm{flip}} & \text{if } t \in \overline{R}_t, \\ \eta_t^{\mathrm{flop}} & \text{if } t \in \underline{R}_t, \end{cases} \quad \text{where} \quad \eta_t^{\mathrm{flip}} = \eta_t^{\mathrm{ftl}} = \infty \quad \text{and} \quad \eta_t^{\mathrm{flop}} = \frac{\ln K}{\underline{\Delta}_{t-1}}. \tag{16}$$

Like for AdaHedge, $\eta_t^{\mathrm{flop}} = \infty$ as long as $\underline{\Delta}_{t-1} = 0$, which now happens for all t such that $\underline{R}_{t-1} = \emptyset$. Note that while the learning rates are defined separately for the two regimes, the exponential weights (9) of the experts are still always determined using the cumulative losses $L_{t,k}$ over *all* rounds. We also point out that, for rounds $t \in \underline{R}$, the learning rate $\eta_t^{\mathrm{ff}} = \eta_t^{\mathrm{flop}}$ is not equal to η_t^{ah}, because it uses $\underline{\Delta}_{t-1}$ instead of $\Delta_{t-1}^{\mathrm{ah}}$. For this reason, the

```
% Returns the losses of FlipFlop
% l(t,k) is the loss of expert k at time t; phi > 1 and alpha > 0 are parameters
function h = flipflop(l, alpha, phi)
    [T, K]  = size(l);
    h       = nan(T,1);
    L       = zeros(1,K);
    Delta  = [0 0];
    scale  = [phi/alpha alpha];
    regime = 1; % 1=FTL, 2=AH

    for t = 1:T
        if regime==1, eta = Inf; else eta = log(K)/Delta(2); end
        [w, Mprev] = mix(eta, L);
        h(t) = w * l(t,:)';
        L = L + l(t,:);
        [~, M] = mix(eta, L);
        delta = max(0, h(t)-(M-Mprev));
        Delta(regime) = Delta(regime) + delta;
        if Delta(regime) > scale(regime) * Delta(3-regime)
          regime = 3-regime;
        end
    end
end
```

Figure 2: FlipFlop, with new ingredients in boldface

FlipFlop regret may be either better or worse than the AdaHedge regret; our results below only preserve the regret *bound* up to a constant factor. In contrast, we do compete with the *actual* regret of FTL.

It remains to define the "flip" regime $\overline{R}_t$ and the "flop" regime $\underline{R}_t$, which we will do by specifying the times at which to switch from one to the other. FlipFlop starts optimistically, with an epoch of the "flip" regime, which means it follows the leader, until $\overline{\Delta}_t$ becomes too large compared to $\underline{\Delta}_t$. At that point it switches to an epoch of the "flop" regime, and keeps using η_t^{flop} until $\underline{\Delta}_t$ becomes too large compared to $\overline{\Delta}_t$. Then the process repeats with the next epochs of the "flip" and "flop" regimes. The regimes are determined as follows:

Definition 11 (FlipFlop's Regimes) *Let $\varphi > 1$ and $\alpha > 0$ be parameters of the algorithm (tuned below in Corollary 16). Then*

- *FlipFlop starts in the "flip" regime.*
- *If t is the earliest time since the start of a "flip" epoch where $\overline{\Delta}_t > (\varphi/\alpha)\underline{\Delta}_t$, then the transition to the subsequent "flop" epoch occurs between rounds t and $t+1$. (Recall that during "flip" epochs $\overline{\Delta}_t$ increases in t whereas $\underline{\Delta}_t$ is constant.)*
- *Vice versa, if t is the earliest time since the start of a "flop" epoch where $\underline{\Delta}_t > \alpha\overline{\Delta}_t$, then the transition to the subsequent "flip" epoch occurs between rounds t and $t+1$.*

This completes the definition of the FlipFlop strategy. See Figure 2 for a MATLAB implementation.

The analysis proceeds much like the analysis for AdaHedge. We first show that, analogously to Lemma 3, the FlipFlop regret can be bounded in terms of the cumulative mixability gap; in fact, we can use the *smallest* cumulative mixability gap that we encountered

in either of the two regimes, at the cost of slightly increased constant factors. This is the fundamental building block in our FlipFlop analysis. We then proceed to develop analogues of Lemmas 5 and 7, whose proofs do not have to be changed much to apply to FlipFlop. Finally, all these results are combined to bound the regret of FlipFlop in Theorem 15, which, after Theorem 8, is the second main result of this paper.

Lemma 12 (FlipFlop version of Lemma 3) *The following two bounds hold simultaneously for the regret of the FlipFlop strategy with parameters $\varphi > 1$ and $\alpha > 0$:*

$$\mathcal{R}^{\mathrm{ff}} \leq \left(\frac{\varphi\alpha}{\varphi-1} + 2\alpha + 1\right)\overline{\Delta} + S\left(\frac{\varphi}{\varphi-1} + 2\right); \tag{17}$$

$$\mathcal{R}^{\mathrm{ff}} \leq \left(\frac{\varphi}{\varphi-1} + \frac{\varphi}{\alpha} + 2\right)\underline{\Delta} + S. \tag{18}$$

Proof The regret can be decomposed as

$$\mathcal{R}^{\mathrm{ff}} = H^{\mathrm{ff}} - L^* = \overline{\Delta} + \underline{\Delta} + \overline{M} + \underline{M} - L^*. \tag{19}$$

Our first step will be to bound the mix loss $\overline{M} + \underline{M}$ in terms of the mix loss M^{flop} of the auxiliary strategy that uses η_t^{flop} for all t. As η_t^{flop} is nonincreasing, we can then apply Lemma 2 and mix loss property #3 to further bound

$$M^{\mathrm{flop}} \leq M^{(\eta_T^{\mathrm{flop}})} \leq L^* + \frac{\ln K}{\eta^{\mathrm{flop}}} = L^* + \underline{\Delta}_{T-1} \leq L^* + \underline{\Delta}. \tag{20}$$

Let $0 = u_1 < u_2 < \ldots < u_b < T$ denote the times just before the epochs of the "flip" regime begin, i.e. round $u_i + 1$ is the first round in the i-th "flip" epoch. Similarly let $0 < v_1 < \ldots < v_b \leq T$ denote the times just before the epochs of the "flop" regime begin, where we artificially define $v_b = T$ if the algorithm is in the "flip" regime after T rounds. These definitions ensure that we always have $u_b < v_b \leq T$. For the mix loss in the "flop" regime we have

$$\underline{M} = (M_{u_2}^{\mathrm{flop}} - M_{v_1}^{\mathrm{flop}}) + (M_{u_3}^{\mathrm{flop}} - M_{v_2}^{\mathrm{flop}}) + \ldots + (M_{u_b}^{\mathrm{flop}} - M_{v_{b-1}}^{\mathrm{flop}}) + (M^{\mathrm{flop}} - M_{v_b}^{\mathrm{flop}}). \tag{21}$$

Let us temporarily write $\eta_t = \eta_t^{\mathrm{flop}}$ to avoid double superscripts. For the "flip" regime, the properties in Lemma 1, together with the observation that η_t^{flop} does not change during the "flip" regime, give

$$\begin{aligned}
\overline{M} &= \sum_{i=1}^{b}\left(M_{v_i}^{(\infty)} - M_{u_i}^{(\infty)}\right) = \sum_{i=1}^{b}\left(M_{v_i}^{(\infty)} - L_{u_i}^*\right) \leq \sum_{i=1}^{b}\left(M_{v_i}^{(\eta_{v_i})} - L_{u_i}^*\right) \\
&\leq \sum_{i=1}^{b}\left(M_{v_i}^{(\eta_{v_i})} - M_{u_i}^{(\eta_{v_i})} + \frac{\ln K}{\eta_{v_i}}\right) = \sum_{i=1}^{b}\left(M_{v_i}^{\mathrm{flop}} - M_{u_i}^{\mathrm{flop}} + \frac{\ln K}{\eta_{u_i+1}}\right) \\
&= \left(M_{v_1}^{\mathrm{flop}} - M_{u_1}^{\mathrm{flop}}\right) + \left(M_{v_2}^{\mathrm{flop}} - M_{u_2}^{\mathrm{flop}}\right) + \ldots + \left(M_{v_b}^{\mathrm{flop}} - M_{u_b}^{\mathrm{flop}}\right) + \sum_{i=1}^{b}\underline{\Delta}_{u_i}.
\end{aligned} \tag{22}$$

From the definition of the regime changes (Definition 11), we know the value of $\underline{\Delta}_{u_i}$ very accurately at the time u_i of a change from a "flop" to a "flip" regime:

$$\underline{\Delta}_{u_i} > \alpha\overline{\Delta}_{u_i} = \alpha\overline{\Delta}_{v_{i-1}} > \varphi\underline{\Delta}_{v_{i-1}} = \varphi\underline{\Delta}_{u_{i-1}}.$$

By unrolling from low to high i, we see that

$$\sum_{i=1}^{b} \underline{\Delta}_{u_i} \leq \sum_{i=1}^{b} \varphi^{1-i} \underline{\Delta}_{u_b} \leq \sum_{i=1}^{\infty} \varphi^{1-i} \underline{\Delta}_{u_b} = \frac{\varphi}{\varphi - 1} \underline{\Delta}_{u_b}.$$

Adding up (21) and (22), we therefore find that the total mix loss is bounded by

$$\overline{M} + \underline{M} \leq M^{\text{flop}} + \sum_{i=1}^{b} \underline{\Delta}_{u_i} \leq M^{\text{flop}} + \frac{\varphi}{\varphi - 1} \underline{\Delta}_{u_b} \leq L^* + \left(\frac{\varphi}{\varphi - 1} + 1 \right) \underline{\Delta},$$

where the last inequality uses (20). Combination with (19) yields

$$\mathcal{R}^{\text{ff}} \leq \left(\frac{\varphi}{\varphi - 1} + 2 \right) \underline{\Delta} + \overline{\Delta}. \tag{23}$$

Our next goal is to relate $\underline{\Delta}$ and $\overline{\Delta}$: by construction of the regimes, they are always within a constant factor of each other. First, suppose that after T trials we are in the bth epoch of the "flip" regime, that is, we will behave like FTL in round $T + 1$. In this state, we know from Definition 11 that $\underline{\Delta}$ is stuck at the value $\underline{\Delta}_{u_b}$ that prompted the start of the current epoch. As the regime change happened after u_b, we have $\underline{\Delta}_{u_b} - S \leq \alpha \overline{\Delta}_{u_b}$, so that $\underline{\Delta} - S \leq \alpha \overline{\Delta}$. At the same time, we know that $\overline{\Delta}$ is not large enough to trigger the next regime change. From this we can deduce the following bounds:

$$\frac{1}{\alpha}(\underline{\Delta} - S) \leq \overline{\Delta} \leq \frac{\varphi}{\alpha} \underline{\Delta}.$$

On the other hand, if after T rounds we are in the bth epoch of the "flop" regime, then a similar reasoning yields

$$\frac{\alpha}{\varphi}(\overline{\Delta} - S) \leq \underline{\Delta} \leq \alpha \overline{\Delta}.$$

In both cases, it follows that

$$\underline{\Delta} < \alpha \overline{\Delta} + S;$$
$$\overline{\Delta} < \frac{\varphi}{\alpha} \underline{\Delta} + S.$$

The two bounds of the lemma are obtained by plugging first one, then the other of these bounds into (23). ∎

The "flop" cumulative mixability gap $\underline{\Delta}$ is related, as before, to the variance of the losses.

Lemma 13 (FlipFlop version of Lemma 5) *The cumulative mixability gap for the "flop" regime is bounded by the cumulative variance of the losses for the "flop" regime:*

$$\underline{\Delta}^2 \leq \underline{V} \ln K + (\tfrac{2}{3} \ln K + 1) S \underline{\Delta}. \tag{24}$$

Proof The proof is analogous to the proof of Lemma 5, with $\underline{\Delta}$ instead of Δ^{ah}, $\underline{V}$ instead of V^{ah}, and using $\eta_t = \eta_t^{\mathrm{flop}} = \ln(K)/\underline{\Delta}_{t-1}$ instead of $\eta_t = \eta_t^{\mathrm{ah}} = \ln(K)/\Delta_{t-1}^{\mathrm{ah}}$. Furthermore, we only need to sum over the rounds $\underline{R}$ in the "flop" regime, because $\underline{\Delta}$ does not change during the "flip" regime. ■

As it is straight-forward to prove an analogue of Theorem 6 for FlipFlop by solving the quadratic inequality in (24), we proceed directly towards establishing an analogue of Theorem 8. The following lemma provides the equivalent of Lemma 7 for FlipFlop. It can probably be strengthened to improve the lower order terms; we provide the version that is easiest to prove.

Lemma 14 (FlipFlop version of Lemma 7) *Suppose $H^{\mathrm{ff}} \geq L^*$. The cumulative loss variance for FlipFlop with parameters $\varphi > 1$ and $\alpha > 0$ satisfies*

$$\underline{V} \leq S\frac{(L^+ - L^*)(L^* - L^-)}{L^+ - L^-} + \left(\frac{\varphi}{\varphi - 1} + \frac{\varphi}{\alpha} + 2\right) S\underline{\Delta} + S^2.$$

Proof The sum of variances satisfies

$$\underline{V} = \sum_{t \in \underline{R}} v_t^{\mathrm{ff}} \leq \sum_{t=1}^{T} v_t^{\mathrm{ff}} \leq S\frac{(L^+ - H^{\mathrm{ff}})(H^{\mathrm{ff}} - L^-)}{L^+ - L^-},$$

where the first inequality simply includes the variances for FTL rounds (which are often all zero), and the second follows from the same reasoning as employed in (14). Subsequently using $L^* \leq H^{\mathrm{ff}}$ (by assumption) and, from Lemma 12, $H^{\mathrm{ff}} \leq L^* + \gamma$, where γ denotes the right-hand side of the bound (18), we find

$$\underline{V} \leq S\frac{(L^+ - L^*)(L^* + \gamma - L^-)}{L^+ - L^-} \leq S\frac{(L^+ - L^*)(L^* - L^-)}{L^+ - L^-} + S\gamma,$$

which was to be shown. ■

Combining Lemmas 12, 13 and 14, we obtain our second main result:

Theorem 15 (FlipFlop Regret Bound) *The regret for FlipFlop with doubling parameters $\varphi > 1$ and $\alpha > 0$ simultaneously satisfies the two bounds*

$$\mathcal{R}^{\mathrm{ff}} \leq \left(\frac{\varphi\alpha}{\varphi - 1} + 2\alpha + 1\right)\mathcal{R}^{\mathrm{ftl}} + S\left(\frac{\varphi}{\varphi - 1} + 2\right),$$

$$\mathcal{R}^{\mathrm{ff}} \leq c_1\sqrt{S\frac{(L^+ - L^*)(L^* - L^-)}{L^+ - L^-}\ln K} + c_1 S\left((c_1 + \tfrac{2}{3})\ln K + \sqrt{\ln K} + 1\right) + S,$$

where $c_1 = \dfrac{\varphi}{\varphi - 1} + \dfrac{\varphi}{\alpha} + 2.$

This shows that, up to a multiplicative factor in the regret, FlipFlop is always as good as the best of Follow-the-Leader and AdaHedge's bound from Theorem 8. Of course, if

AdaHedge significantly outperforms its bound, it is not guaranteed that FlipFlop will outperform the bound in the same way.

In the experiments in Section 5 we demonstrate that the multiplicative factor is not just an artifact of the analysis, but can actually be observed on simulated data.

Proof From Lemma 10, we know that $\overline{\Delta} \leq \Delta^{(\infty)} = \mathcal{R}^{\text{ftl}}$. Substitution in (17) of Lemma 12 yields the first inequality.

For the second inequality, note that $L^* > H^{\text{ff}}$ means the regret is negative, in which case the result is clearly valid. We may therefore assume w.l.o.g. that $L^* \leq H^{\text{ff}}$ and apply Lemma 14. Combination with Lemma 13 yields

$$\underline{\Delta}^2 \leq \underline{V} \ln K + (\tfrac{2}{3} \ln K + 1) S \underline{\Delta} \leq S \frac{(L^+ - L^*)(L^* - L^-)}{L^+ - L^-} \ln K + S^2 \ln K + c_2 S \underline{\Delta},$$

where $c_2 = (c_1 + \frac{2}{3}) \ln K + 1$. We now solve this quadratic inequality as in (13) and relax it using $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for nonnegative numbers a, b to obtain

$$\begin{aligned} \underline{\Delta} &\leq \sqrt{S \frac{(L^+ - L^*)(L^* - L^-)}{L^+ - L^-} \ln K + S^2 \ln K} + c_2 S \\ &\leq \sqrt{S \frac{(L^+ - L^*)(L^* - L^-)}{L^+ - L^-} \ln K} + S \left(\sqrt{\ln K} + c_2 \right). \end{aligned}$$

In combination with Lemma 12, this yields the second bound of the theorem. ■

Finally, we propose to select the parameter values that minimize the constant factor in front of the leading terms of these regret bounds.

Corollary 16 *The parameter values $\varphi^* = 2.37$ and $\alpha^* = 1.243$ approximately minimize the worst of the two leading factors in the bounds of Theorem 15. The regret for FlipFlop with these parameters is simultaneously bounded by*

$$\begin{aligned} &\mathcal{R}^{\text{ff}} \leq 5.64 \mathcal{R}^{\text{ftl}} + 3.73 S, \\ &\mathcal{R}^{\text{ff}} \leq 5.64 \sqrt{S \frac{(L^+ - L^*)(L^* - L^-)}{L^+ - L^-} \ln K} + S \left(35.53 \ln K + 5.64 \sqrt{\ln K} + 6.64 \right). \end{aligned}$$

Proof The leading factors $f(\varphi, \alpha) = \frac{\varphi\alpha}{\varphi - 1} + 2\alpha + 1$ and $g(\varphi, \alpha) = \frac{\varphi}{\varphi - 1} + \frac{\varphi}{\alpha} + 2$ are respectively increasing and decreasing in α. They are equalized for $\alpha(\varphi) = \big(2\varphi - 1 + \sqrt{12\varphi^3 - 16\varphi^2 + 4\varphi + 1}\big)/(6\varphi - 4)$. The analytic solution for the minimum of $f(\varphi, \alpha(\varphi))$ in φ is too long to reproduce here, but it is approximately equal to $\varphi^* = 2.37$, at which point $\alpha(\varphi^*) \approx 1.243$. ■

4. Invariance to Rescaling and Translation

A common simplifying assumption made in the literature is that the losses $\ell_{t,k}$ are translated and normalised to take values in the interval $[0, 1]$. However, doing so requires *a priori*

knowledge of the range of the losses. One would therefore prefer algorithms that do not require the losses to be normalised. As discussed by Cesa-Bianchi et al. (2007), the regret bounds for such algorithms should not change when losses are translated (because this does not change the regret) and should scale by σ when the losses are scaled by a factor $\sigma > 0$ (because the regret scales by σ). They call such regret bounds *fundamental* and show that most of the methods they introduce satisfy such fundamental bounds.

Here we go even further: it is not just our bounds that are fundamental, but also our algorithms, which do not change their output weights if the losses are scaled or translated.

Theorem 17 *Both AdaHedge and FlipFlop are invariant to translation and rescaling of the losses. Starting with losses $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_T$, obtain rescaled, translated losses $\boldsymbol{\ell}'_1, \ldots, \boldsymbol{\ell}'_T$ by picking any $\sigma > 0$ and arbitrary reals $\tau_1, \ldots, \tau_T$, and setting $\ell'_{t,k} = \sigma \ell_{t,k} + \tau_t$ for $t = 1, \ldots, T$ and $k = 1, \ldots, K$. Both AdaHedge and FlipFlop issue the exact same sequence of weights $\boldsymbol{w}'_t = \boldsymbol{w}_t$ on $\boldsymbol{\ell}'_t$ as they do on $\boldsymbol{\ell}_t$.*

Proof We annotate any quantity with a prime to denote that it is defined with respect to the losses $\boldsymbol{\ell}'_t$. We omit the algorithm name from the superscript. First consider AdaHedge. We will prove the following relations by induction on t:

$$\Delta'_{t-1} = \sigma \Delta_{t-1}; \qquad \eta'_t = \frac{\eta_t}{\sigma}; \qquad \boldsymbol{w}'_t = \boldsymbol{w}_t. \tag{25}$$

For $t = 1$, these are valid since $\Delta'_0 = \sigma \Delta_0 = 0$, $\eta'_1 = \eta_1/\sigma = \infty$, and $\boldsymbol{w}'_1 = \boldsymbol{w}_1$ are uniform. Now assume towards induction that (25) is valid for some $t \in \{1, \ldots, T\}$. We can then compute the following values from their definition: $h'_t = \boldsymbol{w}'_t \cdot \boldsymbol{\ell}'_t = \sigma h_t + \tau_t$; $m'_t = -(1/\eta'_t) \ln(\boldsymbol{w}'_t \cdot e^{-\eta'_t \boldsymbol{\ell}'_t}) = \sigma m_t + \tau_t$; $\delta'_t = h'_t - m'_t = \sigma(h_t - m_t) = \sigma \delta_t$. Thus, the mixability gaps are also related by the scale factor σ. From there we can re-establish the induction hypothesis for the next round: we have $\Delta'_t = \Delta'_{t-1} + \delta'_t = \sigma \Delta_{t-1} + \sigma \delta_t = \sigma \Delta_t$, and $\eta'_{t+1} = \ln(K)/\Delta'_t = \eta_{t+1}/\sigma$. For the weights we get $\boldsymbol{w}'_{t+1} \propto e^{-\eta'_{t+1} \boldsymbol{L}'_t} = e^{-(\eta_{t+1}/\sigma)(\sigma \boldsymbol{L}_t)} \propto \boldsymbol{w}_{t+1}$, which means the two must be equal since both sum to one. Thus the relations of (25) are also valid for time $t + 1$, proving the result for AdaHedge.

For FlipFlop, if we assume regime changes occur at the same times for $\boldsymbol{\ell}'$ and $\boldsymbol{\ell}$, then similar reasoning reveals $\overline{\Delta}'_t = \sigma \overline{\Delta}_t$; $\underline{\Delta}'_t = \sigma \underline{\Delta}_t$, $\eta'^{\text{flip}}_t = \eta^{\text{flip}}_t/\sigma = \infty$, $\eta'^{\text{flop}}_t = \eta^{\text{flop}}_t/\sigma$, and $\boldsymbol{w}'_t = \boldsymbol{w}_t$. Remains to check that the regime changes do indeed occur at the same times. Note that in Definition 11, the "flop" regime is started when $\overline{\Delta}'_t > (\varphi/\alpha)\underline{\Delta}'_t$, which is equivalent to testing $\overline{\Delta}_t > (\varphi/\alpha)\underline{\Delta}_t$ since both sides of the inequality are scaled by σ. Similarly, the "flip" regime starts when $\underline{\Delta}'_t > \alpha \overline{\Delta}'_t$, which is equivalent to the test $\underline{\Delta}_t > \alpha \overline{\Delta}_t$. ■

5. Experiments

We performed four experiments on artificial data, designed to clarify how the learning rate determines performance in a variety of Hedge algorithms. These experiments are designed to illustrate as clearly as possible the intricacies involved in the central question of this paper: whether to use a high learning rate (by following the leader) or to play it safe by using a smaller learning rate instead. Rather than mimic real-world data, on which high learning rates often seem to work well (Devaine et al., 2013), we vary the main factor that

appears to drive the best choice of learning rate: the difference in cumulative loss between the experts.

We have kept the experiments as simple as possible: the data are deterministic, and involve two experts. In each case, the data consist of one initial hand-crafted loss vector $\boldsymbol{\ell}_1$, followed by a sequence of loss vectors $\boldsymbol{\ell}_2$, ..., $\boldsymbol{\ell}_T$, which are either $(0,1)$ or $(1,0)$. For each experiment $\xi \in \{1,2,3,4\}$, we want the cumulative loss difference $L_{t,1} - L_{t,2}$ between the experts to follow a target $f_\xi(t)$, which will be a continuous, nondecreasing function of t. As the losses are binary, we cannot make $L_{t,1} - L_{t,2}$ exactly equal to the target $f_\xi(t)$, but after the initial loss $\boldsymbol{\ell}_1$, we choose every subsequent loss vector such that it brings $L_{t,1} - L_{t,2}$ as close as possible to $f_\xi(t)$. All functions f_ξ change slowly enough that $|L_{t,1} - L_{t,2} - f_\xi(t)| \leq 1$ for all t.

For each experiment, we let the number of trials be $T = 1000$, and we first plot the regret $\mathcal{R}^{(\eta)}$ of the Hedge algorithm as a function of the fixed learning rate η. We subsequently plot the regret $\mathcal{R}_t^{\text{alg}}$ as a function of $t = 1, \ldots, T$, for each of the following algorithms "alg":

1. Follow-the-Leader (Hedge with learning rate ∞)
2. Hedge with fixed learning rate $\eta = 1$
3. Hedge with the learning rate that optimizes the worst-case bound (7), which equals $\eta = \sqrt{8\ln(K)/(S^2T)} \approx 0.0745$; we will call this algorithm "safe Hedge" for brevity.
4. AdaHedge
5. FlipFlop, with parameters $\varphi^* = 2.37$ and $\alpha^* = 1.243$ as in Corollary 16
6. Variation MW by Hazan and Kale (2008), using the fixed learning rate that optimises the bound provided in their Theorem 4
7. NormalHedge, described by Chaudhuri et al. (2009)

Note that the safe Hedge strategy (the third item above) can only be used in practice if the horizon T is known in advance. Variation MW (the sixth item) additionally requires precognition of the empirical variance of the sequence of losses of the best expert up until T (that is, $\text{VAR}_T^{\max}$ as defined in Section 1.2), which is not available in practice, but which we are supplying anyway.

We include algorithms 6 and 7 because, as explained in Section 1.2, they are the state of the art in Hedge-style algorithms. Like AdaHedge, Variation MW is a refinement of the CBMS strategy described by Cesa-Bianchi et al. (2007). They modify the definition of the weights in the Hedge algorithm to include second-order terms; the resulting bound is never more than a constant factor worse than the bounds (1) for CBMS and (15) for AdaHedge, but for some easy data it can be substantially better. For this reason it is a natural performance target for AdaHedge.

The bounds for CBMS and AdaHedge are incomparible with the bound for NormalHedge, being better for some, worse for other data. The reason we include it in the experiments is because, compared to the other methods, its performance in practice turns out to be excellent. We do not know whether there are data sequences on which FlipFlop significantly outperforms NormalHedge, nor whether there is a theoretical reason for this good performance, as the NormalHedge bound (Chaudhuri et al., 2009) is not tight for our experiments.

To reduce clutter, we omit results for CBMS; its behaviour is very similar to that of AdaHedge. Below we provide an exact description of each experiment, and discuss the results.

5.1 Experiment 1. Worst Case for FTL

The experiment is defined by $\boldsymbol{\ell}_1 = (\frac{1}{2}\ 0)$, and $f_1(t) = 0$. This yields the following losses:

$$\begin{pmatrix}1/2\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \ldots$$

These data are the worst case for FTL: each round, the leader incurs loss one, while each of the two individual experts only receives a loss once every two rounds. Thus, the FTL regret increases by one every two rounds and ends up around 500. For any learning rate η, the weights used by the Hedge algorithm are repeated every two rounds, so the regret $H_t - L_t^*$ increases by the same amount every two rounds: the regret increases linearly in t for every fixed η that does not vary with t. However, the constant of proportionality can be reduced greatly by reducing the value of η, as the top graph in Figure 3 shows: for $T = 1000$, the regret becomes negligible for any η less than about 0.01. Thus, in this experiment, a learning algorithm must reduce the learning rate to shield itself from incurring an excessive overhead.

The bottom graph in Figure 3 shows the expected breakdown of the FTL algorithm; Hedge with fixed learning rate $\eta = 1$ also performs quite badly. When η is reduced to the value that optimises the worst-case bound, the regret becomes competitive with that of the other algorithms. Note that Variation MW has the best performance; this is because its learning rate is tuned in relation to the bound proved in the paper, which has a relatively large constant in front of the leading term. As a consequence the algorithm always uses a relatively small learning rate, which turns out to be helpful in this case but harmful in later experiments.

FlipFlop behaves as theory suggests it should: its regret increases alternately like the regret of AdaHedge and the regret of FTL. The latter performs horribly, so during those intervals the regret increases quickly, on the other hand the FTL intervals are relatively short-lived so in the end they do not harm the regret by more than a constant factor.

The NormalHedge algorithm still has acceptable performance, although its regret is relatively large in this experiment; we have no explanation for this but in fairness we do observe good performance of NormalHedge in the other three experiments as well as in numerous further unreported simulations.

5.2 Experiment 2. Best Case for FTL

The second experiment is defined by $\boldsymbol{\ell}_1 = (1, 0)$ and $f_2(t) = 3/2$. This leads to the sequence of losses

$$\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \ldots$$

in which the loss vectors are alternating for $t \geq 2$. These data look very similar to the first experiment, but as the top graph in Figure 4 illustrates, because of the small changes at

the start of the sequence, it is now viable to reduce the regret by using a very *high* learning rate. In particular, since there are no leader changes after the first round, FTL incurs a regret of only 1/2.

As in the first experiment, the regret increases linearly in t for every fixed η (provided it is less than ∞); but now the constant of linearity is large only for learning rates close to 1. Once FlipFlop enters the FTL regime for the second time, it stays there indefinitely, which results in bounded regret. After this small change in the setup compared to the previous experiment, NormalHedge also suddenly adapts very well to the data. The behaviour of the other algorithms is very similar to the first experiment: their regret grows without bound.

5.3 Experiment 3. Weights do not Concentrate in AdaHedge

The third experiment uses $\boldsymbol{\ell}_1 = (1, 0)$, and $f_3(t) = t^{0.4}$. The first few loss vectors are the same as in the previous experiment, but every now and then there are two loss vectors $(1, 0)$ in a row, so that the first expert gradually falls behind the second in terms of performance. By $t = T = 1000$, the first expert has accumulated 508 loss, while the second expert has only 492.

For any fixed learning rate η, the weights used by Hedge now concentrate on the second expert. We know from Lemma 4 that the mixability gap in any round t is bounded by a constant times the variance of the loss under the weights played by the algorithm; as these weights concentrate on the second expert, this variance must go to zero. One can show that this happens quickly enough for the cumulative mixability gap to be *bounded* for any fixed η that does not vary with t or depend on T. From (5) we have

$$\mathcal{R}^{(\eta)} = M - L^* + \Delta^{(\eta)} \leq \frac{\ln K}{\eta} + \text{bounded} = \text{bounded}.$$

So in this scenario, as long as the learning rate is kept fixed, we will eventually learn the identity of the best expert. However, if the learning rate is very small, this will happen so slowly that the weights still have not converged by $t = 1000$. Even worse, the top graph in Figure 5 shows that for intermediate values of the learning rate, not only do the weights fail to converge on the second expert sufficiently quickly, but they are sensitive enough to the alternation of the loss vectors to increase the overhead incurred each round.

For this experiment, it really pays to use a large learning rate rather than a safe small one. Thus FTL, Hedge with $\eta = 1$, FlipFlop and NormalHedge perform excellently, while safe Hedge, AdaHedge and Variation MW incur a substantial overhead. Extrapolating the trend in the graph, it appears that the overhead of these algorithms is *not* bounded. This is possible because the three algorithms with poor performance use a learning rate that decreases as a function of t. As a concequence the used learning rate may remain too small for the weights to concentrate. For the case of AdaHedge, this is an example of the "nasty feedback loop" described in Section 3.

5.4 Experiment 4. Weights do Concentrate in AdaHedge

The fourth and last experiment uses $\boldsymbol{\ell}_1 = (1, 0)$, and $f_4(t) = t^{0.6}$. The losses are comparable to those of the third experiment, but the performance gap between the two experts is somewhat larger. By $t = T = 1000$, the two experts have loss 532 and 468, respectively. It

is now so easy to determine which of the experts is better that the top graph in Figure 6 is nonincreasing: the larger the learning rate, the better.

The algorithms that managed to keep their regret bounded in the previous experiment obviously still perform very well, but it is clearly visible that AdaHedge now achieves the same. As discussed below Theorem 6, this happens because the weight concentrates on the second expert quickly enough that AdaHedge's regret is bounded in this setting. The crucial difference with the previous experiment is that now we have $f_\xi(t) = t^\beta$ with $\beta > 1/2$. Thus, while the previous experiment shows that AdaHedge can be tricked into reducing the learning rate while it would be better not to do so, the present experiment shows that on the other hand, sometimes AdaHedge does adapt really nicely to easy data, in contrast to algorithms that are tuned in terms of a worst-case bound.

6. Discussion and Conclusion

The main contributions of this work are twofold. First, we develop a new hedging algorithm called AdaHedge. The analysis simplifies existing results and we obtain improved bounds (Theorems 6 and 8). Moreover, AdaHedge is "fundamental" in the sense that its weights are invariant under translation and scaling of the losses (Section 4) and its bounds are "timeless" in the sense that they do not degenerate when rounds are inserted in which all experts incur the same loss. Second, we explain in detail why it is difficult to tune the learning rate such that good performance is obtained both for easy and for hard data, and we address the issue by developing the FlipFlop algorithm. FlipFlop never performs much worse than the Follow-the-Leader strategy, which works very well on easy data (Lemma 10), but it also retains a worst-case bound similar to the bound for AdaHedge (Theorem 15). As such, this work may be seen as solving a special case of a more general question: can we compete with Hedge for any fixed learning rate? We will now briefly discuss this question and then place our work in a broader context, which provides an ambitious agenda for future work.

6.1 General Question: Competing with Hedge for any Fixed Learning Rate

Up to multiplicative constants, FlipFlop is at least as good as FTL and as (the bound for) AdaHedge. These two algorithms represent two extremes of choosing the learning rate η_t in Hedge: FTL takes $\eta_t = \infty$ to exploit easy data, whereas AdaHedge decreases η_t with t to protect against the worst case. It is now natural to ask whether we can design a "Universal Hedge" algorithm that can compete with Hedge with any fixed learning rate $\eta \in (0, \infty]$. That is, for all T, the regret up to time T of Universal Hedge should be within a constant factor C of the regret incurred by Hedge run with the fixed $\hat{\eta}$ that minimizes the Hedge loss $H^{(\hat{\eta})}$. This appears to be a difficult question, and maybe such an algorithm does not even exist. Yet, even partial results (such as an algorithm that competes with $\eta \in [\sqrt{\ln(K)/(S^2T)}, \infty]$ or with a factor C that increases slowly, say, logarithmically, in T) would already be of significant interest.

In this regard, it is interesting to note that, in practice, the learning rates chosen by sophisticated versions of Hedge do not always perform very well; higher learning rates often do better. This is noted by Devaine et al. (2013), who resolve the issue by adapting the learning rate sequentially in an ad-hoc fashion, which works well in their application, but

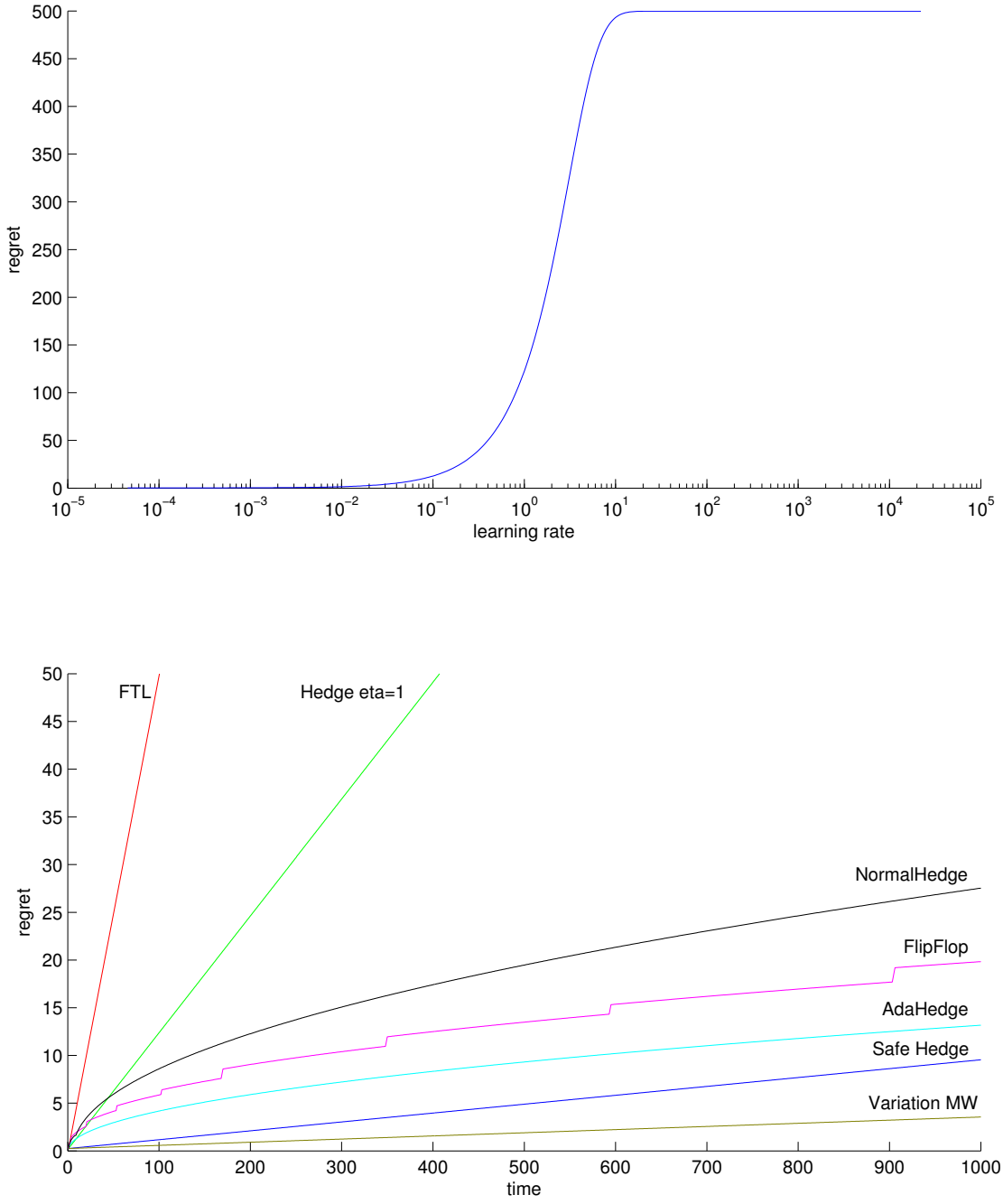


Figure 3: Hedge regret for Experiment 1 (FTL worst-case)

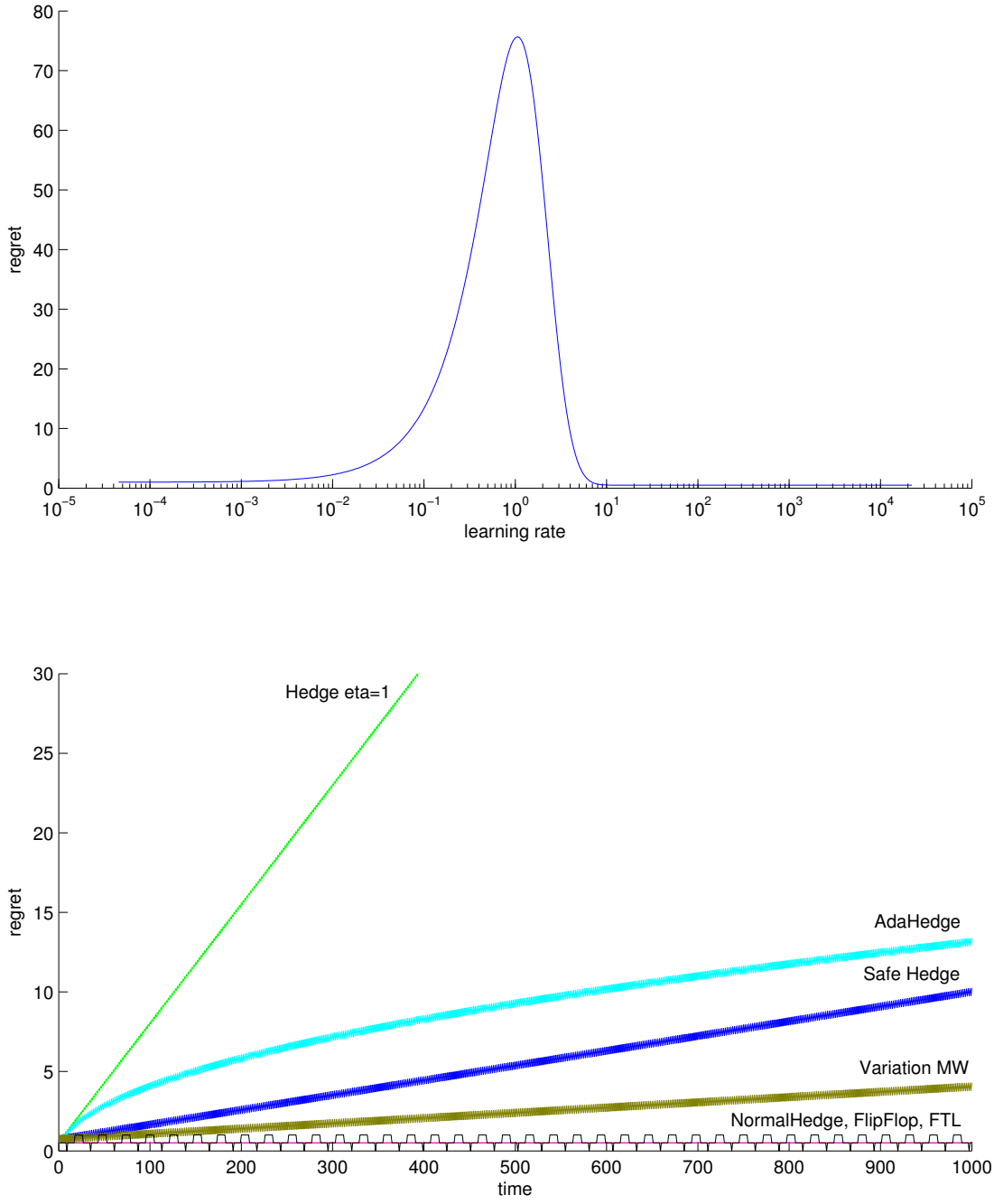


Figure 4: Hedge regret for Experiment 2 (FTL best-case)

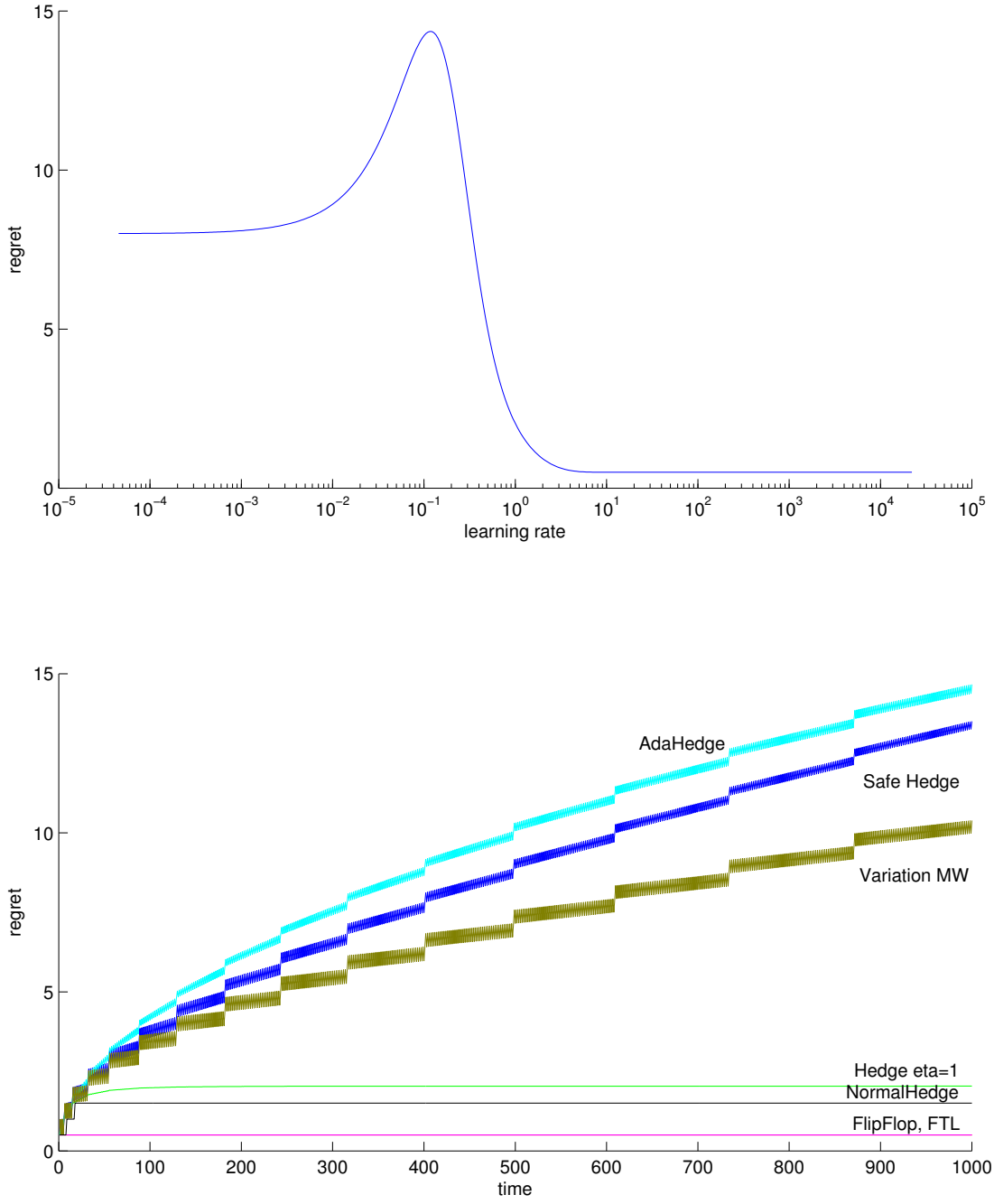


Figure 5: Hedge regret for Experiment 3 (weights do not concentrate in AdaHedge)

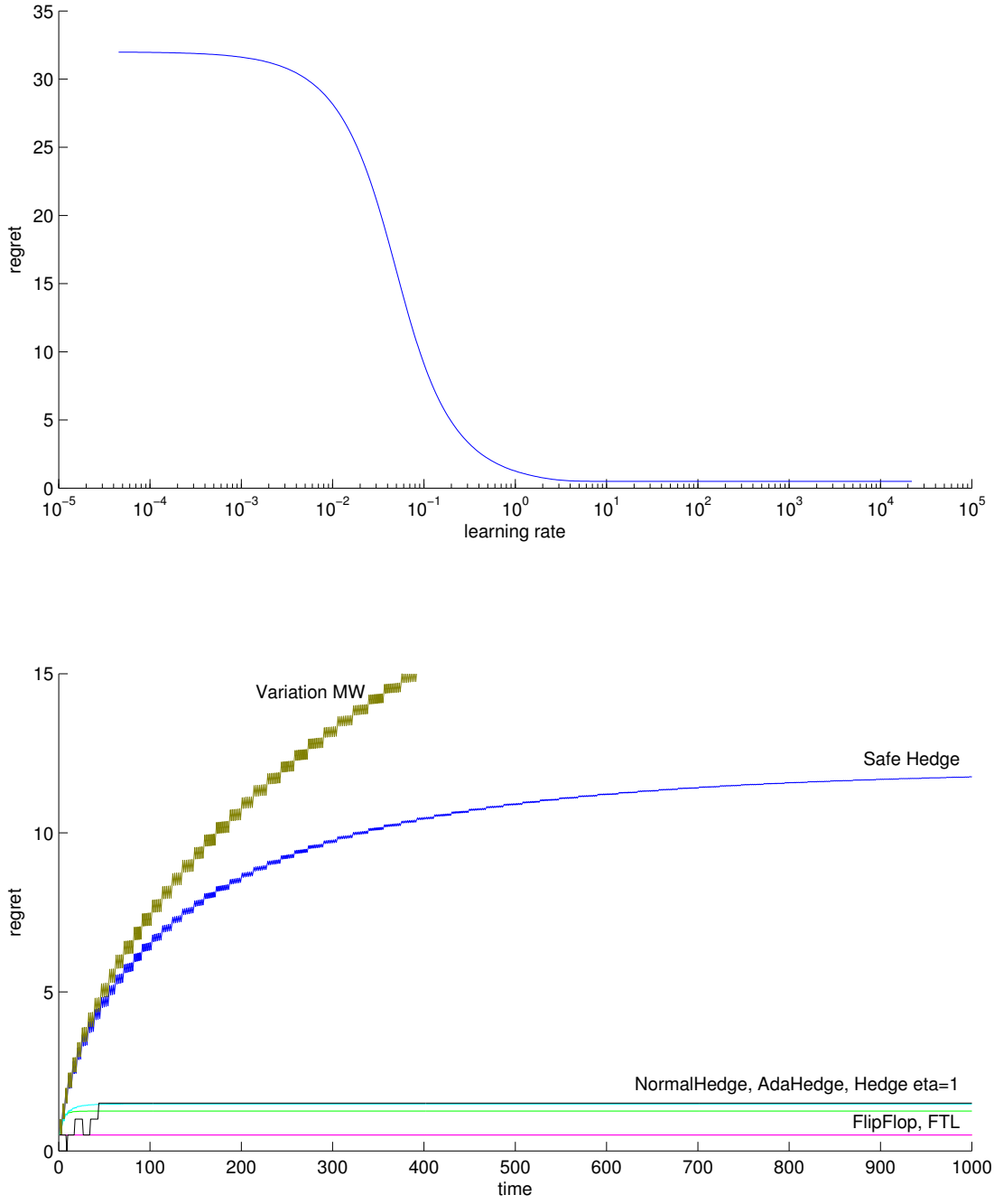


Figure 6: Hedge regret for Experiment 4 (weights do concentrate in AdaHedge)

for which they can provide no guarantees. A Universal Hedge algorithm would adapt to the learning rate that is optimal with hindsight. FlipFlop is a first step in this direction. Indeed, it already has some of the properties of such an ideal algorithm: under some conditions we can show that if Hedge achieves bounded regret using *any* learning rate, then FTL, and therefore FlipFlop, also achieves bounded regret:

Theorem 18 *Fix any $\eta > 0$. For $K = 2$ experts with losses in $\{0, 1\}$ we have*

$$\mathcal{R}^{(\eta)} \text{ is bounded } \Rightarrow \mathcal{R}^{\text{ftl}} \text{ is bounded } \Rightarrow \mathcal{R}^{\text{ff}} \text{ is bounded.}$$

The proof is in Appendix B. While the second implication remains valid for more experts and other losses, we currently do not know if the first implication continues to hold as well.

6.2 The Big Picture

Broadly speaking, a "learning rate" is any single scalar parameter controlling the relative weight of the data and a prior regularization term in a learning task. Such learning rates pop up in batch settings as diverse as L_1/L_2-regularized regression such as Lasso and Ridge, standard Bayesian nonparametric and PAC-Bayesian inference (Zhang, 2006; Audibert, 2004; Catoni, 2007), and—as in this paper—in sequential prediction. All the applications just mentioned can formally be seen as variants of Bayesian inference: Bayesian MAP in Lasso and Ridge, randomized drawing from the posterior ("Gibbs sampling") in the PAC-Bayesian setting and in the Hedge setting. Moreover, in each of these applications, selecting the appropriate learning rate is nontrivial: simply adding the learning rate as another parameter and putting a Bayesian prior on it can lead to very bad results (Grünwald and Langford, 2007). An ideal method for adapting the learning rate would work in all such applications. In addition to the FlipFlop algorithm described here, we currently have methods that are guaranteed to work for several PAC-Bayesian style stochastic settings (Grünwald, 2011, 2012). It is encouraging that all these methods are based on the same, apparently fundamental, quantity, the *mixability gap* as defined before Lemma 1: they all employ different techniques to ensure a learning rate under which the posterior is concentrated and hence the mixability gap is small. This gives some hope that the approach can be taken even further. To give but one example, the "Safe Bayesian" method of Grünwald (2012) uses essentially the same technique as Devaine et al. (2013), with an additional online-to-batch conversion step. Grünwald (2012) proves that this approach adapts to the optimal learning rate in an i.i.d. stochastic setting with arbitrary (countably or uncountably infinite) sets of "experts" (predictors); in contrast, AdaHedge and FlipFlop in the form presented in this paper are suitable for a worst-case setting with a finite set of experts. This raises, of course, the question of whether either the Safe Bayesian method can be extended to the worst-case setting (which would imply formal guarantees for the method of Devaine et al. 2013), or the FlipFlop algorithm can be extended to the setting with infinitely many experts.

Thus, we have two major, interrelated questions for future work: first, as explained in Section 6.1, we would like to be able to compete with all η in some set that contains a whole range rather than just two values. Second, we would like to compete with the best η in a setting with a countably infinite or even uncountable number of experts equipped with an arbitrary prior distribution.

A third question for future work is whether our methods can be extended beyond the standard worst-case Hedge setting and the stochastic i.i.d. setting. A particularly intriguing (and, as initial research suggests, nontrivial) question is whether AdaHedge and FlipFlop can be adapted to settings with limited feedback such as the adversarial *bandit setting* (Cesa-Bianchi and Lugosi, 2006). We would also like to extend our approach to the Hedge-based strategies for combinatorial decision domains like *Component Hedge* by Koolen et al. (2010), and for matrix-valued predictions like those by Tsuda et al. (2005).

Acknowledgments

We would like to thank Wojciech Kotłowski, Gilles Stoltz and two anonymous referees for critical feedback. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778 and by NWO Rubicon grants 680-50-1010 and 680-50-1112.

Appendix A. Proof of Lemma 1

The result for $\eta = \infty$ follows from $\eta < \infty$ as a limiting case, so we may assume without loss of generality that $\eta < \infty$. Then $m_t \leq h_t$ is obtained by using Jensen's inequality to move the logarithm inside the expectation, and $m_t \geq \ell_t^-$ and $h_t \leq \ell_t^+$ follow by bounding all losses by their minimal and maximal values, respectively. The next two items are analogues of similar basic results in Bayesian probability. Item 2 generalizes the chain rule of probability $\Pr(x_1, \ldots, x_T) = \prod_{t=1}^T \Pr(x_t \mid x_1, \ldots, x_{t-1})$:

$$M = -\frac{1}{\eta} \ln \prod_{t=1}^{T} \frac{\boldsymbol{w}_1 \cdot e^{-\eta \boldsymbol{L}_t}}{\boldsymbol{w}_1 \cdot e^{-\eta \boldsymbol{L}_{t-1}}} = -\frac{1}{\eta} \ln(\boldsymbol{w}_1 \cdot e^{-\eta \boldsymbol{L}}).$$

For the third item, use item 2 to write

$$M = -\frac{1}{\eta} \ln \left(\sum_k w_{1,k} e^{-\eta L_{T,k}} \right).$$

The lower bound is obtained by bounding all $L_{T,k}$ from below by L^*; for the upper bound we drop all terms in the sum except for the term corresponding to the best expert and use $w_{1,k} = 1/K$.

For the last item, let $0 < \eta < \gamma$ be any two learning rates. Then Jensen's inequality gives

$$-\frac{1}{\eta} \ln \boldsymbol{w}_1 \cdot e^{-\eta \boldsymbol{L}} = -\frac{1}{\eta} \ln \boldsymbol{w}_1 \cdot \left(e^{-\gamma \boldsymbol{L}}\right)^{\eta/\gamma} \geq -\frac{1}{\eta} \ln \left(\boldsymbol{w}_1 \cdot e^{-\gamma \boldsymbol{L}}\right)^{\eta/\gamma} = -\frac{1}{\gamma} \ln \boldsymbol{w}_1 \cdot e^{-\gamma \boldsymbol{L}}.$$

This completes the proof. ■

Appendix B. Proof of Theorem 18

The second implication follows from Theorem 15, so we only need to prove the first implication. To this end, consider any infinite sequence of losses on which FTL has unbounded regret. We will argue that Hedge with fixed η must have unbounded regret as well.

Our argument is based on finding an infinite subsequence of the losses on which (a) the regret for Hedge with fixed η is at most as large as on the original sequence of losses; and (b) the regret for Hedge is infinite.

To construct this subsequence, first remove all trials t such that $\ell_{t,1} = \ell_{t,2}$ (that is, both experts suffer the same loss), as these trials do not change the regret of either FTL or Hedge, nor their behaviour on any of the other rounds.

Next, we will selectively remove certain local extrema. We call a pair of two consecutive trials $(t, t+1)$ a *local extremum* if the losses in these trials are opposite: either $\boldsymbol{\ell}_t = (0,1)$ and $\boldsymbol{\ell}_{t+1} = (1,0)$ or vice versa. Removing any local extremum will only decrease the regret for Hedge, as may be seen as follows.

We observe that removing a local extremum will not change the cumulative losses of the experts or the behaviour of Hedge on other rounds, so it suffices to consider only the regret incurred on rounds t and $t+1$ themselves. By symmetry it is further sufficient to consider the case that $\boldsymbol{\ell}_t = (0,1)$ and $\boldsymbol{\ell}_{t+1} = (1,0)$. Then, over trials t and $t+1$, the individual experts both suffer loss 1, and for Hedge the loss is $h_t + h_{t+1} = \boldsymbol{w}_t \cdot \boldsymbol{\ell}_t + \boldsymbol{w}_{t+1} \cdot \boldsymbol{\ell}_{t+1} = w_{t,2} + w_{t+1,1}$. Now, since the loss received by expert 1 in round t was less than that of expert 2, some weight shifts to the first expert: we must have $w_{t+1,1} > w_{t,1}$. Substitution gives $h_t + h_{t+1} > w_{t,1} + w_{t,2} = 1$. Thus, Hedge suffers more loss in these two rounds than whichever expert turns out to be best in hindsight, and it follows that removing trials t and $t+1$ will only decrease its regret (by an amount that depends only on η).

We proceed to select the local extrema to remove. To this end, let $d_t = L_{t,2} - L_{t,1}$ denote the difference in cumulative loss between the experts after t trials, and observe that removal of a local extremum at $(t, t+1)$ will simply remove the elements d_t and d_{t+1} from the sequence $d_1, d_2, \ldots$ while leaving the other elements of the sequence unchanged. We will remove local extrema in a way that leads to an infinite subsequence of losses such that

$$d_1, d_2, d_3, d_4, d_5, \ldots = \pm 1,\ 0, \pm 1,\ 0, \pm 1, \ldots \tag{26}$$

In this subsequence, every two consecutive trials still constitute a local extremum, on which Hedge incurs a certain fixed positive regret. Consequently, the Hedge regret $\mathcal{R}_t$ grows linearly in t and is therefore unbounded.

If the losses already satisfy (26), we are done. If not, then observe that there can only be a leader change at time $t+1$ in the sense of Lemma 10 when $d_t = 0$. Since the FTL regret is bounded by the number of leader changes (Lemma 10), and since FTL was assumed to have infinite regret, there must therefore be an infinite number of trials t such that $d_t = 0$. We will remove local extrema in a way that preserves this property. In addition, we must have $|d_{t+1} - d_t| = 1$ for all t, because $d_{t+1} = d_t$ would imply that $\ell_{t+1,1} = \ell_{t+1,2}$ and we have already removed such trials. This second property is automatically preserved regardless of which trials we remove.

If the losses do not yet satisfy (26), there must be a first trial u with $|d_u| \geq 2$. Since there are infinitely many t with $d_t = 0$, there must then also be a first trial $w > u$ with $d_w = 0$. Now choose any $v \in [u, w)$ so that $|d_v| = \max_{t \in [u,w]} |d_t|$ maximizes the discrepancy between the cumulative losses of the experts. Since v attains the maximum and $|d_{t+1} - d_t| = 1$ for all t as mentioned above, we have $|d_{v+1}| = |d_v| - 1$, so that $(v, v+1)$ must be a local extremum, and this is the local extremum we remove. Since $|d_v| \geq |d_u| \geq 2$, we also have $|d_{v+1}| \geq 1$, so that this does not remove any of the trials in which $d_t = 0$. Repetition of this

process will eventually lead to $v = u$, so that trial u is removed. Given any T, the process may therefore be repeated until $|d_t| \leq 1$ for all $t \leq T$. As $|d_{t+1} - d_t| = 1$ for all t, we then match (26) for the first T trials. Hence by letting T go to infinity we obtain the desired result. ■

References

Jean-Yves Audibert. *PAC-Bayesian statistical learning theory.* PhD thesis, Université Paris VI, 2004.

Peter Auer, Nicolò Cesa-Bianchi, and Claudio Gentile. Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, 64:48–75, 2002.

Olivier Catoni. *PAC-Bayesian Supervised Classification.* Lecture Notes-Monograph Series. IMS, 2007.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games.* Cambridge University Press, 2006.

Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, 1997.

Nicolò Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2/3):321–352, 2007.

Kamalika Chaudhuri, Yoav Freund, and Daniel Hsu. A parameter-free hedging algorithm. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 297–305, 2009.

Alexey V. Chernov and Vladimir Vovk. Prediction with advice of unknown number of experts. In Peter Grünwald and Peter Spirtes, editors, *UAI*, pages 117–125. AUAI Press, 2010.

Marie Devaine, Pierre Gaillard, Yannig Goude, and Gilles Stoltz. Forecasting electricity consumption by aggregating specialized experts; a review of the sequential aggregation of specialized experts, with an application to Slovakian and French country-wide one-day-ahead (half-)hourly predictions. *Machine Learning*, 90(2):231–260, February 2013.

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*, 55:119–139, 1997.

Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.

Sébastien Gerchinovitz. *Prédiction de suites individuelles et cadre statistique classique: étude de quelques liens autour de la régression parcimonieuse et des techniques d'agrégation.* PhD thesis, Université Paris-Sud, 2011.

Peter Grünwald. Safe learning: bridging the gap between Bayes, MDL and statistical learning theory via empirical convexity. In *Proceedings of the 24th International Conference on Learning Theory (COLT 2011)*, pages 551–573, 2011.

Peter Grünwald. The safe Bayesian: learning the learning rate via the mixability gap. In *Proceedings of the 23rd International Conference on Algorithmic Learning Theory (ALT 2012)*, 2012.

Peter Grünwald and John Langford. Suboptimal behavior of Bayes and MDL in classification under misspecification. *Machine Learning*, 66(2-3):119–149, 2007. DOI 10.1007/s10994-007-0716-7.

László Györfi and György Ottucsák. Sequential prediction of unbounded stationary time series. *IEEE Transactions on Information Theory*, 53(5):1866–1872, 2007.

Elad Hazan and Satyen Kale. Extracting certainty from uncertainty: Regret bounded by variation in costs. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT)*, pages 57–67, 2008.

Marcus Hutter and Jan Poland. Adaptive online prediction by following the perturbed leader. *Journal of Machine Learning Research*, 6:639–660, 2005.

Adam Kalai and Santosh Vempala. Efficient algorithms for online decision. In *Proceedings of the 16st Annual Conference on Learning Theory (COLT)*, pages 506–521, 2003.

Yuri Kalnishkan and Michael V. Vyugin. The weak aggregating algorithm and weak mixability. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT)*, pages 188–203, 2005.

Wouter M. Koolen, Manfred K. Warmuth, and Jyrki Kivinen. Hedging structured concepts. In A.T. Kalai and M. Mohri, editors, *Proceedings of the 23rd Annual Conference on Learning Theory (COLT 2010)*, pages 93–105, 2010.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

Koji Tsuda, Gunnar Rätsch, and Manfred K. Warmuth. Matrix exponentiated gradient updates for on-line learning and Bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.

Tim van Erven, Peter Grünwald, Wouter M. Koolen, and Steven de Rooij. Adaptive hedge. In *Advances in Neural Information Processing Systems 24 (NIPS 2011)*, pages 1656–1664, 2011.

Vladimir Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56(2):153–173, 1998.

Vladimir Vovk. Competitive on-line statistics. *International Statistical Review*, 69(2):213–248, 2001.

Vladimir Vovk, Akimichi Takemura, and Glenn Shafer. Defensive forecasting. In *Proceedings of AISTATS 2005*, 2005. Archive version available at `http://www.vovk.net/df`.

Tong Zhang. Information theoretical upper and lower bounds for statistical estimation. *IEEE Transactions on Information Theory*, 52(4):1307–1321, 2006.

Journal of Machine Learning Research 14 (2014) 1317-1350 Submitted 6/13; Published 4/14

Structured Prediction via Output Space Search

Janardhan Rao Doppa DOPPA@EECS.OREGONSTATE.EDU
Alan Fern AFERN@EECS.OREGONSTATE.EDU
Prasad Tadepalli TADEPALL@EECS.OREGONSTATE.EDU
School of EECS, Oregon State University
Corvallis, OR 97331-5501, USA

Editor: S.V.N. Vishwanathan

Abstract

We consider a framework for structured prediction based on search in the space of complete structured outputs. Given a structured input, an output is produced by running a time-bounded search procedure guided by a learned cost function, and then returning the least cost output uncovered during the search. This framework can be instantiated for a wide range of search spaces and search procedures, and easily incorporates arbitrary structured-prediction loss functions. In this paper, we make two main technical contributions. First, we describe a novel approach to automatically defining an effective search space over structured outputs, which is able to leverage the availability of powerful classification learning algorithms. In particular, we define the limited-discrepancy search space and relate the quality of that space to the quality of learned classifiers. We also define a sparse version of the search space to improve the efficiency of our overall approach. Second, we give a generic cost function learning approach that is applicable to a wide range of search procedures. The key idea is to learn a cost function that attempts to mimic the behavior of conducting searches guided by the true loss function. Our experiments on six benchmark domains show that a small amount of search in limited discrepancy search space is often sufficient for significantly improving on state-of-the-art structured-prediction performance. We also demonstrate significant speed improvements for our approach using sparse search spaces with little or no loss in accuracy.

Keywords: structured prediction, state space search, imitation learning, cost function

1. Introduction

Research in machine learning has progressed from studying isolated classification tasks to the study of tasks such as information extraction and scene understanding where the inputs and outputs are structured objects. The general field of structured prediction deals with these kind of structured tasks where the goal is to learn a predictor that must produce a complex structured output given a complex structured input. A prototypical example is Part-of-Speech (POS) tagging, where the structured input is a sequence of words and structured output corresponds to the POS tags for those words. Another example is that of image scene labeling, where the structured input is an image and the structured output is a semantic labeling of the image regions.

A typical approach to structured prediction is to learn a cost function $\mathcal{C}(\mathbf{x}, \mathbf{y})$ for scoring a potential structured output $\mathbf{y}$ given a structured input $\mathbf{x}$. Given such a cost function and

a new input $\mathbf{x}$, the output computation then involves solving the so-called Argmin problem:

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y}} \mathcal{C}(\mathbf{x}, \mathbf{y}).$$

For example, approaches such as Conditional Random Fields (CRFs) (Lafferty et al., 2001), Max-Margin Markov Networks (Taskar et al., 2003) and Structured SVMs (Tsochantaridis et al., 2004) represent the cost function as a linear model over template features of both $\mathbf{x}$ and $\mathbf{y}$. Unfortunately exactly solving the Argmin problem is often intractable and efficient solutions exist only in limited cases such as when the dependency structure among features forms a tree. In such cases, one might simplify the features to allow for tractable inference, which can be detrimental to prediction accuracy. Alternatively, a heuristic optimization method can be used such as loopy belief propagation or variational inference. While such methods have shown some success in practice, it can be difficult to characterize their solutions and to predict when they are likely to work well for a new problem.

In this paper, we study a new search-based approach to structured prediction based on search in the space of complete outputs. The approach involves first defining a combinatorial search space over complete structured outputs that allows for traversal of the output space. Next, given a structured input, a state-based search strategy (e.g., best-first or greedy search), guided by a learned cost function, is used to explore the space of outputs for a specified time bound. The least cost output uncovered by the search is then returned as the prediction. This approach is motivated by our observation that for a variety of structured prediction problems, if we use the true loss function of the structured prediction problem to guide the search (even non-decomposable losses), then high-quality outputs are found very quickly. This suggests that similar performance might be achieved if we could learn an appropriate cost function to guide search in place of the true loss function.

A potential advantage of our search-based approach, compared to most structured-prediction approaches (see Section 6), is that it scales gracefully with the complexity of the cost function dependency structure. In particular, the search procedure only needs to be able to efficiently evaluate the cost function at specific input-output pairs, which is generally straightforward even when the corresponding Argmin problem is intractable. Thus, we are free to increase the complexity of the cost function without considering its impact on the inference complexity. Another potential benefit of our approach is that since the search is over complete outputs, our inference is inherently an anytime procedure, meaning that it can be stopped at any time and return the best output discovered so far. This has the flexibility of allowing for the use of more or less inference time for computing outputs as dictated by the application, with the idea that more inference time may sometimes allow for higher quality outputs.

The effectiveness of our approach for a particular problem depends critically on: 1) The identification of an effective combination of search space and search strategy over structured outputs, and 2) Our ability to learn a cost function for effectively guiding the search for high quality outputs. The main contribution of our work is to provide generic solutions to these two issues and to demonstrate their empirical effectiveness.

First, we describe the limited-discrepancy search space, as a generic search space over complete outputs that can be customized to a particular problem by leveraging the power of (non-structured) classification learning algorithms. We show that the quality of this search

space is directly related to the error of the learned classifiers and can be quite favorable compared to more naive search space definitions. We also define a sparse version of the search space to improve the efficiency of our approach. The sparse search spaces tries to reduce the branching factor while not hurting the quality of the search space too much.

Our second contribution is to describe a generic cost function learning algorithm that can be instantiated for a wide class of "ranking-based search strategies." The key idea is to learn a cost function that allows for imitating the search behavior of the algorithm when guided by the true loss function. We give a PAC bound for the approach in the realizable setting, showing a polynomial sample complexity for doing approximately as well as when guiding search with the true loss function.

Finally, we provide experimental results for our approach on a number of benchmark problems and show that even when using a relatively small amount of search, the performance is comparable or better than the state-of-the-art in structured prediction. We also demonstrate significant speed improvements of our approach when used with sparse search spaces.

The remainder of the paper is organized as follows. In Section 2, we introduce our problem setup and give a high-level overview of our framework. In Section 3, we define two search spaces over complete outputs in terms of a recurrent classifier, relate their quality to the accuracy of the classifier, and then, define sparse search spaces to improve the efficiency of our approach. We describe our cost function learning approach in Section 4. Section 5 presents our experimental results and finally Sections 6 and 7 discuss related work and future directions.

2. Problem Setup

A structured prediction problem specifies a space of structured inputs $\mathcal{X}$, a space of structured outputs $\mathcal{Y}$, and a non-negative *loss function* $L : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \mapsto \Re^+$ such that $L(x, y', y)$ is the loss associated with labeling a particular input x by output y' when the true output is y. We are provided with a training set of input-output pairs $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, drawn from an unknown target distribution, where y_i is the true output for input x_i. The goal is to return a function/predictor from structured inputs to outputs whose predicted outputs have low expected loss with respect to the distribution. Since our algorithms will be learning cost functions over input-output pairs we assume the availability of a *feature function* $\Phi : \mathcal{X} \times \mathcal{Y} \mapsto \Re^n$ that computes an n dimensional feature vector for any pair. Intuitively these features should provide some measure of compatibility between (parts of) the structured input and output.

We consider a framework for structured prediction based on state-based search in the space of complete structured outputs. The states of the search space are pairs of inputs and outputs (x, y), representing the possibility of predicting y as the output for x. A search space over those states is specified by two functions: 1) An *initial state function* I such that $I(x)$ returns an initial search state for any input x, and 2) A *successor function* S such that for any search state (x, y), $S((x, y))$ returns a set of successor states $\{(x, y_1), \ldots, (x, y_k)\}$, noting that each successor must involve the same input x as the parent. Section 3 will describe our approach for automatically defining and learning search spaces.

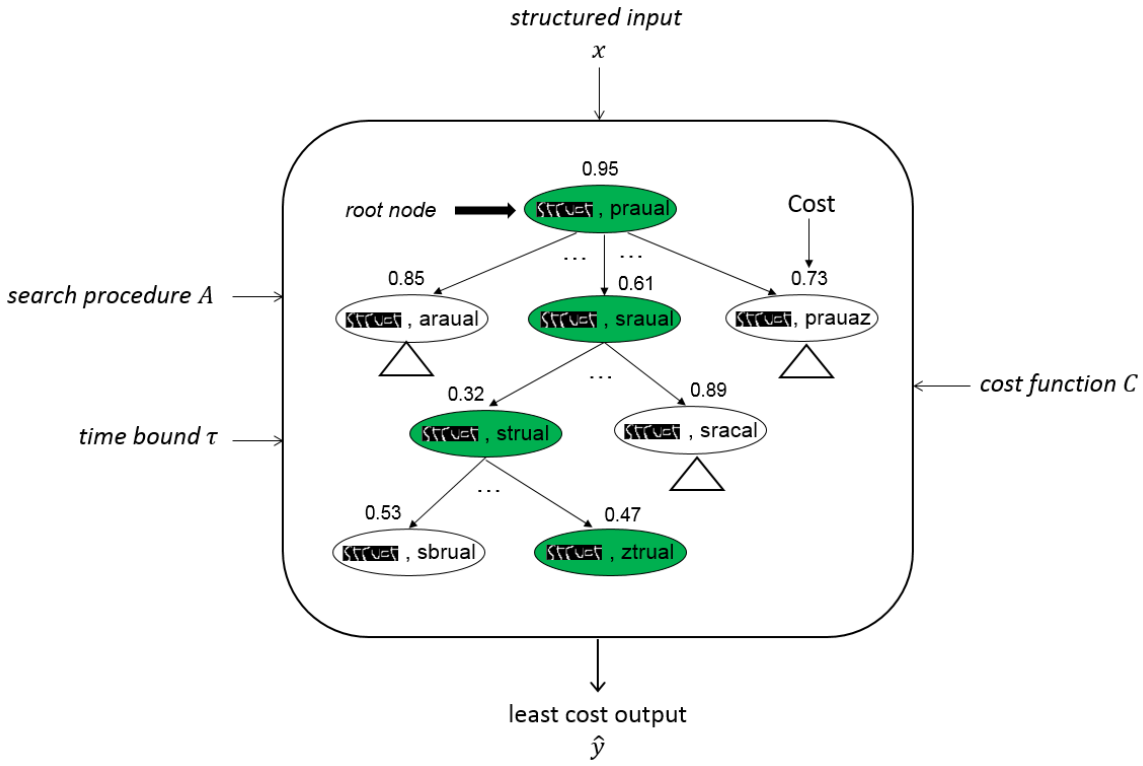


Figure 1: A high level overview of our output space search framework. Given a structured input x, we first instantiate a search space over complete outputs. Each search node in this space consists of a complete input-output pair. Next, we run a search procedure $\mathcal{A}$ (e.g., greedy search) guided by the cost function $\mathcal{C}$ for a time bound τ. The highlighted nodes correspond to the search trajectory traversed by the search procedure, in this case greedy search. We return the least cost output $\hat{y}$ that is uncovered during the search as the prediction for x.

In order to predict outputs, our framework requires two elements in addition to the search space. First, we require a cost function $\mathcal{C}$ that returns a numeric cost for any input-output pair (i.e., search state). Second, we require a search procedure $\mathcal{A}$ (e.g., greedy search or beam search) for traversing search spaces, possibly guided by the cost function. Given these elements, an input x, and a prediction time bound τ we compute an output by executing the search procedure $\mathcal{A}$ starting in the initial state $I(x)$ and guided by the cost function until the time bound is exceeded. We then return the output $\hat{y}$ corresponding to the least cost state that was uncovered during the search as the prediction for x. Figure 1 gives a high-level overview of our search-based framework for structured prediction.

The effectiveness of our search-based framework depends critically on the quality of the search space and the cost function. Ideally we would like the search space to be organized such that high quality outputs are as close as possible to the initial state, which allows the

search procedure to uncover those outputs more easily. In addition, it is critical that the cost function is able to correctly score the generated outputs according to their true losses and also to provide effective guidance to the chosen search procedure. A key contribution of this work is to propose supervised learning mechanisms for producing both high-quality search spaces and cost functions, which are described in the next two sections respectively.

3. Search Spaces Over Complete Outputs

In this section we describe two search spaces over structured outputs: 1) The Flipbit space, a simple but sometimes effective baseline, and 2) The limited-discrepancy search (LDS) space, which is intended to improve on the baseline. The key trainable element of each search space is a recurrent classifier, which once trained will fully define each space. Thus, we start this section by describing recurrent classifiers and how they are learned. Next we describe how the learned recurrent classifier is used to define each of the search spaces by defining the initial state and the successor function.

3.1 Recurrent Classifiers

A recurrent classifier h constructs structured outputs based on a series of discrete decisions. This is formalized for a given structured-prediction problem by defining an appropriate *primitive search space* over the possible sequences of decisions. It is important to keep in mind the distinctions between primitive search spaces, which are used by recurrent classifiers, and the search spaces over complete outputs (e.g., flipbit and LDS) upon which our overall framework is built. A primitive search space is a 5-tuple $\langle I, A, s, f, T \rangle$, where I is a function that maps an input x to an initial search node, A is a finite set of actions (or operators), s is the successor function that maps any search node and action to a successor search node, f is a feature function from search nodes to real-valued feature vectors, and T is the terminal state predicate that maps search nodes to $\{1, 0\}$ indicating whether the node is a terminal or not. Each terminal node in the search space corresponds to a *complete structured output*, while non-terminal nodes correspond to *partial structured outputs*. Thus, the decision process for constructing an output corresponds to selecting a sequence of actions leading from the initial node to a terminal. A recurrent classifier is a function that maps nodes of the primitive search space to actions, where typically the mapping is defined in terms of a feature function $f(n)$ that returns a feature vector for any search node. Thus, given a recurrent classifier, we can produce an output for x by starting at the initial node of the primitive space and following its decisions until reaching a terminal state.

As an example, for sequence labeling problems, the initial state for a given input sequence x is a node containing x with no labeled elements. The actions correspond to the selection of individual labels, and the successor function adds the selected label in the next position. Terminal nodes correspond to fully labeled sequences and the feature function computes a feature vector based on the input and previously assigned labels. Figure 2 provides an illustration of the primitive search space for a simple handwriting recognition problem. Each search state is a pair (x, y') where x is the structured input (binary image of the handwritten word) and y' is a partial labeling of the word. The arcs in this space correspond to search steps that label the characters in the input image in a left-to-right order by extending y' in all possible ways by one element. The terminal states or leaves of

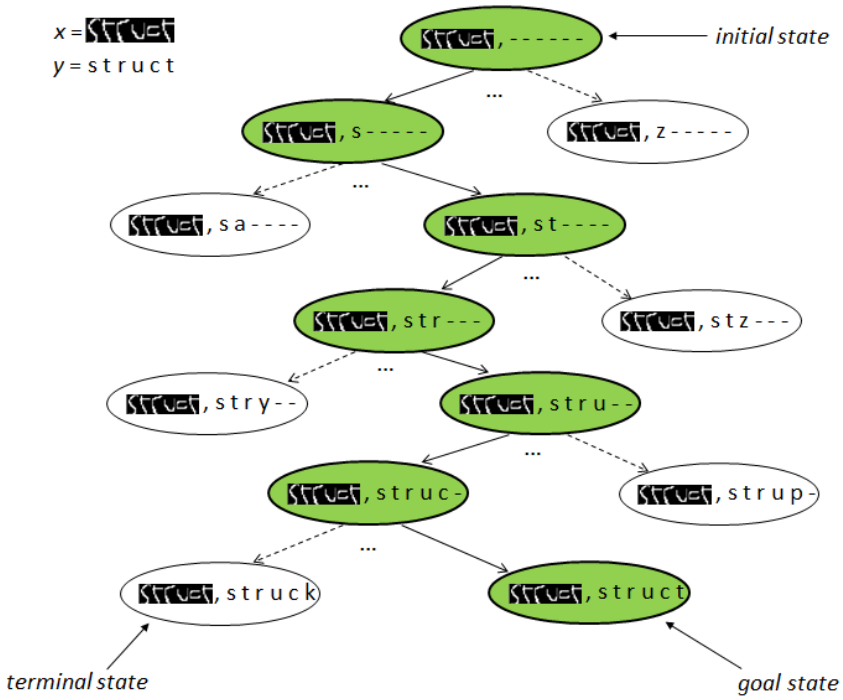


Figure 2: An example primitive search space for the handwriting recognition problem. Arcs represent labeling actions. Solid arcs correspond to the labeling actions taken by the recurrent classifier (optimal classifier in this case).

this space correspond to complete labelings of input x. The terminal state corresponding to the correct output y is labeled as *goal state*. Highlighted nodes correspond to the trajectory of the optimal recurrent classifier (i.e., a classifier that chooses correct action at every state leading to the goal state).

The most basic approach to learning a recurrent classifier is via *exact imitation* of the trajectory followed by the optimal classifier. For this, we assume that for any training input-output pair (x, y) we can efficiently find an action sequence, or *solution path*, for producing y from x. For example, the sequence of highlighted states in Figure 2 correspond to such a solution path. The exact imitation approach learns a classifier by creating a classification training example for each node n on the solution path of a structured example with feature vector $f(n)$ and label equal to the action followed by the path at n. Our experiments will use recurrent classifiers trained via exact imitation (see Algorithm 1), but more sophisticated methods such as Searn (Hal Daumé III et al., 2009) or DAgger (Ross et al., 2011) could also be used.

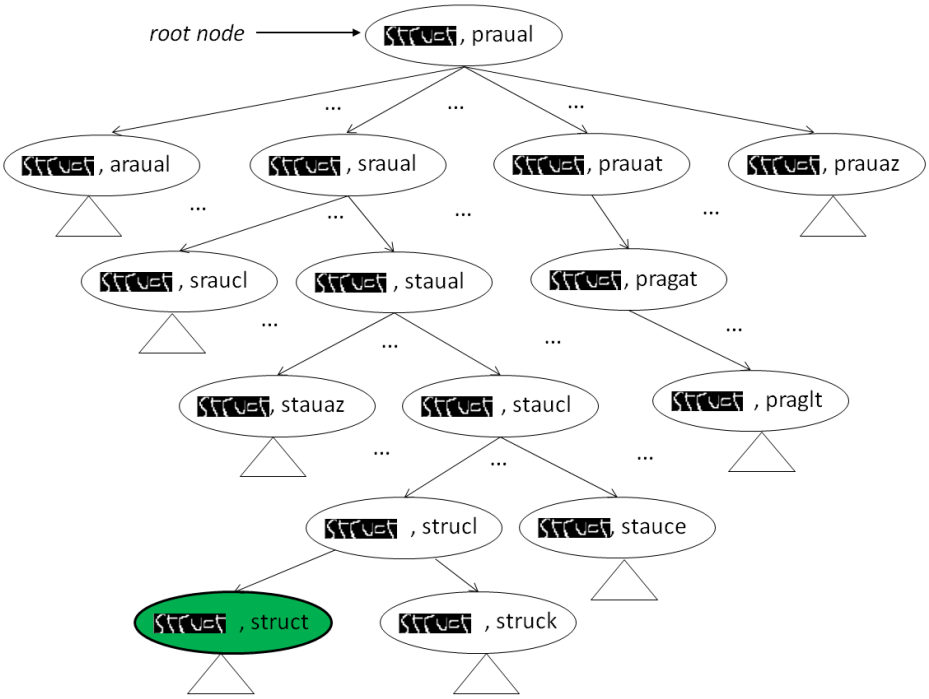


Figure 3: An example Flipbit search space for the handwriting recognition problem

Algorithm 1 Recurrent Classifier Learning via Exact Imitation

Input: $\mathcal{D}$ = Training examples
Output: h, the recurrent classifier

Initialize the set of classification examples $\mathcal{L} = \emptyset$
for each training example $(x, y = y_1 y_2 \cdots y_T) \in \mathcal{D}$ **do**
  for each search step $t = 1$ to T **do**
    Compute features f_n for search node $n = (x, y_1 \cdots y_{t-1})$
    Add classification example (f_n, y_t) to $\mathcal{L}$
  end for
end for
$h =$**Classifier-Learner**$(\mathcal{L})$ *// learn classifier from all the classification examples*
return learned classifier h

3.2 Flipbit Search Space

The *Flipbit search space* is a simple baseline space over complete outputs that uses a given recurrent classifier h for bootstrapping the search. Each search state is represented by a sequence of actions in the primitive space ending in a terminal node representing a complete output. The initial search state corresponds to the actions selected by the classifier, so that $I(x)$ is equal to $(x, h(x))$, where $h(x)$ is the complete output generated by the recurrent

classifier. The search steps generated by the successor function can change the value of one action at any sequence position of the parent state. In a sequence labeling problem, this corresponds to initializing to the recurrent classifier output and then searching over flips of individual labels. The flipbit space is often used by local search techniques (without the classifier initialization) and is similar to the search space underlying Gibbs Sampling.

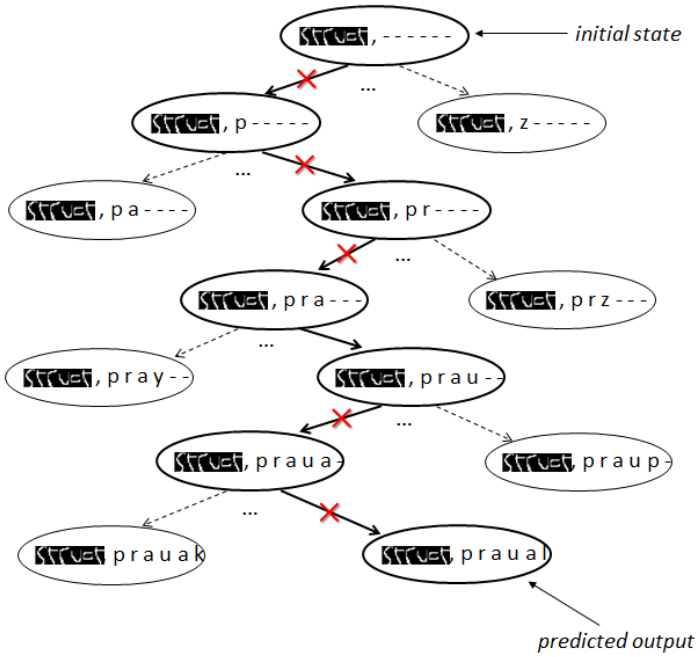
Figure 3 provides an illustration of the flipbit search space via the same handwriting recognition example that was used earlier. Each search state consists of a complete input-output pair and the complete output at every state differs from that of its parent by exactly one label. The highlighted state corresponds to the one with true output y at the smallest depth, which is equal to the number of errors in the output produced by the recurrent classifier.

3.3 Limited-Discrepancy Search Space (LDS)

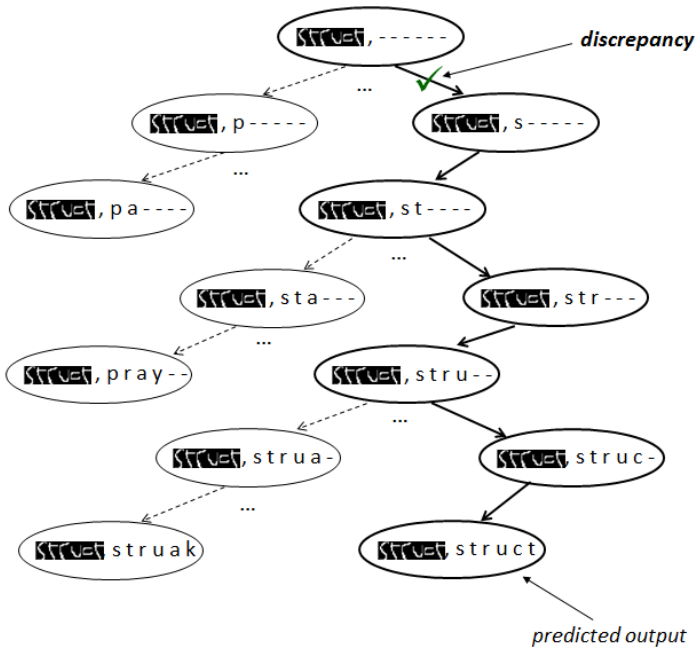
Notice that the Flipbit space only uses the recurrent classifier when initializing the search. The motivation behind the Limited Discrepancy Search (LDS) space is to more aggressively exploit the recurrent classifier in order to improve the search space quality. LDS was originally introduced in the context of problem solving using heuristic search (Harvey and Ginsberg, 1995). To put LDS in context, we will describe it in terms of using a classifier for structured prediction given a primitive search space. If the learned classifier is accurate, then the number of incorrect action selections will be relatively small. However, even a small number of errors can propagate and cause poor outputs. The key idea behind LDS is to realize that if the classifier response was corrected at the small number of critical errors, then a much better output would be produced. LDS conducts a (shallow) search in the space of possible corrections in the hope of finding a solution better than the original.

More formally, given a recurrent classifier h and its selected action sequence of length T, a discrepancy is a pair (i, a) where $i \in \{1, \ldots, T\}$ is the index of a decision step and $a \in A$ is an action, which generally is different from the choice of the classifier at step i. For any set of discrepancies D we let $h[D]$ be a new classifier that selects actions identically to h, except that it returns action a at decision step i if $(i, a) \in D$. Thus, the discrepancies in D can be viewed as overriding the preferred choice of h at particular decision steps, possibly correcting errors, or introducing new errors. For a structured input x, we will let $h[D](x)$ denote the output returned by $h[D]$ for the search space conditioned on x. At one extreme, when D is empty, $h[D](x)$ simply corresponds to the output produced by the recurrent classifier. At the other extreme, when D specifies an action at each step, $h[D](x)$ is not influenced by h at all and is completely specified by the discrepancy set. In practice, when h is reasonably accurate, we will be primarily interested in small discrepancy sets relative to the size of the decision sequence. In particular, if the error rate of the classifier on individual decisions is small, then the number of corrections needed to produce a correct output will be correspondingly small. The problem is that we do not know where the corrections should be made, and thus LDS conducts a search over the discrepancy sets, usually from small to large sets.

Consider the handwriting recognition example in Figure 4. The actual output produced by the classifier for the input image is `praual`, that is, output produced by introducing zero discrepancies (see Figure 4(a)). If we introduce one discrepancy at the first position



(a)



(b)

Figure 4: Illustration of Limited Discrepancy Search: (a) Trajectory of the recurrent classifier with no discrepancies. Arcs with 'X' mark indicate incorrect actions chosen by the classifier. (b) Trajectory of the recurrent classifier with a correction (discrepancy) at the first error. A single correction allows the classifier to correct all the remaining errors.

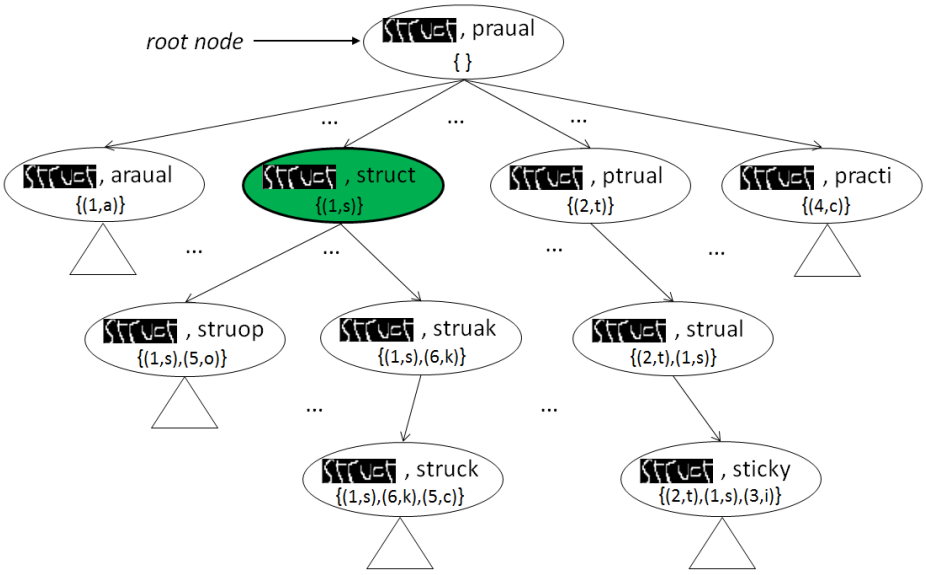


Figure 5: An example Limited Discrepancy Search (LDS) space for the handwriting recognition problem

$(1, s)$ and run the classifier for the remaining labeling, we get `struct` which corrects all the remaining mistakes (see Figure 4(b)). By introducing this single correction, the classifier automatically corrected a number of other previous mistakes, which had been introduced due to propagation of the first error. Thus only one discrepancy was required to produce the correct output even though the original output contained many more errors.

Given a recurrent classifier h, we define the corresponding limited-discrepancy search space over complete outputs S_h as follows. Each search state in the space is represented as (x, D) where x is a structured input and D is a discrepancy set. We view a state (x, D) as equivalent to the input-output state $(x, h[D](x))$. The initial state function I simply returns $(x, \emptyset)$ which corresponds to the original output of the recurrent classifier. The successor function S for a state (x, D) returns the set of states of the form (x, D'), where D' is the same as D, but with an additional discrepancy. In this way, a path through the LDS search space starts at the output generated by the recurrent classifier and traverses a sequence of outputs that differ from the original by some number of discrepancies. Given a reasonably accurate h, we expect that high-quality outputs will be generated at relatively shallow depths of this search space and hence will be generated quickly.

Figure 5 illustrates[1] the limited-discrepancy search space. Each state consists of the input x, a discrepancy set D and the output produced by running the classifier with the specified discrepancy set, that is, $h[D](x)$. The root node has an empty discrepancy set. Nodes at level one contain discrepancy sets of size one and nodes at level two contain

1. It may not be clear from this example, but we allow over-riding the discrepancies to provide the opportunity to recover from the search errors.

discrepancy sets of size two, and so on. The highlighted state corresponds to the smallest depth state containing the true output.

3.4 Search Space Quality

Recall that in our experiments we train recurrent classifiers via exact imitation, which is an extremely simple approach compared to more elaborate methods such as Searn. We now show the desirable property that the "exact imitation accuracy" optimized by that approach is directly related to the "quality" of the LDS search space, where quality relates the expected amount of search needed to uncover the target output. More formally, given an input-output pair (x, y) we define the *LDS target depth* for an example (x, y) and recurrent classifier h to be the minimum depth of a state in the LDS space corresponding to y. Given a distribution over input-output pairs we let $d(h)$ denote the expected LDS target depth of a classifier h. Intuitively, the depth of a state in a search space is highly related to the amount of search time required to uncover the node (exponentially related for exhaustive search, and at least linearly related for more greedy search). Thus, we will use $d(h)$ as a measure of the quality of the LDS space. We now relate $d(h)$ to the classifier error rate.

For simplicity, assume that all decision sequences for the structured-prediction problem have a fixed length T and consider an input-output pair (x, y), which has a corresponding sequence of actions that generates y. Given a classifier h, we define its *exact imitation error* on (x, y) to be e/T where e is the number of mistakes h makes at nodes along the action sequence of (x, y) (i.e., how often does it disagree with the optimal classifier along the path of the optimal classifier). Further, given a distribution over input-output pairs, we let $\epsilon_{ei}(h)$ denote the expected exact imitation error with respect to examples drawn from the distribution. Note that the exact imitation training approach aims to learn a classifier that minimizes $\epsilon_{ei}(h)$. Also, let $\epsilon_r(h)$ denote the *expected recurrent error* of h, which is the expectation over randomly drawn (x, y) of the Hamming distance between the action sequence produced by h when applied to x and the true action sequence for (x, y). The error $\epsilon_r(h)$ is the actual measure of performance of h when applied to structured prediction. Recall that due to error propagation it is possible that $\epsilon_r(h)$ can be much worse than $\epsilon_{ei}(h)$, by as much as a factor of T (e.g., see Ross et al., 2011). The following proposition shows that $d(h)$ is related to $\epsilon_{ei}(h)$ rather than the potentially much larger $\epsilon_r(h)$.

Proposition 1 *For any classifier h and distribution over structured input-outputs, $d(h) = T\epsilon_{ei}(h)$.*

Proof For any example (x, y) the depth of y in S_h is equal to the number of imitation errors made by h on (x, y). To see this, simply create a discrepancy set D that contains a discrepancy at the position of each imitation error that corrects the error. This set is at a depth equal to the number of imitation errors and the classifier $h[D]$ will exactly produce the action sequence that corresponds to y. The result follows by noting that the expected number of imitation errors is equal to $T\epsilon_{ei}(h)$. ■

It is illustrative to compare this result with the Flipbit space. Let $d'(h)$ be the expected target depth in the Flipbit space of a randomly drawn (x, y). It is easy to see that $d'(h) = T\epsilon_r(h)$ since each search step can only correct a single error and the expected number of

errors of the action sequence at the initial node is $T\epsilon_r(h)$. Since in practice and in theory $\epsilon_r(h)$ can be substantially larger than $\epsilon_{ei}(h)$ (by as much as a factor of T), this shows that the LDS space will often be superior to the baseline Flipbit space in terms of the expected target depth. For example, the target depth of the example LDS space in Figure 5 is one and is much smaller than the target depth of the example flipbit space in Figure 3, which is equal to five. Since this depth relates to the difficulty of search and cost-function learning, we can expect the LDS space to be advantageous when $\epsilon_r(h)$ is larger than $\epsilon_{ei}(h)$. In our experiments, we will see that this is indeed the case.

3.5 Sparse Search Spaces

In this section, we first discuss some of the scalability issues that arise in our framework due to the use of LDS and Flipbit spaces as defined above. Next, we describe how to define sparse versions of these search spaces to improve the efficiency of our approach.

In our search-based framework, the most computationally demanding part is the generation of candidate states during the search process. Given a parent state, the number of successor states for both the LDS and flipbit spaces is equal to $T \cdot (L-1)$ where T is the size of the structured output and L is the number of primitive action choices (the number of labels for sequence labeling problems). When T and/or L is large the time required for this expansion and computing the feature vector for each successor can be non-trivial. While we cannot control T, since that is dictated by the size of the input, we can consider reducing the effective size of L via pruning. Intuitively, for many sequence positions of a structured output, there will often be labels that can be easily determined to be bad by looking at the confidence of the recurrent classifier. By explicitly pruning those labels from consideration we can arrive at a sparse successor set that requires significantly less computation to generate.

More formally, our approach for defining a sparse successor function assumes that the recurrent classifier used to define the LDS and flipbit spaces produces confidence scores, rather than just simple classifications. This allows us to provide a ranking of the potential actions, or labels, at each position of the structured output. Using this ranking information it is straightforward to define sparse versions of the LDS and flipbit successor function by only considering successors corresponding to the top k labels at each sequence position. For the flipbit space this means that successors correspond to any way of changing the choice of the recurrent classifier at a sequence position to one of the top k labels at that position. For the LDS space, this means that we only consider introducing discrepancies at position i involving the top k labels at position i. In this way, the number of successors of a state in either space will be $T \cdot k$ rather than $T \cdot (L-1)$. Therefore, these sparse search spaces will lead to significant speed improvements for problems with large L (e.g., POS tagging, Handwriting recognition, and Phoneme prediction).

The potential disadvantage of using a small value of k is that accuracy could be hurt if good solution paths are pruned away due to inaccuracy of the classifier's confidence estimates. Thus, k provides a way to trade-off prediction speed versus accuracy. As we will show later in the experiments, we are generally able to find values of k that lead to significant speedups with little loss in accuracy.

4. Cost Function Learning

In this section, we describe a generic framework for cost function learning that is applicable for a wide range of search spaces and search strategies. This approach is motivated by our empirical observation that for a variety of structured prediction problems, we can uncover high quality outputs if we guide the output-space search using the true loss function as an oracle cost function to guide the search (close to zero error with both LDS and Flipbit spaces). Since the loss function depends on correct target output y^*, which is unknown at test time, we aim to learn a cost function that mimics this oracle search behavior on the training data without requiring knowledge of y^*. With an appropriate choice of hypothesis space of cost functions, good performance on the training data translates to good performance on the test data.

4.1 Cost Function Learning via Imitation Learning

Recall that in our output space search framework, the role of the cost function $\mathcal{C}$ is to evaluate the complete outputs that are uncovered by the search procedure. These evaluations may be used internally by the search procedure as a type of heuristic guidance and also used when the time bound is reached to return the least cost output that has been uncovered as the prediction. Based on our empirical observations, it is very often the case that the true loss function serves these roles very effectively, which might suggest a goal of learning a cost function that is approximately equal to the true loss function L over all possible outputs. However, this objective will often be impractical and fortunately is unnecessary. In particular, the learned cost function need not approximate the true loss function uniformly over the output space, but only needs to make the decisions that are sufficient for leading the time-bounded search to behave as if it were using the true loss function. Often this allows for a much less constrained learning problem, for example, $\mathcal{C}$ may only need to preserve the rankings among certain outputs, rather than exactly matching the values of L. The key idea behind our cost learning approach is to learn such a sufficient $\mathcal{C}$. The main assumptions made by this approach are: 1) the true loss function can provide effective guidance to the search procedure by making a series of ranking decisions, and 2) we can learn to imitate those ranking decisions sufficiently well.

Our goal now is to learn a cost function that causes the search to behave as if it were using loss function L for guiding the search and selecting the final output. We propose to formulate and solve this problem in the framework of imitation learning. In traditional imitation learning, the goal of the learner is to learn to imitate the behavior of an expert performing a sequential-decision making task in a way that generalizes to similar tasks or situations. Typically this is done by collecting a set of trajectories of the expert's behavior on a set of training tasks. Then supervised learning is used to find a policy that can replicate the decisions made on those trajectories. Often the supervised learning problem corresponds to learning a classifier or policy from states to actions and off-the-shelf tools can be used.

In our cost function learning problem, the expert corresponds to the search procedure $\mathcal{A}$ using the loss function L for a search time bound τ_{max}. The behavior that we would like to imitate is the internal behavior of this search procedure, which consists of all decisions made during the search including the final decision of which output to return. Thus, the

goal of cost function learning is to learn the weights of $\mathcal{C}$ so that this behavior is replicated when it is used by the search procedure in place of L. We propose to achieve this goal by directly monitoring the expert search process on all of the structured training examples and generating the set of constraints on L that were responsible for the observed decisions. Then we attempt to learn a $\mathcal{C}$ that satisfies the constraints using an optimization procedure.

Algorithm 2 Cost Function Learning via Exact Imitation

Input: $\mathcal{D}$ = Training examples, (I, S) = Search space definition, L = Loss function, $\mathcal{A}$ = Rank-based search procedure, τ_{max} = search time bound
Output: $\mathcal{C}$, the cost function

1: Initialize the set of ranking examples $\mathcal{R} = \emptyset$
2: **for** each training example $(x, y^*) \in \mathcal{D}$ **do**
3: $s_0 = I(x)$ *// initial state of the search tree*
4: $M_0 = \{s_0\}$ *// set of open nodes in the internal memory of the search procedure*
5: y_{best} =**OutputOf**(s_0) *// best loss output so far*
6: **for** each search step $t = 1$ to τ_{max} **do**
7: Select the state(s) to expand: N_t =**Select**$(\mathcal{A}, L, M_{t-1})$
8: Expand every state $s \in N_t$ using the successor function S: C_t =**Expand**(N_t, S)
9: Prune states and update the internal memory state of the search procedure: M_t =**Prune**$(\mathcal{A}, L, M_{t-1} \cup C_t \setminus N_t)$
10: Update the best loss output y_{best} *// track the best output*
11: Generate ranking examples R_t to imitate this search step
12: Add ranking examples R_t to $\mathcal{R}$: $\mathcal{R} = \mathcal{R} \cup R_t$ *// aggregation of training data*
13: **end for**
14: **end for**
15: $\mathcal{C}$ =**Rank-Learner**$(\mathcal{R})$ *// learn cost function from all the ranking examples*
16: **return** learned cost function $\mathcal{C}$

Algorithm 2 describes our generic approach for cost function learning via exact imitation of searches conducted by the loss function. It is applicable to a wide-range of search spaces, search procedures and loss functions. The learning algorithm takes as input: 1) $\mathcal{D} = \{(x, y^*)\}$, set of training examples for a structured prediction problem (e.g., handwriting recognition); 2) $\mathcal{S}_o = (I, S)$, definition of a search space over complete outputs (e.g., LDS space), where I is the initial state function and S is the successor function; 3) L, a task loss function defined over complete outputs (e.g., hamming loss); 4) $\mathcal{A}$, a rank-based search procedure (e.g., greedy search); 5) τ_{max}, the search time bound (e.g., number of search steps).

First, it runs the search procedure $\mathcal{A}$ with the given loss function L, in the search space $\mathcal{S}_o$ instantiated for every training example (x, y^*), upto the maximum time bound τ_{max} (steps 3-10), and generates a set of pair-wise ranking examples that need to be satisfied to be able to imitate the search behavior with loss function (step 11). Second, the aggregate set of ranking examples collected over all the training examples is then given to a rank-learning algorithm (e.g., Perceptron or SVM-Rank) to learn the weights of the cost function (step 15).

The algorithmic description assumes a best-first search procedure with some level of pruning (e.g., greedy search and best-first beam search). These search procedures typically involve three key steps: 1) Selection, 2) Expansion and 3) Pruning. During selection, the search procedure selects one or more[2] open nodes from its internal memory for expansion (step 7), and expands all the selected nodes to generate the candidate set (step 8). It retains only a subset of all the open nodes after expansion in its internal memory and prunes away all the remaining ones (step 9). For example, greedy search maintains only the best node and best-first beam search with beam width b retains the best b nodes.

The most important step in our cost function learning algorithm is the generation of ranking examples to imitate the search procedure (step 11). In what follows, we first formalize ranking-based search that allows us to specify what these pairwise ranking examples are and then, give a generic description of "sufficient" pair-wise decisions to imitate the search, and illustrate them for greedy search and best-first beam search through a simple example.

4.2 Ranking-based Search

We now precisely define the notion of "guiding the search" with a loss function. If the loss function can be invoked arbitrarily by the search procedure, for example, evaluating and comparing arbitrary expressions involving the cost, then matching its performance would require the cost function to approximate it arbitrarily closely, which is quite demanding in most cases. Hence, we restrict ourselves to ranking-based search defined as follows.

Let $\mathcal{P}$ be an anytime search procedure that takes an input $x \in \mathcal{X}$, calls a cost function $\mathcal{C}$ over the pairs from $\mathcal{X} \times \mathcal{Y}$ some number of times and outputs a structured output $y_{best} \in \mathcal{Y}$. We say that $\mathcal{P}$ is a ranking-based search procedure if the results of calls to $\mathcal{C}$ are only used to compare the relative values for different pairs (x, y) and (x, y') with a fixed tie breaker. Each such comparison with tie-breaking is called a ranking decision and is characterized by the tuple (x, y, y', d), where d is a binary decision that indicates y is a better output than y' for input x. When requested, it returns the best output y_{best} encountered thus far as evaluated by the cost function.

Note that the above constraints prohibit the search procedure from being sensitive to the absolute values of the cost function for particular search states (x, y) pairs, and only consider their relative values. Many typical search strategies such as greedy search, best-first search, and beam search satisfy this property.

A *run* of a ranking-based search is a sequence $x, m_1, o_1, \ldots, m_n, o_n, y$, where x is the input to the predictor, y is the output, and m_i is the internal memory state of the predictor just before the i^{th} call to the ranking function. o_i is the i^{th} ranking decision (x, y_i, y'_i, d_i). Given a hypothesis space $\mathcal{H}$ of cost functions, the cost function learning works as follows. It runs the search procedure $\mathcal{P}$ on each training example (x, y^*) for a maximum search time bound of τ_{max} substituting the loss function $L(x, y, y^*)$ for the cost function $\mathcal{C}(x, y)$. For each run, it records the set of all ranking decisions (x, y_i, y'_i, d_i). The set of all ranking decisions from all the runs is given as input to a binary classifier, which finds a cost function $C \in \mathcal{H}$, consistent with the set of all such ranking decisions.

2. Breadth-first beam search expands all nodes in the beam.

The ranking-based search can be viewed as a Markov Decision Process (MDP), where the internal states of the search procedure correspond to the states of the MDP, and the ranking decision is an action. The following theorem can be proved by adapting the proof of Fern et al. (2006) with minor changes, for example, no discounting, and two actions, and applies to stochastic as well as deterministic search procedures.

Theorem 2 *Let $\mathcal{H}$ be a finite class of ranking functions. For any target ranking function $r \in \mathcal{H}$, and any set of $m = \frac{1}{\epsilon} \ln \frac{|\mathcal{H}|}{\delta}$ independent runs of a rank-based search procedure $\mathcal{P}$ guided by r drawn from a target distribution over inputs, there is a $1 - \delta$ probability that every $\hat{r} \in \mathcal{H}$ that is consistent with the runs satisfies $L(\hat{r}) \leq L(r) + 2\epsilon L_{max}$, where L_{max} is the maximum possible loss of any output and $L(t)$ is the expected loss of running the search procedure $\mathcal{P}$ with the ranking function $t \in \mathcal{H}$.*

Although the theoretical result assumes that the target cost function h is in the hypothesis space, in practice this is not guaranteed as the set of generated constraints might be quite large and diverse. To help reduce the complexity of the learning problem, in practice we only learn from a smaller set of pair-wise ranking decisions that are sufficient (see below) to preserve the best output that is encountered during search at any time step.

4.3 Sufficient Pairwise Decisions

Above we noted that we only need to collect and learn to imitate the "sufficient" pairwise decisions encountered during search. We say that a set of constraints is sufficient for a structured training example (x, y^*), if any cost function that is consistent with the constraints causes the search to follow the same trajectory (sequence of states) and retains the best loss output that is encountered during search so far for input x. The precise specification of these constraints depends on the actual search procedure that is being used. For rank-based search procedures, the sufficient constraints can be categorized into three types:

1. *Selection* constraints, which ensure that the search node(s) from the internal memory state that will be expanded in the next search step is (are) ranked better than all other nodes.

2. *Pruning* constraints, which ensure that the internal memory state (set of search nodes) of the search procedure is preserved at every search step. More specifically, these constraints involve ranking every search node in the internal memory state better (lower $\mathcal{C}$-value) than those that are pruned.

3. *Anytime* constraints, which ensure that the search node corresponding to the best loss output that is encountered during search so far is ranked better than every other node that is uncovered by the search procedure.

Below, we will illustrate these constraints concretely for greedy search and best-first beam search noting that similar formulations for other rank-based search procedures is straightforward.

Greedy Search: This is the most basic rank-based search procedure. For a given input x, it traverses the search space by selecting the next state as the successor of the current

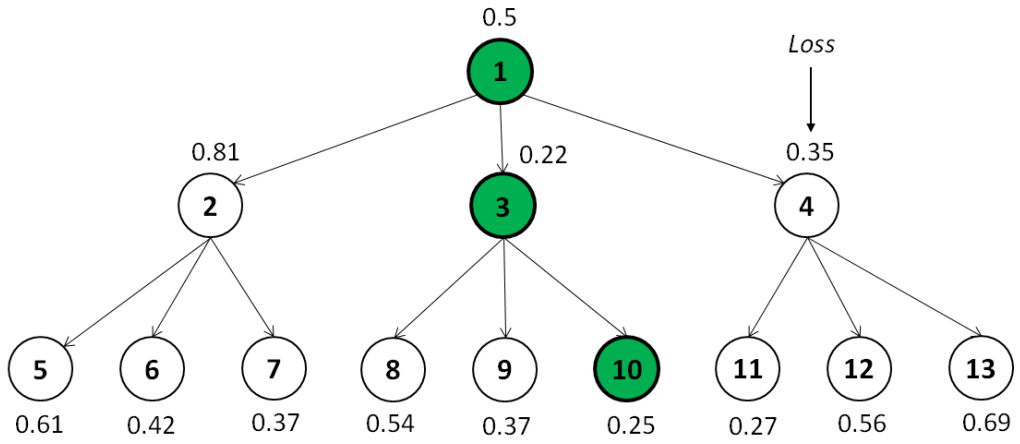


Figure 6: An example search tree that illustrates greedy search with loss function. Each node represents a complete input-output pair and can be evaluated using the loss function. The highlighted nodes correspond to the trajectory of greedy search guided by the loss function.

state that looks best according to the cost function $\mathcal{C}$ (loss function L during training). In particular, if s_i is the search state at step i, greedy search selects $s_{i+1} = \arg min_{s \in S(s_i)} \mathcal{C}(s)$, where $s_0 = I(x)$. In greedy search, the internal memory state of the search procedure at step i consists of only the best open (unexpanded) node s_i. Additionally, it keeps track of the best node s_i^{best} uncovered so far as evaluated by the cost function.

Let (x, y_i) correspond to the input-output pair associated[3] with state s_i. Since greedy search maintains only a single open node s_i in its internal memory at every search step i, there are no selection constraints. Let C_{i+1} be the candidate set after expanding state s_i, that is, $C_{i+1} = S(s_i)$. Let s_{i+1} be the best node in the candidate set C_{i+1} as evaluated by the loss function, that is, $s_{i+1} = \arg min_{s \in C_{i+1}} L(s)$. As greedy search prunes all the nodes in the candidate set other than s_{i+1}, pruning constraints need to ensure that s_{i+1} is ranked better than all the other nodes in C_{i+1}. Therefore, we include one ranking constraint for every node $(x, y) \in C_{i+1} \setminus (x, y_{i+1})$ such that $\mathcal{C}(x, y_{i+1}) < \mathcal{C}(x, y)$. As part of the anytime constraints, we introduce a constraint between (x, y_i^{best}) and (x, y_{i+1}) according to their losses. For example, if $L(x, y_{i+1}, y^*) < L(x, y_i^{best}, y^*)$, we introduce a ranking constraint such that $\mathcal{C}(x, y_{i+1}) < \mathcal{C}(x, y_i^{best})$ and vice versa.

We will now illustrate these ranking constraints through an example. Figure 6 shows an example search tree of depth two with associated losses for every search node. The highlighted nodes correspond to the trajectory of greedy search with loss function that our learner has to imitate. At first search step, $\{\mathcal{C}(3) < \mathcal{C}(2), \mathcal{C}(3) < \mathcal{C}(4)\}$, and $\{\mathcal{C}(3) < \mathcal{C}(1)\}$ are the pruning and anytime constraints respectively. Similarly, $\{\mathcal{C}(10) < \mathcal{C}(8), \mathcal{C}(10) < \mathcal{C}(9)\}$, and $\{\mathcal{C}(3) < \mathcal{C}(10)\}$ form the pruning and anytime constraints at second search step. There-

3. We use input-output pair (x, y_i) and state s_i inter-changeably for the sake of brevity.

fore, the aggregate set of constraints needed to imitate the greedy search behavior shown in Figure 6 are:
$\{\mathcal{C}(3) < \mathcal{C}(2), \mathcal{C}(3) < \mathcal{C}(4), \mathcal{C}(3) < \mathcal{C}(1), \mathcal{C}(10) < \mathcal{C}(8), \mathcal{C}(10) < \mathcal{C}(9), \mathcal{C}(3) < \mathcal{C}(10)\}$.

Best-first Beam Search: This is a more sophisticated search procedure compared to greedy search. Best-first beam search maintains a set of b open nodes B_i in its internal memory at every search step i, where b is the beam width. Greedy search is a special case of beam search, where beam width b equals 1. For a given input x, it traverses the search space by expanding the best node s_i in the current beam B_i (i.e., $s_i = \arg min_{s \in B_i} \mathcal{C}(s)$) and computes the next beam B_{i+1} by retaining the best b open nodes after expanding s_i, where $B_0 = \{I(x)\}$.

Best-first beam search selects the best node s_i at every search step i from its beam B_i for expansion, that is, $s_i = \arg min_{s \in B_i} L(s)$. Therefore, selection constraints need to ensure that s_i is ranked better than all the other nodes in beam B_i. Therefore, we include one ranking constraint for every node $(x, y) \in B_i \setminus (x, y_i)$ such that $\mathcal{C}(x, y_i) < \mathcal{C}(x, y)$. Let C_{i+1} be the candidate set after expanding state s_i, that is, $C_{i+1} = S(s_i) \cup B_i \setminus s_i$ and let B_{i+1} be the best b nodes in the candidate set according to the loss function. As best-first beam search prunes all the nodes in the candidate set other than those in B_{i+1}, pruning constraints need to ensure that every node in B_{i+1} is ranked better than every node in $C_{i+1} \setminus B_{i+1}$. Therefore, we generate one ranking example for every pair of nodes $((x, y_b), (x, y)) \in B_{i+1} \times C_{i+1} \setminus B_{i+1}$, requiring that $\mathcal{C}(x, y_b) < \mathcal{C}(x, y)$. Similar to greedy search, as part of the anytime constraints, we introduce a ranking constraint between (x, y_i^{best}) and (x, y_{i+1}) according to their losses.

We will now illustrate the above ranking constraints for best-first beam search through the example search tree in Figure 6. Let us consider best-first beam search with beam width $b = 2$ and let (B_0, B_1, B_2) correspond to the beam trajectory of search with loss function that needs to be imitated by our learner, where $B_0 = \{1\}$, $B_1 = \{3, 4\}$ and $B_2 = \{4, 10\}$. At first search step, there are no selection constraints as B_0 contains only a single node, and $\{\mathcal{C}(3) < \mathcal{C}(2), \mathcal{C}(4) < \mathcal{C}(2)\}$ and $\{\mathcal{C}(3) < \mathcal{C}(1)\}$ are the pruning and anytime constraints respectively. Similarly, $\{\mathcal{C}(3) < \mathcal{C}(4)\}$, $\{\mathcal{C}(4) < \mathcal{C}(8), \mathcal{C}(4) < \mathcal{C}(9), \mathcal{C}(10) < \mathcal{C}(8), \mathcal{C}(10) < \mathcal{C}(9)\}$ and $\{\mathcal{C}(3) < \mathcal{C}(10)\}$ form the selection, pruning and anytime constraints at second search step.

The only thing that remains to be explained in Algorithm 2 is, how to learn a cost function $\mathcal{C}$ from the aggregate set of ranking examples $\mathcal{R}$ (step 15). Below we describe this rank learning procedure.

4.4 Rank Learner

We can use any off-the-shelf rank-learning algorithm (e.g., Perceptron, SVM-Rank) as our base learner to learn the cost function from the set of ranking examples $\mathcal{R}$. In our specific implementation we employed the online Passive-Aggressive (PA) algorithm (Crammer et al., 2006) as our base learner.[4] Training was conducted for 50 iterations in all of our experiments.

4. In the conference version of this paper, we employed the perceptron learner and followed a training approach that slightly differs from Algorithm 2. Specifically, the ranking examples for exact imitation were generated until reaching y^*, the correct output, and after that we only generate training examples to rank y^* higher than the best cost open node(s) as evaluated by the current cost function and continue the search guided by the cost function. This particular training approach may be beneficial (results in

PA is an online large-margin algorithm, which makes several passes over the training examples $\mathcal{R}$, and updates the weights whenever it encounters a ranking error. Recall that each ranking example in $\mathcal{R}$ is of the form $\mathcal{C}(x, y_1) < \mathcal{C}(x, y_2)$, where x is a structured input with target output y^*, y_1 and y_2 are potential outputs for x such that $L(x, y_1, y^*) < L(x, y_2, y^*)$. Let $\Delta > 0$ be the difference between the losses of the two outputs involved in a ranking example. We experimented with PA variants that use margin scaling (margin scaled by Δ) and slack scaling (errors weighted by Δ) (Tsochantaridis et al., 2005). Since margin scaling performed slightly better than slack scaling, we report the results of the PA variant that employs margin scaling. Below we give the full details of the margin scaling update.

Let w_t be the current weights of the cost function. If there is a ranking error, that is, $w_t \cdot \Phi(x, y_2) - w_t \cdot \Phi(x, y_1) < \sqrt{\Delta}$, the new weights w_{t+1} that corrects the error can be obtained using the following equation.[5]

$$w_{t+1} = w_t + \tau_t(\Phi(x, y_2) - \Phi(x, y_1))$$

where the learning rate τ_t is given by

$$\tau_t = \frac{w_t \cdot \Phi(x, y_1) - w_t \cdot \Phi(x, y_2) + \sqrt{\Delta}}{\|\Phi(x, y_2) - \Phi(x, y_1)\|^2}.$$

This specific update has been previously used for cost-sensitive multiclass classification (Crammer et al., 2006) (See Equation 51) and for structured output problems (Keshet et al., 2005) (See Equation 7).

4.5 Summary of Overall Training Approach

Our search-based framework thus consists of two main learning components: 1) the search space learner, and 2) the cost function learner. We train them sequentially. First, we train the recurrent classifier as described in Section 3.1, which is used to define either the LDS or flipbit search spaces (see Section 3). Second, we train the cost function $\mathcal{C}$ to score the outputs for a given combination of search space over complete outputs S_o and a search procedure $\mathcal{A}$ as described in Algorithm 2. More specifically, for every training example (x, y^*), we run the search procedure $\mathcal{A}$ on the search space S_o instantiated for input x, using the loss function L for the specified time bound τ, and generate imitation training data (ranking examples) for the cost function learning (see Section 4.3). We give the aggregate set of imitation training data to a rank learner to train the cost function $\mathcal{C}$ as described in Section 4.4.

---

the conference paper are slightly better than those presented in this article and in practice, breaking ties via cost function is better than employing a random tie-breaker), but it is computationally very expensive (requires the ranking examples to be generated on-the-fly during each iteration of online training). However, this training methodology can be both beneficial and practical when applied on sparse search spaces.

5. Crammer et al. (2006) prove bounds on the cumulative squared loss and therefore, they employ this particular margin constraint with $\sqrt{\Delta}$.

5. Empirical Results

In this section we empirically investigate our approach along several dimensions and compare it against the state-of-the-art in structured prediction.

5.1 Experimental Setup

We evaluate our approach on the following six structured prediction problems including five benchmark sequence labeling problems and a 2D image labeling problem.

- **Handwriting Recognition (HW).** The input is a sequence of binary-segmented handwritten letters and the output is the corresponding character sequence $[a-z]^+$. This data set contains roughly 6600 examples divided into 10 folds (Taskar et al., 2003). We consider two different variants of this task as in Hal Daumé III et al. (2009). For the `HW-Small` version of the problem,, we employ one fold for training and the remaining 9 folds for testing, and vice-versa in `HW-Large`.

- **NETtalk Stress.** This is a text-to-speech mapping problem, where the task is to assign one of the 5 stress labels to each letter of a word (Sejnowski and Rosenberg, 1987). There are 1000 training words and 1000 test words in the standard data set. We use a sliding window of size 3 for observational features.

- **NETtalk Phoneme.** This is similar to NETtalk Stress except that the task is to assign one of the 51 phoneme labels to each letter of a word.

- **Chunking.** The goal in this task is to syntactically chunk English sentences into meaningful segments. We consider the full syntactic chunking task and use the data set from the CONLL 2000 shared task,[6] which consists of 8936 sentences of training data and 2012 sentences of test data.

- **POS tagging.** We consider the tagging problem for the English language, where the goal is to assign the part-of-speech tag to each word in a sentence. The standard data from Wall Street Journal (WSJ) corpus[7] was used in our experiments.

- **Scene labeling.** This data set contains 700 images of outdoor scenes (Vogel and Schiele, 2007). Each image is divided into patches by placing a regular grid of size 10×10 over the entire image, where each patch takes one of the 9 semantic labels (*sky, water, grass, trunks, foliage, field, rocks, flowers, sand*). Simple appearance features including color, texture and position are used to represent each patch. Training was performed with 600 images, and the remaining 100 images were used for testing.

We used F_1 loss as the loss function for the chunking task and employed Hamming loss for all other tasks.

For all sequence labeling problems, the recurrent classifier labels a sequence using a left-to-right ordering and for scene labeling uses an ordering of top-left to right-bottom in a row-wise raster form. To train the recurrent classifiers, the output label of the previous

6. CONLL task can be found at `http://www.cnts.ua.ac.be/conll2000/chunking/`.
7. WSJ corpus can be found at `http://www.cis.upenn.edu/~treebank/`.

token is used as a feature to predict the label of the current token for all sequence labeling problems with the exception of chunking and POS tagging, where labels of the two previous tokens were used. For scene labeling, the labels of neighborhood (top and left) patches were used. In all our experiments, we train the recurrent classifier using exact imitation (see Section 3) with the Perceptron algorithm for 100 iterations with a learning rate of 1.

Unless otherwise indicated, the cost functions learned over input-output pairs are second order, meaning that they have features over neighboring label pairs and triples along with features of the structured input. For the scene labeling task, we consider pairs and triples along both horizontal and vertical directions. We trained the cost function via exact imitation as described in Section 4 using 50 iterations of Passive-Aggressive training.

5.2 Comparison to State-of-the-Art

We experimented with several instantiations of our framework. First, we consider our framework using a greedy search procedure for both the LDS and flipbit spaces, denoted by **LDS-Greedy** and **FB-Greedy**. Unless otherwise noted, in both training and testing, the greedy search was run for a number of steps equal to the length of the sequence. Using longer runs did not impact results significantly. Second, we performed experiments with best-first beam search for different beam widths and search steps, but we didn't see significant improvements over the results with greedy search. Therefore, we do not report these results. Third, to see the impact of adding additional search at test time to a greedily trained cost function, we also used the cost function learned by LDS-Greedy and FB-Greedy in the context of a best-first beam search (beam width = 100) at test time in both the LDS and flipbit spaces, denoted by **LDS-BST(greedy)** and **FB-BST(greedy)**. We also report the performance of using our trained recurrent classifier (**Recurrent**) to make predictions, which is equivalent to performing no search since both search spaces are initialized to the recurrent classifier output. Finally, we also report the exact imitation accuracy ($100 * (1 - \epsilon_{ei})$), which as described earlier (see Section 3) is the accuracy being directly optimized by the recurrent classifier and is related to the structure of the flipbit and LDS spaces.

We compare our results with other structured prediction algorithms including **CRFs** (Lafferty et al., 2001), **SVM-Struct** (Tsochantaridis et al., 2004), **SEARN** (Hal Daumé III et al., 2009) and **CASCADES** (Weiss and Taskar, 2010). For these algorithms, we report the best published results whenever available. In the remaining cases, we used publicly available code or our own implementation to generate those results. Ten percent of the training data was used to tune the hyper-parameters. CRFs were trained using SGD.[8] SVM^{hmm} was used to train SVM-Struct and the value of the parameter C was chosen from $\{10^{-4}, 10^{-3}, \cdots, 10^{3}, 10^{4}\}$ based on the validation set. Cascades were trained using the implementation[9] provided by the authors, which can be used for sequence labeling problems with Hamming loss. We present two different results for Cascades: 1) Cascades(2012) employs the version of the code at the original time this work was done, 2) Cascades(updated) employs the most recent updated[10] version of the Cascades training

8. SGD code can be found at `http://leon.bottou.org/projects/sgd`.
9. Cascades code can be found at `http://code.google.com/p/structured-cascades/`.
10. Most recent based on personal communication with the author.

Algorithms	Data Sets						
	HW-Small	HW-Large	Stress	Phoneme	Chunk	POS	Scene labeling
a. Comparison to state-of-the-art							
$100 * (1 - \epsilon_{ei})$	73.9	83.99	77.97	77.09	88.84	92.5	**78.61**
Recurrent	65.67	74.87	72.82	73.58	88.51	92.15	56.64
LDS-Greedy	82.59	92.59	78.85	79.09	94.62	**96.93**	72.95
FB-Greedy	80.3	89.38	77.93	78.43	93.96	96.87	67.67
CRF	80.03	86.89	78.52	78.91	**94.77**	96.84	-
SVM-Struct	80.36	87.51	77.99	78.3	93.64	96.81	-
Searn	82.12^{B}	90.58^{B}	76.15	77.26	94.44^{B}	95.83	62.31
Cascades(2012)	69.62	87.95	77.18	69.77	-	96.82	-
Cascades(updated)	**86.98**	**96.78**	**79.59**	**82.44**	-	96.82	-
b. Results with Additional Search							
LDS-BST(greedy)	$\mathbf{83.81}^{+}$	$\mathbf{93.17}^{+}$	78.76	78.87	94.63	96.95	$\mathbf{74.12}^{+}$
FB-BST(greedy)	81.19^{+}	90.21^{+}	77.61	78.32	93.98	96.91	69.23^{+}
c. Results with DAgger							
LDS-Greedy	$\mathbf{83.62}^{+}$	$\mathbf{93.24}^{+}$	$\mathbf{79.81}^{+}$	$\mathbf{79.97}^{+}$	94.61	96.91	$\mathbf{74.27}^{+}$
FB-Greedy	81.28^{+}	90.45^{+}	78.96^{+}	79.23^{+}	93.94	96.89	69.63^{+}
d. Results with Third-Order Features							
LDS-Greedy	85.85^{+}	95.08^{+}	80.21^{+}	81.61^{+}	94.63	96.97	$\mathbf{74.71}^{+}$
FB-Greedy	83.18^{+}	92.66^{+}	79.23^{+}	80.65^{+}	94.17^{+}	96.94	69.81^{+}
Cascades(2012)	81.87^{+}	93.76^{+}	73.48	68.98	-	96.84	-
Cascades(updated)	$\mathbf{89.18}^{+}$	$\mathbf{97.84}^{+}$	$\mathbf{80.49}^{+}$	$\mathbf{82.59}^{+}$	-	96.84	-

Table 1: Prediction accuracy results of different structured prediction algorithms and variations of our framework. A + indicates that the particular variation being considered resulted in improvement.

code, which significantly improves on the Cascades(2012). For Searn we report the best published results with a linear classifier (i.e., linear SVMs instead of Perceptron) as indicated by B in the table and otherwise ran our own implementation of Searn with optimal approximation as described in Hal Daumé III et al. (2009) and optimized the interpolation parameter β over the validation set. Note that we do not compare our results to SampleRank due to the fact that its performance is highly dependent on the hand-designed proposal distribution, which varies from one domain to another.

Table 1a shows the prediction accuracies of different algorithms ('-' indicates that we were not able to generate results for those cases as the software package was not directly applicable). Across all benchmarks we see that the most basic instantiations of our framework, LDS-Greedy and FB-Greedy, produce results that are comparable or significantly better than all the other methods excluding[11] Cascades(updated). This is particularly interesting, since these results are achieved using a relatively small amount of search and the simplest search method, and results tend to be the same or better for our other instantiations. A likely reason that we are outperforming CRFs and SVM-Struct is that we use

11. A followup work (Doppa et al., 2014) that employs two distinct functions for guiding the search and scoring the candidate outputs generated during search performs comparably or better than Cascades(updated) across all benchmarks.

second-order features while those approaches use first-order features, since exact inference with higher order features is too costly, especially during training. As stated earlier, one of the advantages of our approach is that we can use higher-order features with negligible overhead.

Finally, the improvement in the scene labeling domain is the most significant, where Searn achieves an accuracy of 62.31 versus 72.95 for LDS-Greedy. In this domain, most prior work has considered the simpler task of classifying entire images into one of a set of discrete classes, but to the best of our knowledge no one has considered a structured prediction approach for patch classification. The only reported result for patch classification that we are aware of Vogel and Schiele (2007) obtains an accuracy of 71.7 (versus our best performance of 74.27) with non-linear SVMs trained i.i.d. on patches using more sophisticated features than ours.

5.3 Framework Variations

Adding More Search. Table 1b shows that LDS-BST(greedy) and FB-BST(greedy) are generally the same or better than LDS-Greedy and FB-Greedy, with the biggest improvements in handwriting recognition task and the challenging scene labeling ('+' indicates improvement). Results improve from 82.59 to 83.81 in HW-Small, from 92.59 to 93.17 in HW-Large and from 72.95 to 74.12 in the scene labeling task. This shows that it can be an effective strategy to train using greedy search and then insert that cost function into a more elaborate search at test time for further improvement. As noted earlier, in the domains we considered, training with a more sophisticated search procedure like beam search did not improve results over greedy search. This demonstrates the efficiency of our search spaces and can be considered as a positive result.

Exact Imitation vs. DAgger. Our experiments show that the simple exact imitation approach for cost function training performs extremely well on our problems. However, cost functions trained via exact imitation can be prone to error propagation (Kääriäinen, 2006; Ross and Bagnell, 2010). It is interesting to consider whether addressing this issue might improve results further. Therefore, we experimented with DAgger (Ross et al., 2011), a more advanced imitation training regime that addresses error propagation through on-line training and expert demonstrations. At a high-level, DAgger learns on-line from an aggregate data set collected over several iterations. The first iteration corresponds to the data produced by exact imitation of the expert. Further iterations correspond to the actions suggested by the expert on trajectories produced by a mixture of the learned policy from the previous iteration and the expert policy. This allows DAgger to learn from states visited by its possibly erroneous learned policy and correct its mistakes using expert input. In our adaptation, the "learned policy" corresponds to the decisions made by the greedy search guided by the cost function as the heuristic, and the "expert policy" corresponds to the decisions made by greedy search guided by the loss function. Ross et al. (2011) show that during the iterations of DAgger just using the learned policy without mixing the expert policy performs very well across diverse domains. Therefore, we use the same setting in our DAgger experiments.

We picked the best cost function based on a validation set after 5 iterations of DAgger, noting that no noticeable improvement was observed after 5 iterations. Table 1c shows the

results of LDS-Greedy and FB-Greedy obtained by training with DAgger. We see that there are some improvements over the cost function trained with exact imitation, although the improvements are quite small ('+' indicates improvement). As we will show later, we get much more positive results for DAgger in the context of pruned search spaces.

Higher-Order Features. One of the advantages of our framework compared to other approaches for structured prediction is the ability to use more expressive feature spaces with negligible computational overhead. Table 1d shows results using third-order features (compared to second-order results in Table 1a) for LDS-Greedy, FB-Greedy and Cascades.[12] Note that it is not practical to run the other methods (e.g., CRFs and SVM-Struct) using third-order features due to the substantial increase in inference time. The results of LDS-Greedy and FB-Greedy with third-order features improve over the corresponding results with second-order features across the board ('+' indicates improvement). Finally, we note that while Cascades(updated) is able to improve performance by using third-order features, the improvement is negligible for phoneme prediction.

LDS space vs. Flipbit space. We see that generally the instances of our method that use the LDS space outperform the corresponding instances that use the Flipbit space. Interestingly, if there is a large difference between the exact imitation accuracy $1 - \epsilon_{ei}$ and the recurrent classifier accuracy (e.g., Handwriting and Scene labeling), then the LDS space is significantly better than the flip-bit space. This is particularly true in our most complex problem of scene labeling where this difference is quite large, as is the gap between LDS and Flipbit. These results show the benefit of using the LDS space and empirically confirm our observations in Section 3 that the quality of the LDS and Flipbit spaces are related to the exact imitation and recurrent error rates respectively.

5.4 Results with Sparse Search Spaces

Recall that sparse search spaces are parameterized by k, the sparsity parameter (see Section 3.5). Small values of k lead to proportionately smaller branching factors for search. We perform experiments for different values of k to evaluate the effectiveness of sparse search spaces (i.e., LDS-k and FB-k). For example, **LDS-2** and **FB-2** correspond to the configurations where k equals 2. We only report the results for $k = 2$ and $k = 4$ noting that we didn't see major improvements for larger values of k. For all these experiments, we run greedy search for a number of steps equal to the length of the sequence during both training and testing. For greedy search, the computation time can be expected to be linearly related to k since the main computational bottleneck is the generation of $T \cdot k$ successors for each node encountered during the search.

Results of Cost Function Trained on Complete Search Spaces. Table 2b gives the results of using a cost function trained on a complete (non-sparse) search space (as in the previous experiments) to make predictions via the pruned spaces. As we can see, the gap between the results of LDS-2 and FB-2 and the corresponding results obtained using complete search space (see Table 2a) is very small. This means that we get huge speed improvements during testing with only a small loss in accuracy (more details on speedup below). The accuracy loss reduces with less sparse search spaces (LDS-4 and FB-4), but comes at the expense of more computation time.

12. Cascades code can be found at `http://code.google.com/p/structured-cascades/`.

Algorithms	Data Sets						
	HW-Small	HW-Large	Stress	Phoneme	Chunk	POS	Scene labeling
a. Accuracy results of training and testing on complete search space							
LDS	**82.59**	**92.59**	**78.85**	**79.09**	**94.62**	96.93	**72.95**
FB	80.3	89.38	77.93	78.43	93.96	96.87	67.67
b. Accuracy results of cost function trained on complete search space							
LDS-2	81.02	91.38	78.62	79.79	93.95	96.13	69.87
FB-2	79.47	86.95	77.82	79.23	93.18	96.08	65.43
LDS-4	**82.55**	**92.45**	**78.85**	**79.97**	**94.55**	96.77	**72.11**
FB-4	80.43	88.40	77.93	79.23	94.23	96.81	66.98
c. Accuracy results of cost function trained on sparse search space via Exact Imitation							
LDS-2	80.17	90.43	78.72	78.90	94.08	96.29	68.62
FB-2	78.95	87.61	77.57	78.80	94.11	96.45	63.45
LDS-4	**83.12**	**92.84**	**78.85**	79.46	94.55	96.51	**70.69**
FB-4	80.80	90.04	77.93	**79.59**	94.39	96.57	65.67
d. Accuracy results of cost function trained on sparse search space via DAgger							
LDS-2	82.54	92.14	79.27	80.57	94.27	96.38	71.56
FB-2	80.73	89.65	78.94	80.48	94.32	96.55	66.78
LDS-4	**85.53**	**94.14**	**79.81**	**81.23**	94.58	96.85	**73.61**
FB-4	82.87	91.75	78.96	81.26	94.56	96.89	68.71
e. Timing results (avg. time per greedy search step in milli seconds)							
LDS	40.0	40.0	1.0	23.0	421.0	695.0	2660.0
FB	20.0	19.0	1.0	10.0	134.0	170.0	1740.0
LDS-2	3.0	3.8	0.7	1.0	70.0	65.0	580.0
FB-2	2.0	2.0	0.6	0.7	27.0	17.0	350.0
LDS-4	7.0	7.2	1.3	2.0	160.0	140.0	1350.0
FB-4	3.6	3.6	1.2	1.3	60.0	35.0	790.0

Table 2: Prediction accuracy and timing results for greedy search comparing sparse and complete search spaces.

Results of Cost Function Trained on Sparse Search Spaces. It is natural to expect that performance on sparse search spaces might improve if the cost function is trained using the same sparse search space. Further, since conducting searches in the sparse spaces is computationally cheaper, learning directly in sparse spaces can be much more efficient. Table 2c shows the results of training the cost function on the sparse search spaces using exact imitation. As we can see, accuracies of LDS-2 and FB-2 slightly degrade compared to the corresponding results in Table 2b, but the results of LDS-4 and FB-4 equal or slightly improve in almost all cases except for scene labeling. These results show that we get speed improvements during both training and testing with little loss in accuracy.

Contrary to expectation, the above results show that when using the LDS-2 and FB-2 spaces, training directly on those spaces was often slightly worse than training on the complete spaces. One hypothesis for this observation is that the number and variation of

states encountered during training by the exact imitation approach is much less for sparser spaces. This can possibly hurt robustness of the learned cost function. This suggests that a more sophisticated approach such as DAgger might be more effective since it effectively generates a wider diversity of states during training.

Table 2d shows the results of training with DAgger, which confirm the above hypothesis. First, DAgger significantly improves over the results obtained with exact imitation (see Table 2c) across the board. Second, the results of training (via DAgger) and testing on sparse search spaces are better than the results of training on complete search spaces and testing on sparse search spaces (see Table 2b). This agrees with the intuition that training on the search space used for testing should be superior to training on a different search space. Third, the results of LDS-4 and FB-4 with DAgger are significantly better than the results obtained by training and testing on complete search spaces (see Table 2a). This indicates that training on sparse search spaces via DAgger is very effective and gives us speed improvement with no loss in accuracy and sometimes improves accuracy compared to the complete spaces.

Inference Time and Anytime Performance. Table 2e shows the timing results (avg. time per greedy search step in secs.) of our approach during testing using sparse (LDS-k and FB-k) and complete search spaces (LDS and FB). As we can see, we get speed improvement by a factor of ten (roughly) with sparse search spaces. Note that the speedup will generally be larger for problems with larger numbers of labels L, since the number of successors decreases from $T \cdot (L-1)$ to $T \cdot k$. We would like to point out that sparse search spaces will also improve the training time of our approach (fewer ranking examples in step 11 of Algorithm 2), and can be advantageous[13] compared to standard approaches including CRFs and SVM-Struct. Further, we compare the anytime curves of configurations of our approach with sparse and complete search spaces, which show the accuracy achieved by a method versus an inference time bound at prediction time. Figure 7 shows the anytime curves for all the problems except stress prediction, where there is hardly any difference due to the small label set size (5 labels). Note that all these results are for training with DAgger.

From the anytime curves, it is clear that the configurations with sparse search spaces have a much better anytime profile compared to the ones with complete search spaces. LDS-2 and FB-2 reach the respective accuracies of LDS and FB very quickly in all the cases except for POS and Scene labeling, where there is a small loss in accuracy. However, LDS-4 and FB-4 recover the accuracy losses in those two cases. These results demonstrate that sparse search spaces would be highly effective in those situations, where there is a need to make anytime predictions.

Comparing the anytime curves of LDS and FB, we can see that LDS is comparable or better than FB in all cases other than Chunking and POS.[14] This is especially true for the handwriting recognition and scene labeling problems. In the anytime curves for scene labeling task, we can see that LDS is dominant and improves accuracy much more quickly than FB. For example, a 10 second time bound for LDS achieves the same accuracy as FB using 70 seconds. This shows the benefit of using the LDS space. In the case of Chunking and POS, there is almost no difference between the accuracy of the recurrent classifier and

13. It is hard to do a fair comparison of wall clock times due to differences in implementations.
14. The experimental setup only differs in the search space (LDS or FB) employed during training and testing.

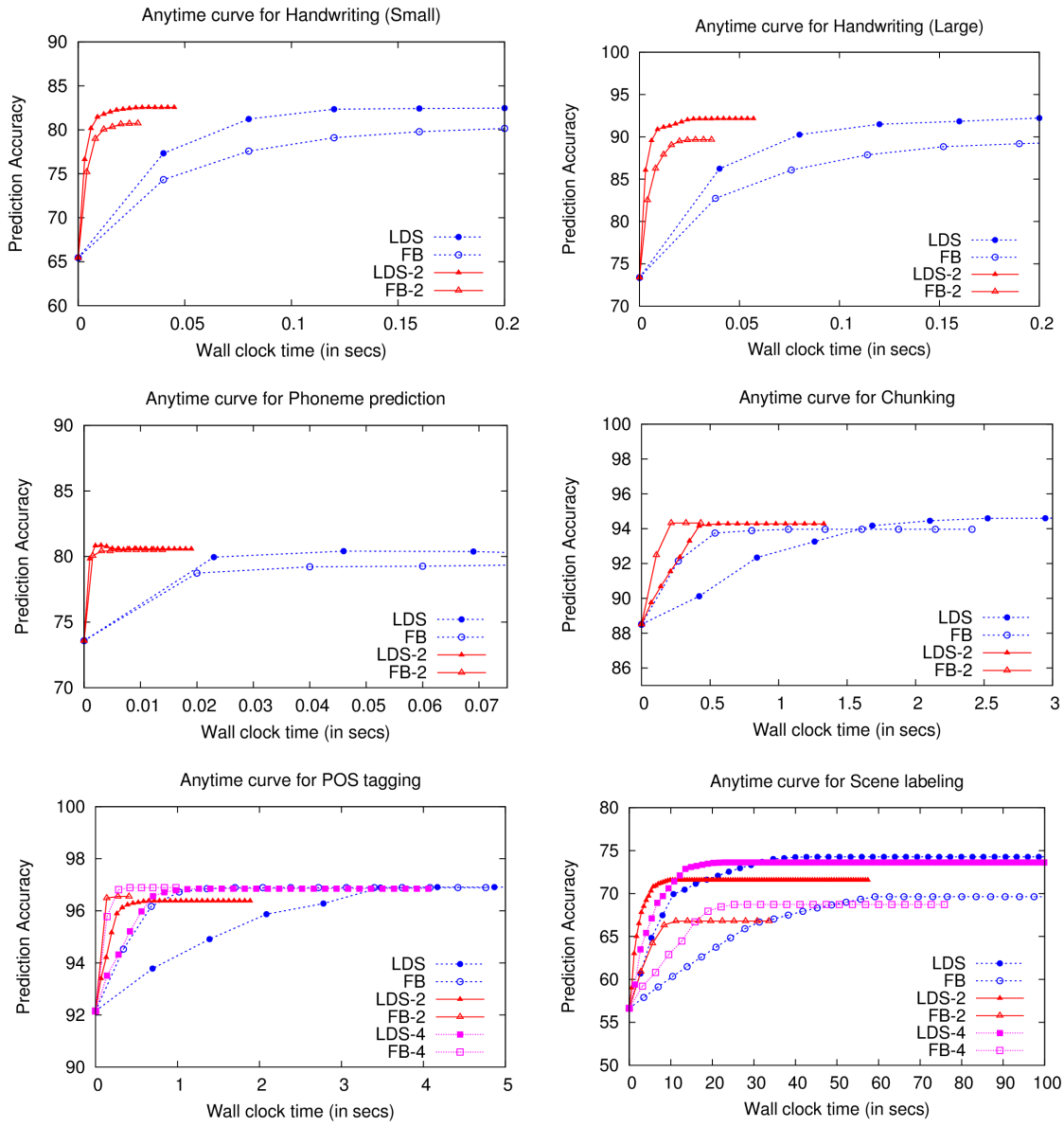


Figure 7: Anytime curves for greedy search comparing sparse and complete search spaces.

exact imitation accuracy (see Table 1a), so LDS does not provide any extra benefit over FB. Recall that there is significant additional overhead for successor generation for LDS compared to flipbit. To generate each successor, LDS must evaluate the recurrent classifier at sequence positions after discrepancies are introduced. On the other hand, the flipbit space need not evaluate the recurrent classifier during successor generation, but only uses the recurrent classifier to generate the initial state. In Chunking and POS, the additional overhead of the LDS search does not payoff in improved accuracy and the anytime curve of

flipbit is accordingly better. All these findings are true for the sparser versions of LDS and FB as well.

6. Comparison to Related Work

As described earlier, the majority of structured prediction work has focused on the use of exact (when possible) or approximate inference techniques, such as loopy belief propagation and relaxation methods, for computing outputs. Learning then is focused on tuning the cost function parameters in order to optimize various objective functions, which differ among learning algorithms (Lafferty et al., 2001; Taskar et al., 2003; Tsochantaridis et al., 2004; McAllester et al., 2010). There are also approximate cost function learning approaches that do not employ any inference routine during training. For example, piece-wise training (Sutton and McCallum, 2009), Decomposed Learning (Samdani and Roth, 2012) and its special case pseudo-max training (Sontag et al., 2010) fall under this category. These training approaches are very efficient, but they still need an inference algorithm to make predictions during testing. In these cases, one could employ the Constrained Conditional Models (CCM) framework (Chang et al., 2012) with some declarative (global) constraints to make predictions using the learned cost function. The CCM framework relies on the Integer Linear Programming (ILP) inference method (Roth and tau Yih, 2005). More recent work has attempted to integrate (approximate) inference and cost function learning in a principled manner (Meshi et al., 2010; Stoyanov et al., 2011; Hazan and Urtasun, 2012; Domke, 2013). Researchers have also worked on using higher-order features for CRFs in the context of sequence labeling under the pattern sparsity assumption (Ye et al., 2009; Qian et al., 2009). However, these approaches are not applicable for the graphical models where the sparsity assumption does not hold.

An alternative approach to addressing inference complexity is cascade training (Felzenszwalb and McAllester, 2007; Weiss and Taskar, 2010; Weiss et al., 2010), where efficient inference is achieved by performing multiple runs of inference from a coarse level to a fine level of abstraction. While such approaches have shown good success, they place some restrictions on the form of the cost functions to facilitate "cascading." Another potential drawback of cascades and most other approaches is that they either ignore the loss function of a problem (e.g., by assuming Hamming loss) or require that the loss function be decomposable in a way that supports "loss augmented inference". Our approach is sensitive to the loss function and makes minimal assumptions about it, requiring only that we have a blackbox that can evaluate it for any potential output.

Our approach is partly inspired by the alternative framework of classifier-based structured prediction. These algorithms avoid directly solving the Argmin problem by assuming that structured outputs can be generated by making a series of discrete decisions. The approach then attempts to learn a *recurrent classifier* that given an input $\mathbf{x}$ is iteratively applied in order to generate the series of decisions for producing the target output $\mathbf{y}^*$. Simple training methods (e.g., Dietterich et al., 1995) have shown good success and there are some positive theoretical guarantees (Syed and Schapire, 2010; Ross and Bagnell, 2010). However, recurrent classifiers can be prone to error propagation (Kääriäinen, 2006; Ross and Bagnell, 2010). Recent work, for example, Searn (Hal Daumé III et al., 2009), SMiLe (Ross and Bagnell, 2010), and DAgger (Ross et al., 2011), attempts to address this issue using more

sophisticated training techniques and have shown state-of-the-art structured-prediction results. However, all these approaches use classifiers to produce structured outputs through a single sequence of greedy decisions. Unfortunately, in many problems, some decisions are difficult to predict by a greedy classifier, but are crucial for good performance.

In contrast, our approach leverages recurrent classifiers to define good quality search spaces over complete outputs, which allows decision making by comparing multiple complete outputs and choosing the best. There are also non-greedy methods that learn a scoring function to search in the space of partial structured outputs (Hal Daumé III and Marcu, 2005; Daumé III, 2006; Xu et al., 2009; Huang et al., 2012; Yu et al., 2013). All these methods perform online training, and differ only in the way search errors are defined and how the weights are updated when errors occur. Unfortunately, training the scoring function can be difficult because it is hard to evaluate states with partial outputs and the theoretical guarantees of the learned scoring function (e.g., convergence and generalization results) rely on very strong assumptions (Xu et al., 2009).

A closely related framework to ours is the SampleRank framework (Wick et al., 2011) for structured prediction, which also learns a cost function for guiding search in the space of complete outputs. While it shares with our work the idea of explicit search in the output space, there are some significant differences. The SampleRank framework is mainly focused on Monte-Carlo search, and the underlying flipbit search space, whereas our approach can be instantiated for a wide range of search spaces (e.g., LDS space that leverages powerful recurrent classifiers) and rank-based search algorithms (e.g., greedy search, beam search and best-first search). We believe that this flexibility is important since it is well-understood in the search literature that the best search space formulation and the most appropriate search algorithm change from problem to problem. In addition, the SampleRank framework is highly dependent on a hand-designed "proposal distribution" for guiding the search or effectively defining the search space. In contrast, we describe a generic approach for constructing search spaces that is shown to be effective across a variety of domains.

Our approach is also related to Re-Ranking (Collins, 2002), which uses a generative model to propose a k-best list of outputs, which are then ranked by a separate ranking function. In contrast, rather than restricting to a generative model for producing potential outputs, our approach leverages generic search over efficient search spaces guided by a learned cost function that has minimal representational restrictions, and employs the same cost function to rank the candidate outputs. Recent work on generating multiple diverse solutions in a probabilistic framework can be considered as another way of producing candidate outputs. A representative set of approaches in this line of work are diverse M-best (Batra et al., 2012), M-best modes (Park and Ramanan, 2011; Chen et al., 2013) and Determinantal Point Processes (Kulesza and Taskar, 2012).

The general area of local search techniques applied to combinatorial optimization problems is very much relevant to our work. For example, STAGE (Boyan and Moore, 2000) learns an evaluation function over the states to improve the performance of search, where the value of a state corresponds to the performance of a local search algorithm starting from that state, and Zhang and Dietterich (1995) uses Reinforcement Learning (RL) methods to learn heuristics for job shop scheduling with the goal of minimizing the duration of the schedule. For combinatorial optimization problems, the cost function to be optimized is

known a priori, where as such a function is not given for structured prediction problems. Therefore, our approach learns a cost function to score the structured outputs.

7. Summary and Future Work

We studied a general framework for structured prediction based on search in the space of complete outputs.We showed how powerful classifiers can be leveraged to define an effective search space over complete outputs, and gave a generic cost function learning approach to score the outputs for any given combination of search space and search strategy. Our experimental results showed that a very small amount of search is needed to improve upon the state-of-the-art performance, validating the effectiveness of our framework. We also addressed some of the scalability issues via a simple pruning strategy that creates sparse search spaces that are more efficient to search in.

In a follow-up work, we introduce a more general framework called $\mathcal{HC}$-Search that employs two distinct functions for guiding the search and scoring the candidate outputs generated during search (Doppa et al., 2013, 2014). $\mathcal{HC}$-Search further improves upon the results in the current paper on most benchmark tasks, and performs comparably or better than Cascades. Future work includes applying this framework to more challenging problems in natural language processing and computer vision (e.g., Coreference resolution, tracking players in sports videos, and object detection in biological images (Lam et al., 2013)), studying principled ways of using domain knowledge (e.g., nearly sound constraints) for pruning, and exploring algorithms to train for anytime performance by trading off speed and accuracy as part of the learning objective (Grubb and Bagnell, 2012; Xu et al., 2012).

Acknowledgments

The authors would like to thank the anonymous reviewers for their feedback. The first author would also like to thank Tom Dietterich for his encouragement and support throughout this work. This work was supported in part by NSF grants IIS 1219258, IIS 1018490 and in part by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. FA8750-13-2-0033. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF, the DARPA, the Air Force Research Laboratory (AFRL), or the US government. Some of the material in this article was first published at ICML-2012 (Doppa et al., 2012).

References

Dhruv Batra, Payman Yadollahpour, Abner Guzmán-Rivera, and Gregory Shakhnarovich. Diverse m-best solutions in Markov random fields. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 1–16, 2012.

Justin A. Boyan and Andrew W. Moore. Learning evaluation functions to improve optimization by local search. *Journal of Machine Learning Research (JMLR)*, 1:77–112, 2000.

Ming-Wei Chang, Lev-Arie Ratinov, and Dan Roth. Structured learning with constrained conditional models. *Machine Learning Journal (MLJ)*, 88(3):399–431, 2012.

Chao Chen, Vladimir Kolmogorov, Yan Zhu, Dimitris Metaxas, and Christoph H. Lampert. Computing the M most probable modes of a graphical model. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013.

Michael Collins. Ranking algorithms for named entity extraction: Boosting and the voted perceptron. In *Proceedings of Association of Computational Linguistics (ACL) Conference*, 2002.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*, 7:551–585, 2006.

Hal Daumé III. *Practical Structured Learning Techniques for Natural Language Processing.* PhD thesis, University of Southern California, Los Angeles, CA, 2006.

Thomas G. Dietterich, Hermann Hild, and Ghulum Bakiri. A comparison of ID3 and backpropagation for English text-to-speech mapping. *Machine Learning Journal (MLJ)*, 18(1):51–80, 1995.

Justin Domke. Structured learning via logistic regression. In *Advances in Neural Information Processing Systems (NIPS)*, pages 647–655, 2013.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. Output space search for structured prediction. In *Proceedings of International Conference on Machine Learning (ICML)*, 2012.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. HC-Search: Learning heuristics and cost functions for structured prediction. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2013.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. HC-Search: A learning framework for search-based structured prediction. *To appear in Journal of Artificial Intelligence Research (JAIR)*, 2014.

Pedro F. Felzenszwalb and David A. McAllester. The generalized A* architecture. *Journal of Artificial Intelligence Research (JAIR)*, 29:153–190, 2007.

Alan Fern, Sung Wook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias: Solving relational Markov decision processes. *Journal of Artificial Intelligence Research (JAIR)*, 25:75–118, 2006.

Alexander Grubb and Drew Bagnell. Speedboost: Anytime prediction with uniform near-optimality. *Journal of Machine Learning Research - Proceedings Track*, 22:458–466, 2012.

Hal Daumé III and Daniel Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of International Conference on Machine Learning (ICML)*, 2005.

Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning Journal (MLJ)*, 75(3):297–325, 2009.

William D. Harvey and Matthew L. Ginsberg. Limited discrepancy search. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 607–615, 1995.

Tamir Hazan and Raquel Urtasun. Efficient learning of structured predictors in general graphical models. *CoRR*, abs/1210.2346, 2012.

Liang Huang, Suphan Fayong, and Yang Guo. Structured perceptron with inexact search. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, pages 142–151, 2012.

Matti Kääriäinen. Lower bounds for reductions. In *Atomic Learning Workshop*, 2006.

Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer, and Dan Chazan. Phoneme alignment based on discriminative learning. In *Proceedings of Annual Conference of the International Speech Communication Association (Interspeech)*, pages 2961–2964, 2005.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2-3):123–286, 2012.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

Michael Lam, Janardhan Rao Doppa, Xu Hu, Sinisa Todorovic, Thomas Dietterich, Abigail Reft, and Marymegan Daly. Learning to detect basal tubules of nematocysts in SEM images. In *Proceedings of ICCV Workshop on Computer Vision for Accelerated Biosciences (CVAB)*. IEEE, 2013.

David A. McAllester, Tamir Hazan, and Joseph Keshet. Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1594–1602, 2010.

Ofer Meshi, David Sontag, Tommi Jaakkola, and Amir Globerson. Learning efficiently with approximate inference via dual losses. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 783–790, 2010.

Dennis Park and Deva Ramanan. N-best maximal decoders for part models. In *Proccedings of IEEE International Conference on Computer Vision (ICCV)*, pages 2627–2634, 2011.

Xian Qian, Xiaoqian Jiang, Qi Zhang, Xuanjing Huang, and Lide Wu. Sparse higher order conditional random fields for improved sequence labeling. In *Proceedings of International Conference on Machine Learning (ICML)*, 2009.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. *Journal of Machine Learning Research - Proceedings Track*, 9:661–668, 2010.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research - Proceedings Track*, 15:627–635, 2011.

Dan Roth and Wen tau Yih. Integer linear programming inference for conditional random fields. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 736–743, 2005.

Rajhans Samdani and Dan Roth. Efficient decomposed learning for structured prediction. In *Proceedings of International Conference on Machine Learning (ICML)*, 2012.

Terrence J. Sejnowski and Charles R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.

David Sontag, Ofer Meshi, Tommi Jaakkola, and Amir Globerson. More data means less inference: A pseudo-max approach to structured learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2181–2189, 2010.

Veselin Stoyanov, Alexander Ropson, and Jason Eisner. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 725–733, 2011.

Charles A. Sutton and Andrew McCallum. Piecewise training for structured prediction. *Machine Learning Journal (MLJ)*, 77(2-3):165–194, 2009.

Umar Syed and Rob Schapire. A reduction from apprenticeship learning to classification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2253–2261, 2010.

Benjamin Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of International Conference on Machine Learning (ICML)*, 2004.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, 2005.

Julia Vogel and Bernt Schiele. Semantic modeling of natural scenes for content-based image retrieval. *International Journal of Computer Vision (IJCV)*, 72(2):133–157, 2007.

David Weiss and Benjamin Taskar. Structured prediction cascades. *Journal of Machine Learning Research - Proceedings Track*, 9:916–923, 2010.

David Weiss, Ben Sapp, and Ben Taskar. Sidestepping intractable inference with structured ensemble cascades. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2415–2423, 2010.

Michael L. Wick, Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, and Andrew McCallum. Samplerank: Training factor graphs with atomic gradients. In *Proceedings of International Conference on Machine Learning (ICML)*, 2011.

Yuehua Xu, Alan Fern, and Sung Wook Yoon. Learning linear ranking functions for beam search with application to planning. *Journal of Machine Learning Research (JMLR)*, 10: 1571–1610, 2009.

Zhixiang Xu, Kilian Weinberger, and Olivier Chapelle. The greedy miser: Learning under test-time budgets. In *Proceedings of International Conference on Machine Learning (ICML)*, 2012.

Nan Ye, Wee Sun Lee, Hai Leong Chieu, and Dan Wu. Conditional random fields with high-order features for sequence labeling. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2196–2204, 2009.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. Max-violation perceptron and forced decoding for scalable MT training. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1112–1123, 2013.

Wei Zhang and Thomas G. Dietterich. A reinforcement learning approach to job-shop scheduling. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1114–1120, 1995.

Journal of Machine Learning Research 15 (2014) 1351-1369 Submitted 7/13; Revised 11/13; Published 4/14

Fully Simplified Multivariate Normal Updates in Non-Conjugate Variational Message Passing

Matt P. Wand MATT.WAND@UTS.EDU.AU
School of Mathematical Sciences
University of Technology, Sydney
P.O. Box 123, Broadway NSW 2007, Australia

Editor: David M. Blei

Abstract

Fully simplified expressions for Multivariate Normal updates in non-conjugate variational message passing approximate inference schemes are obtained. The simplicity of these expressions means that the updates can be achieved very efficiently. Since the Multivariate Normal family is the most common for approximating the joint posterior density function of a continuous parameter vector, these fully simplified updates are of great practical benefit.

Keywords: Bayesian computing, graphical models, matrix differential calculus, mean field variational Bayes, variational approximation

1. Introduction

Recently Knowles and Minka (2011) proposed a prescription for handling non-conjugate exponential family factors in variational message passing approximate inference schemes. Dubbed *non-conjugate variational message passing*, it widens the scope of tractable models for variational message passing and mean field variational Bayes in general. For a given exponential family factor, the non-conjugate variational message passing updates depend on the inverse covariance matrix of the natural statistic and derivatives of the non-entropy component of the Kullback-Leibler divergence.

The Multivariate Normal distribution is the most common multivariate exponential family distribution and a prime candidate for approximating the joint posterior density function of a continuous parameter vector, such as a set of regression coefficients. Knowles and Minka (2011) provide formulae for Univariate Normal updates, which correspond to less accurate diagonal covariance matrix approximations to joint posterior density functions. However, when combined with the derived variable infrastructure described Appendix A of Minka and Winn (2008), the Univariate Normal updates in Knowles and Minka (2011) are able to produce full covariance matrix Multivariate Normal approximations for regression models. This fact is utilized by the Infer.NET computational framework (Minka et al., 2013), although the mathematical description of the updates is somewhat verbose. This aspect hinders extension to more complicated models, including those not supported by Infer.NET.

Recently, Tan and Nott (2013) utilized non-conjugate variational message passing for approximate Bayesian inference in hierarchical generalized linear mixed models. Their numerical studies showed that their variational algorithms can achieve high levels of accuracy.

This accuracy is partly due to their use of Multivariate Normal, rather than Univariate Normal, factors.

This article's main contribution is full simplification of the inverse covariance matrix of the natural statistic and then to show that the updates admit a particularly simple form in terms of derivatives with respect to the common Multivariate Normal parameters, that is, the mean and covariance matrix. When combined with an additional novel matrix result, this article's second theorem, non-conjugate mean field variational Bayes algorithms involving Multivariate Normal updates are straightforward to derive and implement. This explicitness allows much easier accommodation of Multivariate Normal posterior density functions within the non-conjugate variational message passing framework. Algorithm 3 of Tan and Nott (2013) relies on Theorems 1 and 2, presented in Section 4. This leads to considerable computational efficiency for the methodology in Tan and Nott (2013).

Non-conjugate variational message passing (Knowles and Minka, 2011) is one of several recent contributions aimed at widening the set of models that can be handled via the mean field variational Bayes paradigm. Others include Braun and McAuliffe (2010), Wand et al. (2011) and Wang and Blei (2013).

Section 2 lays out notation needed for the main theorems, which are presented in Section 4. The utility of these theorems is then illustrated in Section 5 for a Bayesian Poisson mixed model and a heteroscedastic additive model. A series of appendices contains proofs of the theorems and other mathematical details.

2. Notation

The main results makes ample use of the matrix differential calculus technology of Magnus and Neudecker (1999). Therefore, I mainly adhere to their notation.

2.1 The vec, vech and Duplication Matrix Notations

If $\boldsymbol{A}$ is a $d \times d$ matrix then $\text{vec}(\boldsymbol{A})$ denotes the $d^2 \times 1$ vector obtained by stacking the columns of $\boldsymbol{A}$ underneath each other in order from left to right. Also, $\text{vech}(\boldsymbol{A})$ denotes the $\frac{1}{2}\,d(d+1) \times 1$ vector obtained from $\text{vec}(\boldsymbol{A})$ by eliminating each of the above-diagonal entries of $\boldsymbol{A}$. For example,

$$\text{vec}\left(\begin{bmatrix} 5 & 2 \\ 9 & 4 \end{bmatrix}\right) = \begin{bmatrix} 5 \\ 9 \\ 2 \\ 4 \end{bmatrix} \quad \text{while} \quad \text{vech}\left(\begin{bmatrix} 5 & 2 \\ 9 & 4 \end{bmatrix}\right) = \begin{bmatrix} 5 \\ 9 \\ 4 \end{bmatrix}.$$

If $\boldsymbol{A}$ is a symmetric, but otherwise arbitrary $d \times d$ matrix, then $\text{vech}(\boldsymbol{A})$ contains each of the distinct entries of $\boldsymbol{A}$ whereas $\text{vec}(\boldsymbol{A})$ repeats the off-diagonal entries. It follows that there is a unique $d^2 \times \frac{1}{2}d(d+1)$ matrix $\boldsymbol{D}_d$ of zeros and ones such that

$$\boldsymbol{D}_d\,\text{vech}(\boldsymbol{A}) = \text{vec}(\boldsymbol{A}) \quad \text{for} \quad \boldsymbol{A} = \boldsymbol{A}^T$$

and is called the *duplication matrix of order d*. The Moore-Penrose inverse of $\boldsymbol{D}_d$ is

$$\boldsymbol{D}_d^+ \equiv (\boldsymbol{D}_d^T\,\boldsymbol{D}_d)^{-1}\boldsymbol{D}_d^T.$$

Note that

$$\boldsymbol{D}_d^+ \operatorname{vec}(\boldsymbol{A}) = \operatorname{vech}(\boldsymbol{A}) \quad \text{for} \quad \boldsymbol{A} = \boldsymbol{A}^T.$$

The simplest non-trivial examples of $\boldsymbol{D}_d$ and $\boldsymbol{D}_d^+$ are

$$\boldsymbol{D}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{D}_2^+ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Note that, for general d, $\boldsymbol{D}_d$ can be obtained via the `duplication.matrix()` function in the package `matrixcalc` (Novomestky, 2008) within the R computing environment (R Development Core Team, 2013).

If $\boldsymbol{a}$ is a $d^2 \times 1$ vector then $\operatorname{vec}^{-1}(\boldsymbol{a})$ is defined to be the $d \times d$ matrix formed from listing the entries of $\boldsymbol{a}$ in a column-wise fashion in order from left to right. Note that vec^{-1} is the usual function inverse when the domain of vec is restricted to square matrices. In particular,

$$\operatorname{vec}^{-1}(\operatorname{vec}(\boldsymbol{A})) = \boldsymbol{A} \quad \text{for } d \times d \text{ matrices } \boldsymbol{A}$$

and

$$\operatorname{vec}(\operatorname{vec}^{-1}(\boldsymbol{a})) = \boldsymbol{a} \quad \text{for } d^2 \times 1 \text{ vectors } \boldsymbol{a}.$$

There are numerous identities involving vec, vech, $\boldsymbol{D}_d$ and $\boldsymbol{D}_d^+$, and some of these are given in Chapter 3 of Magnus and Neudecker (1999). One that is relevant to the current article is:

Lemma 1 *If $\boldsymbol{A}$ is a symmetric $d \times d$ matrix then*

$$\operatorname{vec}(\boldsymbol{A}) = \boldsymbol{D}_d^{+T} \boldsymbol{D}_d^T \operatorname{vec}(\boldsymbol{A}).$$

2.2 The diagonal and diag Notations

If $\boldsymbol{A}$ is a $d \times d$ matrix then diagonal($\boldsymbol{A}$) denotes the $d \times 1$ vector containing the diagonal entries of $\boldsymbol{A}$. If $\boldsymbol{a}$ is a $d \times 1$ vector then diag($\boldsymbol{a}$) is the $d \times d$ matrix with the entries of $\boldsymbol{a}$ on the diagonal and all other entries equal to zero. For example,

$$\operatorname{diagonal}\left(\begin{bmatrix} 8 & 1 & -7 \\ 3 & 6 & 24 \\ -4 & 11 & -9 \end{bmatrix}\right) = \begin{bmatrix} 8 \\ 6 \\ -9 \end{bmatrix} \quad \text{and} \quad \operatorname{diag}\left(\begin{bmatrix} -4 \\ 7 \\ 31 \end{bmatrix}\right) = \begin{bmatrix} -4 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 31 \end{bmatrix}.$$

2.3 Derivative Vector and Hessian Matrix Notation

Let f be a $\mathbb{R}^p$-valued function with argument $\boldsymbol{x} \in \mathbb{R}^d$. The *derivative vector* of f with respect to $\boldsymbol{x}$, $\mathsf{D}_{\boldsymbol{x}} f$, is the $p \times d$ matrix whose (i, j) entry is

$$\frac{\partial f_i(\boldsymbol{x})}{\partial x_j}.$$

where $f_i(\boldsymbol{x})$ is the ith entry of $f(\boldsymbol{x})$ and x_j is the jth entry of $\boldsymbol{x}$. For example

$$\begin{aligned}\mathsf{D}_{\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right]} \begin{bmatrix} \tan(x_1 + 7x_2) \\ 3x_1^4(8 + 9x_2^3) \end{bmatrix} &= \begin{bmatrix} \frac{\partial}{\partial x_1}\{\tan(x_1 + 7x_2)\} & \frac{\partial}{\partial x_2}\{\tan(x_1 + 7x_2)\} \\ \frac{\partial}{\partial x_1}\{3x_1^4(8 + 9x_2^3)\} & \frac{\partial}{\partial x_2}\{3x_1^4(8 + 9x_2^3)\} \end{bmatrix} \\ &= \begin{bmatrix} \sec^2(x_1 + 7x_2) & 7\sec^2(x_1 + 7x_2) \\ 12x_1^3(8 + 9x_2^3) & 81x_1^4x_2^2 \end{bmatrix}.\end{aligned}$$

In the case $p = 1$, the *Hessian matrix* of f with respect to $\boldsymbol{x}$, $\mathsf{H}_{\boldsymbol{x}} f$, is the $d \times d$ matrix

$$\mathsf{H}_{\boldsymbol{x}} f = \mathsf{D}_{\boldsymbol{x}}\{(\mathsf{D}_{\boldsymbol{x}} f)^T\}.$$

3. Non-Conjugate Variational Message Passing

Non-conjugate variational message passing (Knowles and Minka, 2011) is an extension of mean field variational Bayes (e.g. Wainwright and Jordan, 2008) where, due to difficulties arising from non-conjugacy, one or more density functions is forced to have a particular exponential family distribution.

Consider a hierarchical Bayesian model with data vector $\boldsymbol{y}$ and parameter vectors $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. Mean field variational Bayes approximates the joint posterior density function $p(\boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{y})$ by

$$q_{\boldsymbol{\theta}_1}(\boldsymbol{\theta}_1) \cdots q_{\boldsymbol{\theta}_M}(\boldsymbol{\theta}_M)\, q_{\boldsymbol{\phi}}(\boldsymbol{\phi}), \tag{1}$$

where $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M\}$ is a partition of $\boldsymbol{\theta}$ and each subscripted q is an unrestricted density function. The solutions satisfy

$$\begin{aligned} &q^*_{\boldsymbol{\theta}_i}(\boldsymbol{\theta}_i) \propto \exp[E_{q(-\boldsymbol{\theta}_i)}\{\log p\,(\boldsymbol{\theta}_i|\boldsymbol{y}, \boldsymbol{\theta}\backslash\boldsymbol{\theta}_i, \boldsymbol{\phi})\}], \quad 1 \le i \le M, \\ &q^*_{\boldsymbol{\phi}}(\boldsymbol{\phi}) \propto \exp[E_{q(-\boldsymbol{\phi})}\{\log p\,(\boldsymbol{\phi}|\boldsymbol{y}, \boldsymbol{\theta})\}], \end{aligned}$$

where $\boldsymbol{\theta}\backslash\boldsymbol{\theta}_i$ means $\boldsymbol{\theta}$ with $\boldsymbol{\theta}_i$ excluded and $E_{q(-\boldsymbol{\theta}_i)}$ denotes expectation with respect to the q-densities of all parameters except $\boldsymbol{\theta}_i$. A similar definition applies to $E_{q(-\boldsymbol{\phi})}$.

In the event that $E_{q(-\boldsymbol{\phi})}\{\log p\,(\boldsymbol{\phi}|\boldsymbol{y}, \boldsymbol{\theta})\}$ is intractable, non-conjugate variational message passing offers a way out by replacing (1) with

$$q_{\boldsymbol{\theta}_1}(\boldsymbol{\theta}_1) \cdots q_{\boldsymbol{\theta}_M}(\boldsymbol{\theta}_M)\, q_{\boldsymbol{\phi}}(\boldsymbol{\phi}; \boldsymbol{\eta}),$$

where $q_{\boldsymbol{\phi}}(\boldsymbol{\phi}; \boldsymbol{\eta})$ is an exponential family density function with *natural parameter vector* $\boldsymbol{\eta}$ and *natural statistic* $\boldsymbol{T}(\boldsymbol{\phi})$. Then, with backing from Theorem 1 of Knowles and Minka (2011), the optimal densities $q^*(\theta_1), \ldots, q^*(\theta_M)$ and $q^*(\boldsymbol{\phi}; \boldsymbol{\eta})$ may be found using

$$\begin{aligned} &q^*_{\boldsymbol{\theta}_i}(\boldsymbol{\theta}_i) \propto \exp[E_{q(-\boldsymbol{\theta}_i)}\{\log p(\theta_i|\boldsymbol{y}, \boldsymbol{\theta}\backslash\boldsymbol{\theta}_i, \boldsymbol{\phi})\}], \quad 1 \le i \le M, \\ &\boldsymbol{\eta} \leftarrow [\,\mathrm{var}\{\boldsymbol{T}(\boldsymbol{\phi})\}]^{-1}\, [\mathsf{D}_{\boldsymbol{\eta}}\, E_{q(\boldsymbol{\theta},\boldsymbol{\phi})}\{\log p(\boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\phi})\}]^T. \end{aligned} \tag{2}$$

Here and elsewhere $\mathrm{var}(\boldsymbol{v})$ denotes the covariance matrix of a random vector $\boldsymbol{v}$. As pointed out in Knowles and Minka (2011), the graphical structure of the hierarchical Bayesian model can be used to provide a simpler expression for $\mathsf{D}_{\boldsymbol{\eta}}\, E_{q(\boldsymbol{\theta},\boldsymbol{\phi})}\{\log p(\boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\phi})\}$ that only depends on factors of $p(\boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\phi})$ involving $\boldsymbol{\phi}$.

3.1 Multivariate Normal Factor

Now consider the special case where $q(\boldsymbol{\phi};\boldsymbol{\eta})$ corresponds to a d-dimensional Multivariate Normal density function. Then the natural statistic (defined in Section 4) is

$$\boldsymbol{T}(\boldsymbol{\phi}) \equiv \left[\begin{array}{c} \boldsymbol{\phi} \\ \mathrm{vech}(\boldsymbol{\phi}\,\boldsymbol{\phi}^T) \end{array} \right].$$

Since $\boldsymbol{T}(\boldsymbol{\phi})$ has $d + d(d+1)/2$ entries, the number of entries in $\mathrm{var}\{\boldsymbol{T}(\boldsymbol{\phi})\}$ is quartic in d. Consequently, for large d, the $\boldsymbol{\eta}$ update in (2) is numerically challenging if done directly. In Section 4 I present theoretical results that allow explicit updating without the need for inversion of $\mathrm{var}\{\boldsymbol{T}(\boldsymbol{\phi})\}$. I also present results in terms of the common Multivariate Normal parametrization, involving mean vectors and covariance matrices.

4. Main Results

Consider a generic Multivariate Normal $d \times 1$ random vector $\boldsymbol{x}$

$$\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{3}$$

Then the density function of $\boldsymbol{x}$ is

$$\begin{aligned} p(\boldsymbol{x}) &= (2\pi)^{-d/2}|\boldsymbol{\Sigma}|^{-1/2}\exp\{-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\} \\ &= \exp\{\boldsymbol{T}(\boldsymbol{x})^T\boldsymbol{\eta} - A(\boldsymbol{\eta}) - \tfrac{d}{2}\log(2\pi)\}. \end{aligned}$$

Here

$$\boldsymbol{T}(\boldsymbol{x}) \equiv \left[\begin{array}{c} \boldsymbol{x} \\ \mathrm{vech}(\boldsymbol{x}\,\boldsymbol{x}^T) \end{array} \right], \quad \boldsymbol{\eta} \equiv \left[\begin{array}{c} \boldsymbol{\eta}_1 \\ \boldsymbol{\eta}_2 \end{array} \right] \equiv \left[\begin{array}{c} \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ -\frac{1}{2}\,\boldsymbol{D}_d^T\mathrm{vec}(\boldsymbol{\Sigma}^{-1}) \end{array} \right] \tag{4}$$

defines the *natural statistic* and *natural parameter* pairing and

$$A(\boldsymbol{\eta}) = -\tfrac{1}{4}\,\boldsymbol{\eta}_1^T\left\{\mathrm{vec}^{-1}(\boldsymbol{D}_d^{+T}\boldsymbol{\eta}_2)\right\}^{-1}\boldsymbol{\eta}_1 - \tfrac{1}{2}\log\left|-2\,\mathrm{vec}^{-1}(\boldsymbol{D}_d^{+T}\boldsymbol{\eta}_2)\right|$$

is the *log-partition* function.

Note that the inverse of the natural parameter transformation is

$$\left\{ \begin{array}{ll} \boldsymbol{\mu} = & -\frac{1}{2}\left\{\mathrm{vec}^{-1}(\boldsymbol{D}_d^{+T}\boldsymbol{\eta}_2)\right\}^{-1}\boldsymbol{\eta}_1 \\ \boldsymbol{\Sigma} = & -\frac{1}{2}\left\{\mathrm{vec}^{-1}(\boldsymbol{D}_d^{+T}\boldsymbol{\eta}_2)\right\}^{-1} \end{array} \right. \tag{5}$$

and can be derived from (4) using Lemma 1.

Theorem 1 *Consider the $d \times 1$ random vector $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with natural statistic vector $T(\boldsymbol{x})$ and natural parameter vector $\boldsymbol{\eta}$ given by (4) and define*

$$\boldsymbol{U} \equiv \left\{\mathsf{D}_{\boldsymbol{\eta}}\left[\begin{array}{c} \boldsymbol{\mu} \\ \mathrm{vec}(\boldsymbol{\Sigma}) \end{array} \right]\right\}^T, \quad \boldsymbol{V} \equiv \mathrm{var}\{T(\boldsymbol{x})\} = \mathsf{H}_{\boldsymbol{\eta}}A(\boldsymbol{\eta}),$$

$$\boldsymbol{M} \equiv 2\,\boldsymbol{D}_d^+(\boldsymbol{\mu} \otimes \boldsymbol{I}_d) \quad \textit{and} \quad \boldsymbol{S} \equiv 2\boldsymbol{D}_d^+(\boldsymbol{\Sigma} \otimes \boldsymbol{\Sigma})\boldsymbol{D}_d^{+T}.$$

Then

$$(a) \qquad \boldsymbol{U} = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} \\ \boldsymbol{M}\boldsymbol{\Sigma} & \boldsymbol{S}\boldsymbol{D}_d^T \end{bmatrix},$$

$$(b) \qquad \boldsymbol{S}^{-1} = \tfrac{1}{2}\boldsymbol{D}_d^T(\boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{\Sigma}^{-1})\boldsymbol{D}_d,$$

$$(c) \qquad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Sigma}\boldsymbol{M}^T \\ \boldsymbol{M}\boldsymbol{\Sigma} & \boldsymbol{S} + \boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T \end{bmatrix},$$

$$(d) \qquad \boldsymbol{V}^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}^{-1} + \boldsymbol{M}^T\boldsymbol{S}^{-1}\boldsymbol{M} & -\boldsymbol{M}^T\boldsymbol{S}^{-1} \\ -\boldsymbol{S}^{-1}\boldsymbol{M} & \boldsymbol{S}^{-1} \end{bmatrix},$$

$$(e) \qquad \boldsymbol{V}^{-1}\boldsymbol{U} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{M}^T\boldsymbol{D}_d^T \\ \boldsymbol{0} & \boldsymbol{D}_d^T \end{bmatrix}$$

and

$$(f) \qquad \boldsymbol{V}^{-1}\boldsymbol{U}\begin{bmatrix} \boldsymbol{g} \\ \mathrm{vec}(\boldsymbol{G}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{g} - 2\,\boldsymbol{G}\boldsymbol{\mu} \\ \boldsymbol{D}_d^T\mathrm{vec}(\boldsymbol{G}) \end{bmatrix}$$

for every $d \times 1$ vector $\boldsymbol{g}$ and symmetric $d \times d$ matrix $\boldsymbol{G}$.

Appendix A contains a proof of Theorem 1.

Let s be a smooth function of $\boldsymbol{\eta}$, the natural parameter vector in a Multivariate Normal factor, and consider an iterative scheme with updates of the form

$$\boldsymbol{\eta} \leftarrow \boldsymbol{V}^{-1}(\mathsf{D}_{\boldsymbol{\eta}}\, s)^T. \tag{6}$$

Note that the update in (2) is a special case of (6) with $s(\boldsymbol{\eta}) = \mathsf{D}_{\boldsymbol{\eta}}\, E_{q(\boldsymbol{\theta},\boldsymbol{\phi})}\{\log\, p(\boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{\eta})\}$. By the chain rule of matrix differential calculus (Theorem 8, Chapter 5, of Magnus and Neudecker, 1999)

$$\boldsymbol{V}^{-1}(\mathsf{D}_{\boldsymbol{\eta}}\, s)^T = \boldsymbol{V}^{-1}\left[\left\{\mathsf{D}_{\left[\begin{smallmatrix}\boldsymbol{\mu}\\ \mathrm{vec}(\boldsymbol{\Sigma})\end{smallmatrix}\right]}\, s\right\}\left\{\mathsf{D}_{\boldsymbol{\eta}}\begin{bmatrix}\boldsymbol{\mu}\\ \mathrm{vec}(\boldsymbol{\Sigma})\end{bmatrix}\right\}\right]^T = \boldsymbol{V}^{-1}\boldsymbol{U}\begin{bmatrix}(\mathsf{D}_{\boldsymbol{\mu}}\, s)^T \\ (\mathsf{D}_{\mathrm{vec}(\boldsymbol{\Sigma})}\, s)^T\end{bmatrix}.$$

Let $(\boldsymbol{\mu}_{\mathrm{old}}, \boldsymbol{\Sigma}_{\mathrm{old}})$ and $(\boldsymbol{\mu}_{\mathrm{new}}, \boldsymbol{\Sigma}_{\mathrm{new}})$, respectively, denote the old and new values of $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ in the updating scheme (6). Then, it follows from Theorem 1(f) that

$$\begin{aligned} \boldsymbol{\Sigma}_{\mathrm{new}}^{-1}\boldsymbol{\mu}_{\mathrm{new}} &= \left[(\mathsf{D}_{\boldsymbol{\mu}}\, s)^T - 2\,\mathrm{vec}^{-1}\left((\mathsf{D}_{\mathrm{vec}(\boldsymbol{\Sigma})}\, s)^T\right)\boldsymbol{\mu}\right]_{\boldsymbol{\mu}=\boldsymbol{\mu}_{\mathrm{old}},\boldsymbol{\Sigma}=\boldsymbol{\Sigma}_{\mathrm{old}}} \\ \text{and} \quad \boldsymbol{D}_d^T\mathrm{vec}(-\tfrac{1}{2}\boldsymbol{\Sigma}_{\mathrm{new}}^{-1}) &= \left[\boldsymbol{D}_d^T(\mathsf{D}_{\mathrm{vec}(\boldsymbol{\Sigma})}\, s)^T\right]_{\boldsymbol{\mu}=\boldsymbol{\mu}_{\mathrm{old}},\boldsymbol{\Sigma}=\boldsymbol{\Sigma}_{\mathrm{old}}}. \end{aligned}$$

The mean and covariance parameter updates are therefore given by

$$\begin{aligned} \boldsymbol{\Sigma}_{\text{new}} &= \{-2\,\text{vec}^{-1}(\{[\mathsf{D}_{\text{vec}(\boldsymbol{\Sigma})}\, s]_{\boldsymbol{\mu}=\boldsymbol{\mu}_{\text{old}},\boldsymbol{\Sigma}=\boldsymbol{\Sigma}_{\text{old}}}\}^T)\}^{-1} \\ \text{and} \quad \boldsymbol{\mu}_{\text{new}} &= \boldsymbol{\mu}_{\text{old}} + \boldsymbol{\Sigma}_{\text{new}}([\mathsf{D}_{\boldsymbol{\mu}}\, s]_{\boldsymbol{\mu}=\boldsymbol{\mu}_{\text{old}},\boldsymbol{\Sigma}=\boldsymbol{\Sigma}_{\text{old}}})^T. \end{aligned}$$

It follows that (6) is equivalent to the updates:

$$\left\{ \begin{array}{ll} \boldsymbol{\Sigma} \leftarrow & \left\{ -2\,\text{vec}^{-1}((\mathsf{D}_{\text{vec}(\boldsymbol{\Sigma})}\, s)^T) \right\}^{-1} \\ \boldsymbol{\mu} \leftarrow & \boldsymbol{\mu} + \boldsymbol{\Sigma}\,(\mathsf{D}_{\boldsymbol{\mu}}\, s)^T. \end{array} \right. \tag{7}$$

The simplified form of $\boldsymbol{V}^{-1}\boldsymbol{U}$ in Theorem 1 can be explained via the inverse relationship that exists between $\boldsymbol{V} = \mathsf{H}_{\boldsymbol{\eta}} A(\boldsymbol{\eta})$ and the derivative of the *mean parameter* vector $E\{\boldsymbol{T}(\boldsymbol{x})\}$ with respect to the natural parameter vector $\boldsymbol{\eta}$. This relationship is pointed out in Section 4.1 of Hensman et al. (2012). Note that my $\boldsymbol{U}$ involves the derivative of $[\boldsymbol{\mu}^T\ \text{vec}(\boldsymbol{\Sigma})^T]^T$, rather than $E\{\boldsymbol{T}(\boldsymbol{x})\}$, with respect to $\boldsymbol{\eta}$ in the chain rule. This corresponds to differentiation of s with respect to the more convenient $\text{vec}(\boldsymbol{\Sigma})$.

The update for $\boldsymbol{\Sigma}$, given at (7), involves $\text{vec}^{-1}((\mathsf{D}_{\text{vec}(\boldsymbol{\Sigma})}\, s)^T)$. Simplification of this expression for regression models is aided by:

Theorem 2 *Let $\boldsymbol{A}$ be an $n \times d$ matrix, $\boldsymbol{B}$ be a $d \times d$ matrix and $\boldsymbol{b}$ be an $n \times 1$ vector. Define*

$$\mathcal{Q}(\boldsymbol{A}) \equiv (\boldsymbol{A} \otimes \boldsymbol{1}^T) \odot (\boldsymbol{1}^T \otimes \boldsymbol{A})$$

where $\boldsymbol{1}$ is the $d \times 1$ vector with all entries equal to 1. Then

$$(a) \qquad \text{diagonal}\,(\boldsymbol{A}\boldsymbol{B}\boldsymbol{A}^T) = \mathcal{Q}(\boldsymbol{A})\,\text{vec}(\boldsymbol{B})$$

and

$$(b) \qquad \text{vec}\,(\boldsymbol{A}^T \text{diag}(\boldsymbol{b})\boldsymbol{A}) = \mathcal{Q}(\boldsymbol{A})^T\,\boldsymbol{b}.$$

See Appendix B for a proof of Theorem 2.

The following section illustrates the usefulness of Theorems 1 and 2 for assembling non-conjugate variational message passing algorithms involving Multivariate Normal factors.

5. Illustrations

We now provide illustrations of non-conjugate variational message passing that use Multivariate Normal updates. The first Illustration involves a Poisson mixed model and simulated data. We show, in detail, how Theorems 1 and 2 lead to a simple variational algorithm for such models. The second illustration involves heteroscedastic additive model analysis of data from an air pollution study, using non-conjugate variational message passing methodology with Multivariate Normal factors, recently developed by Menictas and Wand (2014).

5.1 Poisson Mixed Model

Consider the single variance component Poisson mixed model:

$$\begin{aligned} &y_i \,|\, \boldsymbol{\beta}, \boldsymbol{u} \quad \text{independently distributed as} \quad \text{Poisson}[\exp\{(\boldsymbol{X}\,\boldsymbol{\beta} + \boldsymbol{Z}\,\boldsymbol{u})_i\}], \quad 1 \le i \le n, \\ &\boldsymbol{u} \,|\, \boldsymbol{\sigma}^2 \sim N(\boldsymbol{0}, \sigma^2\,\boldsymbol{I}_K), \quad \sigma \sim \text{Half-Cauchy}(A) \quad \text{and} \quad \boldsymbol{\beta} \sim N(\boldsymbol{0}, \sigma_\beta^2 \boldsymbol{I}_p), \end{aligned} \tag{8}$$

where $\boldsymbol{X}$ is a $n \times p$ fixed effects design matrix, $\boldsymbol{Z}$ is a $n \times K$ random effects design matrix and $\sigma \sim \text{Half-Cauchy}(A)$ means that

$$p(\sigma) = \frac{2}{\pi A\{1 + (\sigma/A)^2\}}, \qquad \sigma > 0.$$

Note that, courtesy of Result 5 of Wand et al. (2011), one can replace $\sigma \sim \text{Half-Cauchy}(A)$ by the more convenient auxiliary variable representation

$$\sigma^2 |\, a \sim \text{Inverse-Gamma}(\tfrac{1}{2}, 1/a), \qquad a \sim \text{Inverse-Gamma}(\tfrac{1}{2}, 1/A^2),$$

where $v \sim \text{Inverse-Gamma}(A, B)$ means that

$$p(v) = \frac{B^A}{\Gamma(A)}\, v^{-A-1}\, \exp(-B/v), \qquad v > 0.$$

Consider the mean field approximation

$$p(\sigma^2, a, \boldsymbol{\beta}, \boldsymbol{u}, |\boldsymbol{y}) \approx q(\sigma^2)\, q(a)\, q(\boldsymbol{\beta}, \boldsymbol{u}; \boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}, \boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}) \tag{9}$$

where

$$q(\boldsymbol{\beta}, \boldsymbol{u}; \boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}, \boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}) \quad \text{is the} \quad N(\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}, \boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}) \quad \text{density function.}$$

Then application of (2) leads to the optimal q-densities for σ^2 and a being such that

$$\begin{aligned} &q^*(\sigma^2) \text{ is an Inverse-Gamma}(\tfrac{1}{2}(K+1), B_{q(\sigma^2)}) \text{ density function} \\ \text{and} \quad &q^*(a) \text{ is an Inverse-Gamma}(1, B_{q(a)}) \text{ density function} \end{aligned}$$

for rate parameters $B_{q(\sigma^2)}$ and $B_{q(a)}$. Let

$$\mu_{q(1/\sigma^2)} = E_{q(\sigma^2)}(1/\sigma^2) = \tfrac{1}{2}(K+1)/B_{q(\sigma^2)}$$

and $\mu_{q(1/a)}$ be defined similarly. Also let $\boldsymbol{\mu}_{q(\boldsymbol{u})}$ and $\boldsymbol{\Sigma}_{q(\boldsymbol{u})}$ be mean vector and covariance matrix of $q^*(\boldsymbol{u})$. Lastly, let

$$\boldsymbol{C} = [\boldsymbol{X}\ \boldsymbol{Z}].$$

Algorithm 1 provides explicit forms of the updates required to obtain the optimal parameters of $q^*(\boldsymbol{\beta}, \boldsymbol{u})$, $q^*(a)$ and $q^*(\sigma^2)$.

The derivation of Algorithm 1 is given in Appendix C. The approximate marginal log-likelihood admits the explicit expression:

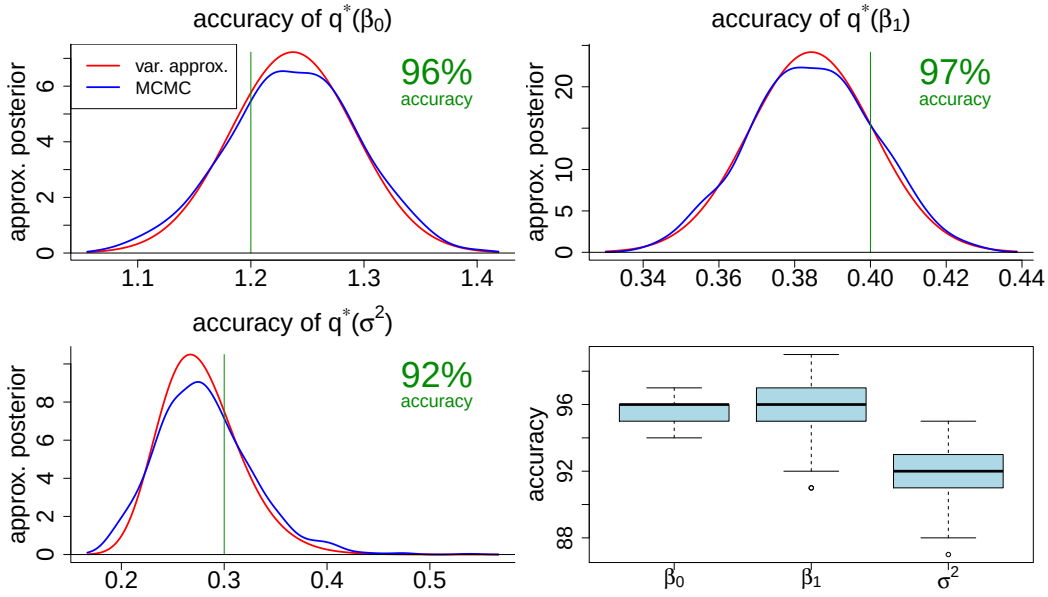


Figure 1: Upper panels and lower left panel: approximate posterior density functions for β_0, β_1 and σ^2 based on the variational approximation scheme described by Algorithm 1 and MCMC, for the first replication of the simulation study described in the text. Accuracy values, according to (11), with the exact posterior density function replaced by the MCMC-based posterior density function are also given. Lower right panel: Side-by-side boxplots of all 1000 accuracy values obtained for each parameter in the simulation study.

$$\begin{aligned}\log \underline{p}(\boldsymbol{y};q) \;=\; & \tfrac{1}{2}(K+p)+\log\Gamma(\tfrac{1}{2}(K+1))-\log(\pi)-\log(A)-\mathbf{1}^T\log(\boldsymbol{y}!)-\tfrac{1}{2}p\log(\sigma^2_{\boldsymbol{\beta}}) \\ & +\boldsymbol{y}^T\boldsymbol{C}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}-\mathbf{1}^T\exp\left\{\boldsymbol{C}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}+\tfrac{1}{2}\text{diagonal}(\boldsymbol{C}\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}\boldsymbol{C}^T)\right\} \\ & -\tfrac{1}{2\sigma^2_{\boldsymbol{\beta}}}\{\|\boldsymbol{\mu}_{q(\boldsymbol{\beta})}\|^2+\text{tr}(\boldsymbol{\Sigma}_{q(\boldsymbol{\beta})})\}+\tfrac{1}{2}\log|\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}| \\ & -\tfrac{1}{2}(K+1)\log\left(\tfrac{1}{2}\{\|\boldsymbol{\mu}_{q(\boldsymbol{u})}\|^2+\text{tr}(\boldsymbol{\Sigma}_{q(\boldsymbol{u})})\}+\mu_{q(1/a)}\right) \\ & -\log(\mu_{q(1/\sigma^2)}+A^{-2})+\mu_{q(1/\sigma^2)}\mu_{q(1/a)}.\end{aligned}$$

It is noteworthy that the variational message passing algorithm with derived variables, as described in Appendix A of Minka and Winn (2008), leads to an alternative to Algorithm 1 that requires only Univariate Normal updates corresponding to (7) of Knowles and Minka (2011). Such an approach is used in the Infer.NET computational framework (Minka et al., 2013). However, this alternative version is not as succinct as Algorithm 1. The simplified version that arises from Theorems 1 and 2 allows easier extension to more complicated models.

Initialize: $\mu_{q(1/\sigma^2)} > 0$, $\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}$ a $(p+K)\times 1$ vector and $\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}$ a $(p+K)\times(p+K)$ positive definite matrix.
Cycle:

$$\boldsymbol{w}_{q(\boldsymbol{\beta},\boldsymbol{u})} \leftarrow \exp\left\{\boldsymbol{C}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} + \tfrac{1}{2}\text{diagonal}(\boldsymbol{C}\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}\boldsymbol{C}^T)\right\}$$

$$\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})} \leftarrow \left(\boldsymbol{C}^T\text{diag}\{\boldsymbol{w}_{q(\boldsymbol{\beta},\boldsymbol{u})}\}\,\boldsymbol{C} + \begin{bmatrix} \sigma_{\boldsymbol{\beta}}^{-2}\boldsymbol{I}_p & \boldsymbol{0} \\ \boldsymbol{0} & \mu_{q(1/\sigma^2)}\boldsymbol{I}_K \end{bmatrix}\right)^{-1}$$

$$\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} \leftarrow \boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} + \boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}\left\{\boldsymbol{C}^T(\boldsymbol{y} - \boldsymbol{w}_{q(\boldsymbol{\beta},\boldsymbol{u})}) - \begin{bmatrix} \sigma_{\boldsymbol{\beta}}^{-2}\boldsymbol{I}_p & \boldsymbol{0} \\ \boldsymbol{0} & \mu_{q(1/\sigma^2)}\boldsymbol{I}_K \end{bmatrix}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}\right\}$$

$$\mu_{q(1/\sigma^2)} \leftarrow \frac{K+1}{2\mu_{q(1/a)} + \|\boldsymbol{\mu}_{q(\boldsymbol{u})}\|^2 + \text{tr}(\boldsymbol{\Sigma}_{q(\boldsymbol{u})})} \quad ; \quad \mu_{q(1/a)} \leftarrow 1/(\mu_{q(1/\sigma^2)} + A^{-2}).$$

until the absolute change in $\underline{p}(\boldsymbol{y};q)$ is negligible.

Algorithm 1: Iterative scheme for determination of the optimal parameters in $q^*(\boldsymbol{\beta},\boldsymbol{u})$, $q^*(\sigma^2)$ and $q^*(a)$ for the posterior density function approximation (9).

I replicated 1000 data-sets corresponding to the simulation setting

$$\begin{gathered} y_{ij}|U_i \sim \text{Poisson}\left(\exp(\beta_0 + \beta_1\, x_{ij} + U_i)\right),\ U_i|\sigma^2 \sim N(0,\sigma^2), \\ 1 \le i \le m,\ 1 \le j \le n, \\ \sigma^2|a \sim \text{Inverse-Gamma}(\tfrac{1}{2}, 1/a), \quad a \sim \text{Inverse-Gamma}(\tfrac{1}{2}, A^{-2}), \quad \boldsymbol{\beta} \sim N(\boldsymbol{0}, \sigma_{\boldsymbol{\beta}}^2\,\boldsymbol{I}) \end{gathered} \tag{10}$$

The hyperparameters were set at $\sigma_{\boldsymbol{\beta}} = A = 10^5$ and the sample sizes were $m = 100$, $n = 10$. Note that (10) is a special case of (8) with $\boldsymbol{Z} = \boldsymbol{I}_m \otimes \boldsymbol{1}_n$, where $\boldsymbol{1}_n$ is the $n \times 1$ vector with all entries equal to one.

For each data-set I obtained approximate posterior density functions for β_0, β_1 and σ^2 using both Algorithm 1 and Markov chain Monte Carlo (MCMC). For MCMC I used the package `BRugs` (Ligges et al., 2012) within the R computing environment (R Development Core Team, 2013) with a burnin of size 5000 followed by the generation of 5000 samples, with a thinning factor of 5. This resulted in MCMC samples of size 1000 being retained for inference. The iterations in Algorithm 1 were terminated when the relative change in $\log \underline{p}(\boldsymbol{y};q)$ fell below 10^{-4}.

Figure 1 displays side-by-side boxplots of accuracy scores defined by

$$\text{accuracy}(q^*) = 100\left(1 - \tfrac{1}{2}\int_{-\infty}^{\infty} \left|\, q^*(\theta) - p(\theta|\boldsymbol{y}) \,\right| d\theta\right)\%. \tag{11}$$

for a generic parameter θ, and with $p(\theta|\boldsymbol{y})$ replaced by a kernel density estimate based on the MCMC sample. The boxplots show that the majority of accuracy scores exceed 95%, and that they rarely drop below 90%.

Figure 1 allows visual assessment of the variational approximate posterior density functions against the MCMC-based benchmark for a single replication of the simulation study. The accuracy is seen to be excellent for β_0 and β_1 and very good for σ^2.

As discussed in Knowles and Minka (2011), convergence of non-conjugate variational message passing is not guaranteed. In the simulation study the algorithm converged in all replications regardless of starting values, but in about 2% of the cases this required some adjustment to avoid inverting a singular matrix in the $\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}$ update during the early iterations. The adjustment involves adding $\varepsilon \boldsymbol{I}$ to the matrix requiring inversion, with $\varepsilon > 0$ chosen so that the condition number stayed below 10^{16}. In almost all cases, this adjustment was only necessary for the first few iterations.

In this section we have shown that non-conjugate variational message passing leads to an attractive variational inference algorithm for Poisson mixed models. Since the exponential moments of Multivariate Normal random vectors are available in closed form, no quadrature is required in the Poisson case. Other generalized linear mixed models, such as logistic mixed models, require quadrature. The logistic analogue of (8) is such that only univariate quadrature is required. Details are given in Appendix B of Tan and Nott (2013).

5.2 Heteroscedastic Additive Model

This illustration involves analysis of data from the Californian air pollution study described in Breiman and Friedman (1985). The response variable is

$$y = \text{ozone concentration (ppm) at Sandburg Air Force Base}$$

and three predictors variables are

$$\begin{aligned} x_1 &= \text{pressure gradient (mm Hg) from Los Angeles International Airport to Daggett, California,} \\ x_2 &= \text{inversion base height (feet)} \\ \text{and } x_3 &= \text{inversion base temperature (degrees Fahrenheit).} \end{aligned}$$

The data comprises 345 measurements on each of these 4 variables. Let $(x_{i1}, x_{i3}, x_{i3}, y_i)$, $1 \leq i \leq 345$ denote the full regression data set.

We entertained the *heteroscedastic additive model*

$$y_i \sim N\Big(\beta_0 + f_1(x_{1i}) + f_2(x_{2i}) + f_3(x_{3i}), \exp\big(\gamma_0 + h_1(x_{1i}) + h_2(x_{2i}) + h_3(x_{3i})\big)\Big), \qquad (12)$$

for $1 \leq i \leq 345$. Here f_j and g_j, $j = 1, 2, 3$, and smooth but otherwise arbitrary functions. Bayesian mixed model-based penalized splines (e.g. Ruppert et al., 2003) were used to model the smooth functions as follows:

$$\begin{aligned} f_j(x) &= \beta_j\, x + \sum_{k=1}^{K_j} u_{jk}\, z_{jk}(x), \quad u_{jk} \text{ iid } N(0, \sigma_{uj}^2) \\ \text{and} \quad h_j(x) &= \gamma_j\, x + \sum_{k=1}^{K_j} v_{jk}\, z_{jk}(x), \quad v_{jk} \text{ iid } N(0, \sigma_{vj}^2). \end{aligned} \qquad (13)$$

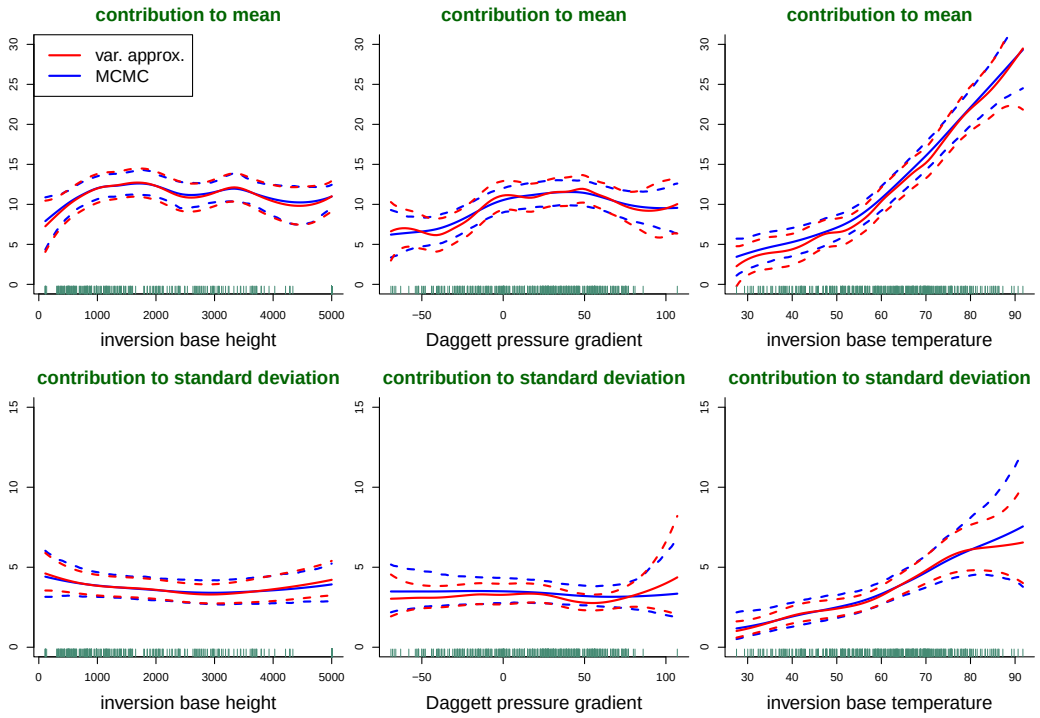


Figure 2: Upper panels: approximate pointwise posterior means and 95% credible sets for the mean function contributions f_1, f_2 and f_3 according to the heteroscedastic additive model (12). Vertical alignment of the estimated functions is described in the text. Lower panels: approximate pointwise posterior means and 95% credible sets for the standard deviation function contributions $\exp(h_2/2)$, $\exp(h_2/2)$ and $\exp(h_3/2)$. Approximate Bayesian inference is based on both non-conjugate variational message passing and MCMC.

where iid stands for 'independently and identically distributed as'. The $\{z_{jk} : 1 \le k \le K_j\}$, $j = 1, 2, 3$, are spline bases of sizes K_j respectively. My default for the z_{jk} are suitably transformed cubic O'Sullivan splines, as described in Section 4 of Wand and Ormerod (2008). The priors on the regression coefficients and standard deviation parameters are

$$\beta_j \text{ iid } N(0, \sigma^2_{\boldsymbol{\beta}}),\ \gamma_j \text{ iid } N(0, \sigma^2_{\boldsymbol{\gamma}}),\ \sigma_{uj} \text{ iid Half-Cauchy}(A_u),\ \sigma_{vj} \text{ iid Half-Cauchy}(A_v). \quad (14)$$

The regression data replaced by standardized versions and the hyperparameters were set to be $\sigma_{\boldsymbol{\beta}} = \sigma_{\boldsymbol{\gamma}} = A_u = A_v = 10^5$, corresponding to non-informativity. The results were transformed to the original units after fitting. The basis function sizes were all fixed at $K_1 = K_2 = K_3 = 18$.

The Bayesian model given by (12), (13) and (14) admits a closed form non-conjugate variational message passing algorithm, with the regression coefficients for the full mean and variance functions each being Multivariate Normal. Details are given in Menictas and Wand (2014). Figure 2 shows the estimated mean function (f_j) contributions, and the standard deviation function ($\exp(h_j/2)$) contributions based on both variational approximation

and MCMC. The MCMC inference was carried out in the same fashion as for the illustration described in Section 5.1. The abbreviated names `inversion base height`, `Daggett pressure gradient` and `inversion base temperature` are used for x_1, x_2 and x_3. The estimated f_1 display is vertically aligned to match the response data by evaluating the estimate of f_2 at $\overline{x}_2$ and estimate of f_3 at $\overline{x}_3$, where $\overline{x}_2$ and $\overline{x}_3$ are the sample means of the x_{2i} and x_{3i}, respectively. Analogous alignment strategies were used for the f_2 and f_3 displays.

Figure 2 shows that there is excellent agreement between non-conjugate variational message passing, with Multivariate Normal coefficient vectors, and MCMC. The former approach is considerably faster. The heteroscedasticity is seen to be relatively mild for `inversion base height` and `Daggett pressure gradient`. However, there is pronounced heteroscedasticity in `inversion base temperature` that is captured by model (12).

Acknowledgments

This research was partially supported by Australian Research Council Discovery Project DP110100061. The author is grateful to Cathy Lee, Jan Luts, Marianne Menictas, Tom Minka, David Nott and Linda Tan for their comments.

Appendix A: Proof of Theorem 1

Proof of (a)

For the upper-left block of $\boldsymbol{U}$ note that $\boldsymbol{\mu} = \boldsymbol{\Sigma}\,\boldsymbol{\eta}_1$ and so $d_{\boldsymbol{\eta}_1}\boldsymbol{\mu} = \boldsymbol{\Sigma}\,d\boldsymbol{\eta}_1$. Theorem 6 of Magnus and Neudecker (1999) leads to $\mathsf{D}_{\boldsymbol{\eta}_1}\boldsymbol{\mu} = \boldsymbol{\Sigma}$. The lower-right block of $\boldsymbol{U}$ involves the relation $\boldsymbol{\Sigma} = -\frac{1}{2}\{\mathrm{vec}^{-1}(\boldsymbol{D}_d^{+T}\boldsymbol{\eta}_2)\}^{-1}$ given in (5). Then Rule 3.3.5 in Wand (2002) and the identity

$$\mathrm{vec}(\boldsymbol{ABC}) = (\boldsymbol{C}^T \otimes \boldsymbol{A})\,\mathrm{vec}(\boldsymbol{B}) \tag{15}$$

leads to

$$d_{\boldsymbol{\eta}_2}\,\mathrm{vec}(\boldsymbol{\Sigma}) = 2\,\mathrm{vec}(\boldsymbol{\Sigma}\{\mathrm{vec}^{-1}(\boldsymbol{D}_d^{+T}\,d\boldsymbol{\eta}_2)\}\,\boldsymbol{\Sigma}) = 2(\boldsymbol{\Sigma}\otimes\boldsymbol{\Sigma})\,\boldsymbol{D}_d^{+T}\,d\boldsymbol{\eta}_2.$$

Hence, making use of Theorem 13 (b), Chapter 3, of Magnus and Neudecker (1999),

$$\mathsf{D}_{\boldsymbol{\eta}_2}\,\mathrm{vec}(\boldsymbol{\Sigma}) = 2(\boldsymbol{\Sigma}\otimes\boldsymbol{\Sigma})\,\boldsymbol{D}_d^{+T} = 2\boldsymbol{D}_d\boldsymbol{D}_d^{+}(\boldsymbol{\Sigma}\otimes\boldsymbol{\Sigma})\,\boldsymbol{D}_d^{+T} = \boldsymbol{D}_d\boldsymbol{S} = (\boldsymbol{S}\boldsymbol{D}_d^{T})^T.$$

The expression for the lower-left block of $\boldsymbol{U}$ follows from

$$d_{\boldsymbol{\eta}_2}\,\boldsymbol{\mu} = 2\boldsymbol{\Sigma}\{\mathrm{vec}^{-1}(\boldsymbol{D}_d^{+T}\,d\boldsymbol{\eta}_2)\}\,\boldsymbol{\Sigma}\boldsymbol{\eta}_1 = 2\,\mathrm{vec}(\boldsymbol{\Sigma}\{\mathrm{vec}^{-1}(\boldsymbol{D}_d^{+T}\,d\boldsymbol{\eta}_2)\}\,\boldsymbol{\mu}) = 2(\boldsymbol{\mu}^T\otimes\boldsymbol{\Sigma})\boldsymbol{D}_d^{+T}\,d\boldsymbol{\eta}_2$$

where Rule 3.3.5 in Wand (2002) and (15) have been used again. This gives

$$\mathsf{D}_{\boldsymbol{\eta}_2}\,\boldsymbol{\mu} = 2(\boldsymbol{\mu}^T\otimes\boldsymbol{\Sigma})\boldsymbol{D}_d^{+T} = \{2\boldsymbol{D}_d^{+}(\boldsymbol{\mu}\otimes\boldsymbol{\Sigma})\}^T = \{2\boldsymbol{D}_d^{+}(\boldsymbol{\mu}\otimes\boldsymbol{I}_d)(1\otimes\boldsymbol{\Sigma})\}^T = (\boldsymbol{M}\boldsymbol{\Sigma})^T.$$

For the upper-left block note that, from (5), $d_{\boldsymbol{\eta}_1}\mathrm{vec}(\boldsymbol{\Sigma}) = \boldsymbol{0}\,d_{\boldsymbol{\eta}_1}$ and so $\mathsf{D}_{\boldsymbol{\eta}_1}\mathrm{vec}(\boldsymbol{\Sigma}) = \boldsymbol{0}$.

Proof of (b)

This is an immediate consequence of Theorem 13(d), Chapter 3, of Magnus and Neudecker (1999).

Proof of (c)

The upper-left block is $\text{var}(\boldsymbol{x}) = \boldsymbol{\Sigma}$. The lower-right block is

$$\begin{aligned}\text{var}\{\text{vech}(\boldsymbol{x}\boldsymbol{x}^T)\} &= \boldsymbol{D}_d^+ \,\text{var}\{\text{vec}(\boldsymbol{x}1\boldsymbol{x}^T)\}\boldsymbol{D}_d^{+T} = \boldsymbol{D}_d^+ \,\text{var}(\text{vec}(\boldsymbol{x}\otimes\boldsymbol{x}^T))\boldsymbol{D}_d^{+T} \\ &= \boldsymbol{D}_d^+ \,(\boldsymbol{I}_{d^2} + \boldsymbol{K}_d)(\boldsymbol{\Sigma}\otimes\boldsymbol{\Sigma} + \boldsymbol{\Sigma}\otimes\boldsymbol{\mu}\boldsymbol{\mu}^T + \boldsymbol{\mu}\boldsymbol{\mu}^T\otimes\boldsymbol{\Sigma})\boldsymbol{D}_d^{+T}\end{aligned}$$

where (15) and Theorem 4.3 (iv) of Magnus and Neudecker (1979) has been used. Here $\boldsymbol{K}_d$ denotes the *commutation matrix* of order d, defined by $\boldsymbol{K}_d(\boldsymbol{A}\otimes\boldsymbol{B}) = (\boldsymbol{B}\otimes\boldsymbol{A})\boldsymbol{K}_d$ for arbitrary $d\times d$ matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. Noting the identity $\frac{1}{2}\,\boldsymbol{D}_d^+(\boldsymbol{I}_{d^2}+\boldsymbol{K}_d) = \boldsymbol{D}_d^+$, which is an immediate consequence of (15) in Chapter 3 of Magnus and Neudecker (1999), one then gets

$$\text{var}\{\text{vech}(\boldsymbol{x}\boldsymbol{x}^T)\} = \boldsymbol{S} + 2\boldsymbol{D}_d^+(\boldsymbol{\Sigma}\otimes\boldsymbol{\mu}\boldsymbol{\mu}^T + \boldsymbol{\mu}\boldsymbol{\mu}^T\otimes\boldsymbol{\Sigma})\boldsymbol{D}_d^{+T}.$$

Theorem 12(a), Chapter 3, of Magnus and Neudecker (1999) states that $\boldsymbol{K}_d\boldsymbol{D}_d = \boldsymbol{D}_d$, which implies that

$$\boldsymbol{D}_d^+ = \{(\boldsymbol{K}_d\boldsymbol{D}_d)^T\boldsymbol{K}_d\boldsymbol{D}_d\}^{-1}\boldsymbol{D}_d^T\boldsymbol{K}_d^T = (\boldsymbol{D}_d^T\boldsymbol{K}_d^T\boldsymbol{K}_d\boldsymbol{D}_d)^{-1}\boldsymbol{D}_d^T\boldsymbol{K}_d = \boldsymbol{D}_d^+\boldsymbol{K}_d. \qquad (16)$$

Here I have used $\boldsymbol{K}^T = \boldsymbol{K}_d^{-1} = \boldsymbol{K}_d$ as stated in (2) of Chapter 3 of Magnus and Neudecker (1999). The identity $\boldsymbol{D}_d^+ = \boldsymbol{D}_d^+\boldsymbol{K}_d$ leads to

$$\boldsymbol{D}_d^+(\boldsymbol{\Sigma}\otimes\boldsymbol{\mu}\boldsymbol{\mu}^T)\boldsymbol{D}_d^{+T} = \boldsymbol{D}_d^+\boldsymbol{K}_d(\boldsymbol{\Sigma}\otimes\boldsymbol{\mu}\boldsymbol{\mu}^T)\boldsymbol{D}_d^{+T} = \boldsymbol{D}_d^+(\boldsymbol{\mu}\boldsymbol{\mu}^T\otimes\boldsymbol{\Sigma})\boldsymbol{K}_d^T\boldsymbol{D}_d^{+T} = \boldsymbol{D}_d^+(\boldsymbol{\mu}\boldsymbol{\mu}^T\otimes\boldsymbol{\Sigma})\boldsymbol{D}_d^{+T}$$

leading to $\text{var}\{\text{vech}(\boldsymbol{x}\boldsymbol{x}^T)\} = \boldsymbol{S} + 4\boldsymbol{D}_d^+(\boldsymbol{\mu}\boldsymbol{\mu}^T\otimes\boldsymbol{\Sigma})\boldsymbol{D}_d^{+T}$. Since

$$(\boldsymbol{\mu}\boldsymbol{\mu}^T\otimes\boldsymbol{\Sigma}) = (\boldsymbol{\mu}\otimes\boldsymbol{\Sigma})(\boldsymbol{\mu}^T\otimes\boldsymbol{I}_d) = (\boldsymbol{\mu}\otimes\boldsymbol{I}_d)(1\otimes\boldsymbol{\Sigma})(\boldsymbol{\mu}^T\otimes\boldsymbol{I}_d) = (\boldsymbol{\mu}\otimes\boldsymbol{I}_d)\boldsymbol{\Sigma}(\boldsymbol{\mu}\otimes\boldsymbol{I}_d)^T.$$

I conclude that

$$\text{var}\{\text{vech}(\boldsymbol{x}\boldsymbol{x}^T)\} = \boldsymbol{S} + \boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T.$$

The (i,j) entry of the lower-left block is

$$\text{cov}(\text{vech}(\boldsymbol{x}\boldsymbol{x}^T)_i, x_j) = \text{cov}(\{\boldsymbol{D}_d^+\text{vec}(\boldsymbol{x}\boldsymbol{x}^T)\}_i, x_j) = \sum_{k=1}^{d^2}(\boldsymbol{D}_d^+)_{ik}\,\text{cov}(\text{vec}(\boldsymbol{x}\boldsymbol{x}^T)_k, x_j). \qquad (17)$$

Let $\lfloor x\rfloor$ denote the largest integer less than or equal to x. Then using one of the fundamental identities for generalized cumulants given on page 58 of McCullagh (1987),

$$\begin{aligned}\text{cov}(\text{vec}(\boldsymbol{x}\boldsymbol{x}^T)_k, x_j) &= \text{cov}(x_{k-d\lfloor(k-1)/d\rfloor}\, x_{\lfloor(k-1)/d\rfloor+1}\,,\, x_j) \\ &= \mu_{k-d\lfloor(k-1)/d\rfloor}\Sigma_{\lfloor(k-1)/d\rfloor+1,j} + \mu_{\lfloor(k-1)/d\rfloor+1}\Sigma_{k-d\lfloor(k-1)/d\rfloor,j} \\ &= (\boldsymbol{\Sigma}\otimes\boldsymbol{\mu})_{kj} + (\boldsymbol{\mu}\otimes\boldsymbol{\Sigma})_{kj}.\end{aligned}$$

Combining this with (17), the lower-left block equals $\boldsymbol{D}_d^+(\boldsymbol{\Sigma}\otimes\boldsymbol{\mu} + \boldsymbol{\mu}\otimes\boldsymbol{\Sigma})$. But, courtesy of (16), this equals

$$\boldsymbol{D}_d^+(\boldsymbol{\mu}\otimes\boldsymbol{\Sigma}) + \boldsymbol{D}_d^+\boldsymbol{K}_d(\boldsymbol{\Sigma}\otimes\boldsymbol{\mu}) = 2\boldsymbol{D}_d^+(\boldsymbol{\mu}\otimes\boldsymbol{\Sigma}) = \boldsymbol{M}\boldsymbol{\Sigma}.$$

Proof of (d)

It is straightforward to verify that

$$\begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Sigma}\boldsymbol{M}^T \\ \boldsymbol{M}\boldsymbol{\Sigma} & \boldsymbol{S} + \boldsymbol{M}\boldsymbol{\Sigma}\boldsymbol{M}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}^{-1} + \boldsymbol{M}^T\boldsymbol{S}^{-1}\boldsymbol{M} & -\boldsymbol{M}^T\boldsymbol{S}^{-1} \\ -\boldsymbol{S}^{-1}\boldsymbol{M} & \boldsymbol{S}^{-1} \end{bmatrix} = \boldsymbol{I}_{d+d(d+1)/2}.$$

The stated expression for $\boldsymbol{V}^{-1}$ immediately follows.

Proof of (e)

$$\boldsymbol{V}^{-1}\boldsymbol{U} = \begin{bmatrix} \boldsymbol{\Sigma}^{-1} + \boldsymbol{M}^T\boldsymbol{S}^{-1}\boldsymbol{M} & -\boldsymbol{M}^T\boldsymbol{S}^{-1} \\ -\boldsymbol{S}^{-1}\boldsymbol{M} & \boldsymbol{S}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} \\ \boldsymbol{M}\boldsymbol{\Sigma} & \boldsymbol{S}\boldsymbol{D}_d^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{M}^T\boldsymbol{D}_d^T \\ \boldsymbol{0} & \boldsymbol{D}_d^T \end{bmatrix}.$$

Proof of (f)

First note that

$$\boldsymbol{V}^{-1}\boldsymbol{U} \begin{bmatrix} \boldsymbol{g} \\ \mathrm{vec}(\boldsymbol{G}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{M}^T\boldsymbol{D}_d^T \\ \boldsymbol{0} & \boldsymbol{D}_d^T \end{bmatrix} \begin{bmatrix} \boldsymbol{g} \\ \mathrm{vec}(\boldsymbol{G}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{g} - \boldsymbol{M}^T\boldsymbol{D}_d^T\,\mathrm{vec}(\boldsymbol{G}) \\ \boldsymbol{D}_d^T\mathrm{vec}(\boldsymbol{G}) \end{bmatrix}.$$

With the help of Lemma 1 and (15) one then has

$$\boldsymbol{M}^T\boldsymbol{D}_d^T\,\mathrm{vec}(\boldsymbol{G}) = 2(\boldsymbol{\mu}^T \otimes \boldsymbol{I}_d)\boldsymbol{D}_d^{+T}\boldsymbol{D}_d^T\mathrm{vec}(\boldsymbol{G}) = 2(\boldsymbol{\mu}^T \otimes \boldsymbol{I}_d)\,\mathrm{vec}(\boldsymbol{G}) = 2\,\boldsymbol{G}\boldsymbol{\mu}$$

and the stated result is obtained.

Appendix B: Proof of Theorem 2

Proof of (a)

Let A_{ij} and B_{ij}, respectively, denote the (i,j) entry of $\boldsymbol{A}$ and $\boldsymbol{B}$. Then a listing of the entries of $\mathcal{Q}(\boldsymbol{A})$ reveals that its entry is (i,j) is

$$\mathcal{Q}(\boldsymbol{A})_{ij} = A_{i,\lfloor (j-1)/d \rfloor + 1}\, A_{i,j-d\lfloor (j-1)/d \rfloor}, \quad 1 \le i \le n,\ 1 \le j \le d^2. \tag{18}$$

Similarly, the ith entry of $\mathrm{vec}(\boldsymbol{B})$ is

$$\mathrm{vec}(\boldsymbol{B})_j = B_{j-d\lfloor (j-1)/d \rfloor, \lfloor (j-1)/d \rfloor + 1}, \quad 1 \le j \le d^2. \tag{19}$$

Hence

$$\begin{aligned}
\{\mathcal{Q}(\boldsymbol{A})\,\mathrm{vec}(\boldsymbol{B})\}_i &= \sum_{j=1}^{d^2} A_{i,\lfloor (j-1)/d \rfloor + 1}\, A_{i,j-d\lfloor (j-1)/d \rfloor}\, B_{j-d\lfloor (j-1)/d \rfloor, \lfloor (j-1)/d \rfloor + 1} \\
&= \left(\sum_{j=1}^{d} + \sum_{j=d+1}^{2d} + \ldots + \sum_{j=(d-1)\,d+1}^{d^2} \right) A_{i,\lfloor (j-1)/d \rfloor + 1}\, A_{i,j-d\lfloor (j-1)/d \rfloor} \\
&\qquad \times B_{j-d\lfloor (j-1)/d \rfloor, \lfloor (j-1)/d \rfloor + 1} \\
&= \sum_{j=1}^{d} A_{i1}A_{ij}B_{1j} + \sum_{j=1}^{d} A_{i2}A_{ij}B_{2j} + \ldots + \sum_{j=1}^{d} A_{id}A_{ij}B_{dj} \\
&= \sum_{j=1}^{d}\sum_{j'=1}^{d} A_{ij}A_{ij'}B_{jj'} = \mathrm{diagonal}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{A}^T)_i
\end{aligned}$$

and the result follows immediately.

Proof of (b)

Letting b_i denote the ith entry of $\boldsymbol{b}$ and making use of (18) one has

$$\begin{aligned}\{\mathcal{Q}(\boldsymbol{A})^T\,\boldsymbol{b}\}_j &= \sum_{i=1}^{n}\{\mathcal{Q}(\boldsymbol{A})^T\}_{ji}\,b_i = \sum_{i=1}^{n}\mathcal{Q}(\boldsymbol{A})_{ij}\,b_i \\ &= \sum_{i=1}^{n} b_i\,A_{i,\lfloor(j-1)/d\rfloor+1}\,A_{i,j-d\lfloor(j-1)/d\rfloor}.\end{aligned}$$

Application of (19) to $\boldsymbol{A}^T\mathrm{diag}(\boldsymbol{b})\boldsymbol{A}$ gives

$$\begin{aligned}\mathrm{vec}(\boldsymbol{A}^T\mathrm{diag}(\boldsymbol{b})\boldsymbol{A})_j &= (\boldsymbol{A}^T\mathrm{diag}(\boldsymbol{b})\boldsymbol{A})_{j-d\lfloor(j-1)/d\rfloor,\,\lfloor(j-1)/d\rfloor+1} \\ &= \sum_{i=1}^{n} b_i(\boldsymbol{A}^T)_{j-d\lfloor(j-1)/d\rfloor,i}\boldsymbol{A}_{i,\lfloor(j-1)/d\rfloor+1} \\ &= \sum_{i=1}^{n} b_i\,A_{i,\lfloor(j-1)/d\rfloor+1}\,A_{i,j-d\lfloor(j-1)/d\rfloor} = \{\mathcal{Q}(\boldsymbol{A})^T\,\boldsymbol{b}\}_j\end{aligned}$$

which proves equality between $\mathcal{Q}(\boldsymbol{A})^T\,\boldsymbol{b}$ and $\mathrm{vec}(\boldsymbol{A}^T\mathrm{diag}(\boldsymbol{b})\boldsymbol{A})$.

Appendix C: Derivation of Algorithm 1

Derivation of $q^*(\sigma^2)$

$$\begin{aligned}\log q^*(\sigma^2) &= E_q\{\log p(\sigma^2|\text{rest})\} + \text{const} \\ &= \{-\tfrac{1}{2}(K+1)-1\}\log(\sigma^2) - \{\tfrac{1}{2}E_q\|\boldsymbol{u}\|^2 + \mu_{q(1/a)}\}/\sigma^2 + \text{const.}\end{aligned}$$

where 'const' denotes terms not involving σ^2. Using

$$E_q\|\boldsymbol{u}\|^2 = \|\boldsymbol{\mu}_{q(\boldsymbol{u})}\|^2 + \mathrm{tr}(\boldsymbol{\Sigma}_{q(\boldsymbol{\mu})})$$

I then get $q^*(\sigma^2) \sim \text{Inverse-Gamma}(\frac{1}{2}(K+1), B_{q(\sigma^2)})$ where

$$B_{q(\sigma^2)} = \tfrac{1}{2}\{\|\boldsymbol{\mu}_{q(\boldsymbol{u})}\|^2 + \mathrm{tr}(\boldsymbol{\Sigma}_{q(\boldsymbol{\mu})})\} + \mu_{q(1/a)}.$$

Derivation of $q^*(a)$

$$\begin{aligned}\log q^*(a) &= E_q\{\log p(a|\text{rest})\} + \text{const} \\ &= (-1-1)\log(a) - (\mu_{q(1/\sigma^2)} + A^{-2})/a + \text{const.}\end{aligned}$$

This gives $q^*(a) \sim \text{Inverse-Gamma}(1, B_{q(a)})$ where

$$B_{q(a)} = \mu_{q(1/\sigma^2)} + A^{-2}.$$

Derivation of the $(\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}, \boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})})$ Updates

Note that

$$\begin{aligned} E_q\{\log p(\boldsymbol{y},\boldsymbol{\beta},\boldsymbol{u},\sigma^2,a)\} &= E_q\{\log\, p(\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{u}) + \log\, p(\boldsymbol{\beta},\boldsymbol{u}|\sigma^2) + \log\, p(\sigma^2|a) + \log\, p(a)\} \\ &= S + \text{terms not involving } \boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} \text{ or } \boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})} \end{aligned}$$

where

$$S \equiv S(\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}, \boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}) \equiv E_q\{\log\, p(\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{u}) + \log\, p(\boldsymbol{\beta},\boldsymbol{u}|\sigma^2)\}.$$

Then

$$\begin{aligned} S &= \boldsymbol{y}^T\boldsymbol{C}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} - \mathbf{1}^T \exp\Big\{\boldsymbol{C}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} + \tfrac{1}{2}\text{diagonal}(\boldsymbol{C}\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}\boldsymbol{C}^T)\Big\} \\ &\quad -\tfrac{1}{2}\text{tr}\left(\begin{bmatrix} \sigma_{\boldsymbol{\beta}}^{-2}\boldsymbol{I}_p & \mathbf{0} \\ \mathbf{0} & \mu_{q(1/\sigma^2)}\boldsymbol{I}_K \end{bmatrix}\{\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}^T + \boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}\}\right) \\ &\quad -\tfrac{1}{2}(p+K)\log(2\pi) - \tfrac{1}{2}\,p\log(\sigma_{\boldsymbol{\beta}}^2) - \tfrac{1}{2}\,K\,E_q\{\log(\sigma^2)\} - \mathbf{1}^T\log(\boldsymbol{y}!) \end{aligned}$$

and so

$$\begin{aligned} d_{\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}} S &= \boldsymbol{y}^T\boldsymbol{C}\, d\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} \\ &\quad -\mathbf{1}^T\text{diag}[\exp\{\boldsymbol{C}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} + \tfrac{1}{2}\text{diagonal}(\boldsymbol{C}\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}\boldsymbol{C}^T)\}]\boldsymbol{C}\, d\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} \\ &\quad -\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}^T\begin{bmatrix} \sigma_{\boldsymbol{\beta}}^{-2}\boldsymbol{I}_p & \mathbf{0} \\ \mathbf{0} & \mu_{q(1/\sigma^2)}\boldsymbol{I}_K \end{bmatrix} d\,\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} \\ &= \Big(\Big[\boldsymbol{y} - \exp\{\boldsymbol{C}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} + \tfrac{1}{2}\text{diagonal}(\boldsymbol{C}\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}\boldsymbol{C}^T)\}\Big]^T\boldsymbol{C} \\ &\quad -\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}^T\begin{bmatrix} \sigma_{\boldsymbol{\beta}}^{-2}\boldsymbol{I}_p & \mathbf{0} \\ \mathbf{0} & \mu_{q(1/\sigma^2)}\boldsymbol{I}_K \end{bmatrix}\Big) d\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}. \end{aligned}$$

Thus, by Theorem 6, Chapter 5, of Magnus and Neudecker (1999),

$$\begin{aligned} \{\mathsf{D}_{\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}} S\}^T &= \boldsymbol{C}^T\Big[\boldsymbol{y} - \exp\{\boldsymbol{C}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} + \tfrac{1}{2}\text{diagonal}(\boldsymbol{C}\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}\boldsymbol{C}^T)\}\Big] \\ &\quad -\begin{bmatrix} \sigma_{\boldsymbol{\beta}}^{-2}\boldsymbol{I}_p & \mathbf{0} \\ \mathbf{0} & \mu_{q(1/\sigma^2)}\boldsymbol{I}_K \end{bmatrix}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})}. \end{aligned}$$

Next, using Theorem 2 of Section 4 and Rule 3.3.2 of Wand (2002),

$$\begin{aligned} d_{\text{vec}(\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})})} S &= -\mathbf{1}^T\text{diag}[\exp\{\boldsymbol{C}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} + \tfrac{1}{2}\text{diagonal}(\boldsymbol{C}\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}\boldsymbol{C}^T)\}]\tfrac{1}{2}\mathcal{Q}(\boldsymbol{C})\, d\text{vec}(\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}) \\ &\quad -\tfrac{1}{2}\text{vec}\left(\begin{bmatrix} \sigma_{\boldsymbol{\beta}}^{-2}\boldsymbol{I}_p & \mathbf{0} \\ \mathbf{0} & \mu_{q(1/\sigma^2)}\boldsymbol{I}_K \end{bmatrix}\right)^T d\,\text{vec}(\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}) \\ &= \Big(-\tfrac{1}{2}\exp\{\boldsymbol{C}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} + \tfrac{1}{2}\text{diagonal}(\boldsymbol{C}\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}\boldsymbol{C}^T)\}^T\mathcal{Q}(\boldsymbol{C}) \\ &\quad -\tfrac{1}{2}\text{vec}\left(\begin{bmatrix} \sigma_{\boldsymbol{\beta}}^{-2}\boldsymbol{I}_p & \mathbf{0} \\ \mathbf{0} & \mu_{q(1/\sigma^2)}\boldsymbol{I}_K \end{bmatrix}\right)^T\Big) d\,\text{vec}(\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})} \\ &= -\tfrac{1}{2}\text{vec}\Big(\boldsymbol{C}^T\text{diag}[\exp\{\boldsymbol{C}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} + \tfrac{1}{2}\text{diagonal}(\boldsymbol{C}\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}\boldsymbol{C}^T)\}]\boldsymbol{C} \\ &\quad +\begin{bmatrix} \sigma_{\boldsymbol{\beta}}^{-2}\boldsymbol{I}_p & \mathbf{0} \\ \mathbf{0} & \mu_{q(1/\sigma^2)}\boldsymbol{I}_K \end{bmatrix}\Big)^T d\,\text{vec}(\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}) \end{aligned}$$

and so

$$\text{vec}^{-1}\left((\mathsf{D}_{\text{vec}(\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})})}\,S)^T\right) = -\tfrac{1}{2}\left(\boldsymbol{C}^T\text{diag}[\exp\{\boldsymbol{C}\boldsymbol{\mu}_{q(\boldsymbol{\beta},\boldsymbol{u})} + \tfrac{1}{2}\text{diagonal}(\boldsymbol{C}\boldsymbol{\Sigma}_{q(\boldsymbol{\beta},\boldsymbol{u})}\boldsymbol{C}^T)\}]\,\boldsymbol{C} + \begin{bmatrix} \sigma_{\boldsymbol{\beta}}^{-2}\boldsymbol{I}_p & \boldsymbol{0} \\ \boldsymbol{0} & \mu_{q(1/\sigma^2)}\boldsymbol{I}_K \end{bmatrix}\right).$$

References

M. Braun and J. McAuliffe. Variational inference for large-scale models of discrete choice. *Journal of the American Statistical Association*, 105:324–335, 2010.

L. Breiman and J. Friedman. Estimating optimal transformations for multiple regression and correlation (with discussion). *Journal of the American Statistical Association*, 80: 580–619, 1985.

J. Hensman, M. Rattray, and N. D. Lawrence. Fast variational inference in the conjugate exponential family. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2897–2905, 2012.

D. A. Knowles and T. P. Minka. Non-conjugate message passing for multinomial and binary regression. In J. Shawe-Taylor, R.S. Zamel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1701–1709, 2011.

U. Ligges, S. Sturtz, A. Gelman, G. Gorjanc, and C. Jackson. *BRugs. Fully-interactive R interface to the OpenBUGS software for Bayesian analysis using MCMC sampling*, 2012. URL `http://cran.r-project.org`. R package version 0.8-0.

J. R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics, Revised Edition*. Wiley, Chichester UK, 1999.

J.R. Magnus and H. Neudecker. The commutation matrix: some properties and applications. *The Annals of Statistics*, 7:381–394, 1979.

P. McCullagh. *Tensor Methods in Statistics*. Chapman and Hall, London, 1987.

M. Menictas and M. P. Wand. Variational inference for heteroscedastic semiparametric regression. 2014. Unpublished manuscript.

T. Minka and J. Winn. Gates: A graphical notation for mixture models. *Microsoft Research Technical Report Series*, MSR-TR-2008-185:1–16, 2008.

T. Minka, J. Winn, J. Guiver, and D. Knowles. *Infer.NET 2.5*, 2013. URL `http://research.microsoft.com/infernet`. Microsoft Research Cambridge.

F. Novomestky. *matrixcalc. A collection of functions to support matrix differential calculus as presented in Magnus and Neudecker (1999)*, 2008. URL `http://cran.r-project.org`. R package version 1.0-1.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. URL `http://www.R-project.org/`. ISBN 3-900051-07-0.

D. Ruppert, M. P. Wand, and R. J. Carroll. *Semiparametric Regression*. Cambridge University Press, New York, 2003.

L. S. L. Tan and D. J. Nott. Variational inference for generalized linear mixed models using partially noncentered parametrizations. *Statistical Science*, 28:168–188, 2013.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundation and Trends in Machine Learning*, 1:1–305, 2008.

M. P. Wand. Vector differential calculus in statistics. *The American Statistician*, 56:55–62, 2002.

M. P. Wand and J. T. Ormerod. On semiparametric regression with O'Sullivan penalized splines. *Australian and New Zealand Journal of Statistics*, 50:179–198, 2008.

M. P. Wand, J. T. Ormerod, S. A. Padoan, and R. Frühwirth. Mean field variational Bayes for elaborate distributions. *Bayesian Analysis*, 6(4):847–900, 2011.

C. Wang and D. M. Blei. Variational inference in nonconjugate models. *Journal of Machine Learning Research*, 14:1005–1031, 2013.